ECE 470: Final Project Report

Team ALANDR
Mario Perez (mariop2), David Stier (stier2),Yutong Xie (yutongx6), Yuelin Zhao (yuelinz3)

Youtube: link
Code Repository: link
December 16th, 2019

1 Abstract

The ALANDR (Autonomous, Localization and Navigation Delivery Robot) team strongly believes home robots should aid the human experience in day to day tasks. On occasion cats are known to intentionally and without discretion knock items onto the floor. The ALANDR team has designed and created a robotic platform with the same name that is capable of transporting objects from one location safely to another. This capability will aid the user in picking items from the floor when their cat decides to go on a rampage. In addition, ALANDR can map a region in real time creating a 3D representation of its environment allowing the user to visualize the state of the room at any time.

Testing was done to ensure ALANDR performed as expected and error for important aspects of the system were documented and plotted. ALANDR proved to be successful 30 out of 30 tests. Further error analysis and results can be found in section 5.

In conclusion, completion of the project taught the ALANDR team the complexity of such systems and the capabilities of simulation. It became apparent early in the project that by using simulation the team had access to data regarding the simulation that would be hard or impossible to obtain in the real world. The team strongly believes that if given a bit more time the full project as envisioned could be realized. Struggles experienced by the team and plan for moving forward are described in depth within the report.

2 Introduction

Robotic arms are often seen as "steppingstones" and have been applied broadly in industry, as they can perform various tasks automatically or under remote control, like pick and place, assembly, and welding. However, most of them can only perform tasks in stationary work space, which is one of the most apparent limitations of robotic arms. Mobile robots, on the other hand, are capable of moving around their environment. By using a mobile robot as the base, a robotic arm can be moved to different locations to perform tasks. Such a combination can dramatically broaden the work space of a robotic arm and increase the variety of tasks it can perform. Therefore, the objective of the project is to design and control such a robot that combines a wheeled mobile robot and a robotic arm.

As the acronym ALANDR outlines, the designed robot should be able to localize itself in an unknown environment, generate a path and move to the destination while avoiding stationary obstacles, then pick or place objects there. In this project, V-REP is used as the simulator to test the robot, and the Python variant of Robot Operation System (ROS) is used to control the robot.

In order to achieve our goals, the team needs to implement various techniques, including simultaneous localization and mapping (SLAM), path and motion planning, forward kinematics and inverse kinematics. Most of the techniques are learned in this class, except for SLAM. SLAM or in this case Hector SLAM is one of the most popular techniques in modern robotics, as it can use Lidar sensor data to construct and update the map of an unknown environment while simultaneously tracking its own position. By the end of this project, the team was able to generate the map using the hector slam package [1] and a path for the robot using the navigation package [2]. These packages show that it is possible for the robot to localize itself in the map and then move to its destination autonomously. However, coordinate transformation issues between ROS and VREP prevented the team from fully incorporating these packages into the robot's workflow. Therefore, map and path generation are shown but not implemented in robots current control logic. More details will be provided in the following sections.

3 Method of Approach

V-REP allowed for a dynamic design process. Utilizing V-REP the team was capable of choosing the systems in real time by placing components on the model and seeing how they reacted within the overall system. This meant the design process was a prototype based process. The same prototyping type design process was utilized with the programming where many simulations were ran to test ALANDR throughout the project.

The team's goal is to control ALANDR to deliver a block from one position to another. The team separates this task into five steps: Initialize the robot $\rightarrow$ Travel to block position $\rightarrow$ Pick up block $\rightarrow$ Travel to the destination $\rightarrow$ Drop off the block. The complete robot task process is showed in the block diagram in Figure 1. To simplify the problem, the team assumes the following:

- The initial position of the robot is within a predefined region and its pose is known
- The block is within a predefined region and its approximate position is known
- The drop-off destination for block is known

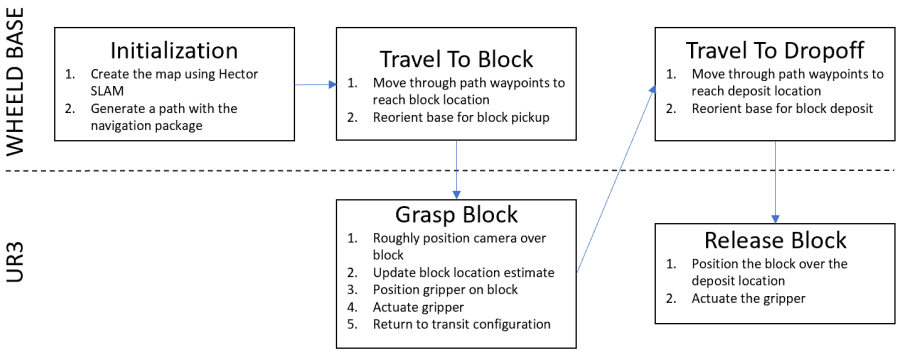


Figure 1: Functional block diagram of robot task process. The dashed line delineates the task responsibility between the UR3 arm and the wheeled robotic base.

3.1 Initialization

The *Initialization* module takes two steps, creating the map and generating the path. An occupied grid map can be created based on Lidar point cloud with black indicating obstacles and white indicating free space. With the occupied grid, the team can conduct path planning and navigation. Then, the robot tracks the trajectory to the goal and finish the following operations.

1. Create the map using Hector SLAM
2. Generate a path with the navigation package

3.1.1 Create the map using Hector SLAM

In this part, the team uses Hector SLAM method to build an occupied grid map of our experiment scene. The Hector SLAM method can be used without odometry. It only needs the data from Lidar and relies on scan matching approach to construct a complete map. The pioneer robot is equipped with a **Hokuyo URG 04LX UG01_Fast Lidar**, which can help robot perceive the environment. The data published to the "/scan" ROS topic is shown in Figure 2.

Figure 2: Scan Information

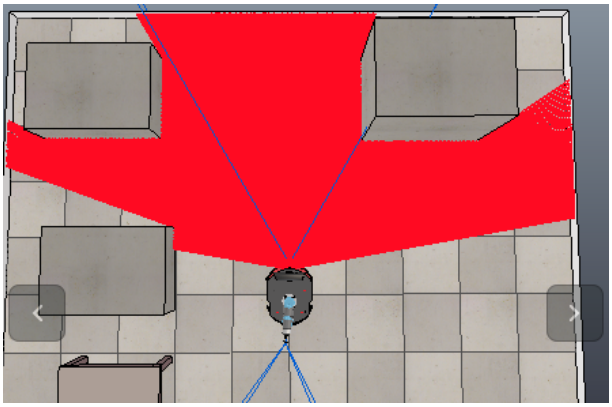


Figure 3: Lidar scan in V-rep scene

3.1.2 Generate a path with the navigation package

After the team constructs the map of the scene, the team can use this map to help robot generate path. In this section, the team takes the ROS navigation package and the amcl method to navigate the robot in the pre-defined map. After the rviz is launched, the initial pose and the goal can be set using a mouse. Then, a path will be generated and the robot will follow the path to the navigation goal.

However, the path generated by the navigation package was in the map coordinate and the team encountered a problem with the transformation between map coordinate and V-rep world coordinate. Hence, in the experiment part, the team used a pre-defined path instead of path generated by navigation package. The problem is analyzed in section 5.

3.2 Travel to Block

The *Travel to Block* module assumes that the robot's position, the robot's orientation, the block's position, and a path to the block is known. Since there were issues with the localization and navigation packages implemented in Section 3.1, the positions are found querying V-REP and the path is mostly predefined. The module proceeds by carrying out the following steps:

1. Move through path waypoints to reach block location
2. Reorient base for block pickup

3.2.1 Move through path waypoints to reach block location

The robot base is controlled by a two-wheel differential drive. V-REP allows for control of the robot by setting the angular velocity of each wheel's motor. To make the robot follow a path, the PD controller from [3] was used.

$$\omega_L = \frac{1}{r}(V_c - K_p\theta_e - K_d\dot{\theta}_e) \tag{1}$$

$$\omega_R = \frac{1}{r}(V_c + K_p\theta_e + K_d\dot{\theta}_e) \tag{2}$$

where ω_L is the angular velocity of the left wheel's motor, ω_R is the angular velocity of the right wheel's motor, r is the radius of each wheel, V_c is the desired speed of the base, K_p is the proportional gain, K_d is the differential gain, θ_e is the path angle error, and $\dot{\theta}_e$ is the time rate of change of the path angle error. The path angle error is defined to be the angle between the robots forward direction and the direction from the robot to the current waypoint as illustrated by Figure 4.

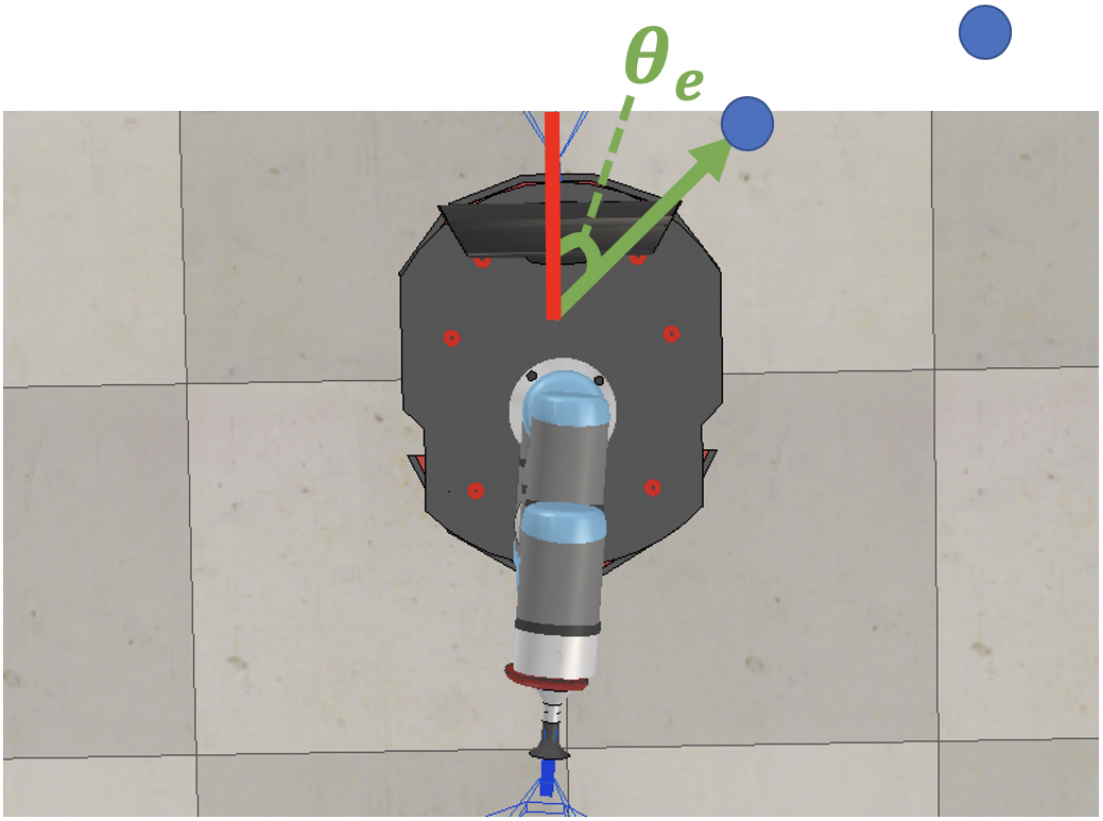


Figure 4: Definition of the path angle error, θ_e. The red line is the forward direction, the blue dots are waypoints, and the green line is the direction from the robot to the current waypoint.

The controller gains used are $K_d = 0.5$ and $K_p = 0.25$. The desired velocity was set according to the following equation:

$$V_c = \begin{cases} 0, & \text{if } |\theta_e| > \frac{\pi}{6} \text{ or } |\dot{\theta}_e| > 0.1 \\ 0.1, & \text{if not first case and } d < 0.2 \\ 0.3, & \text{otherwise} \end{cases} \tag{3}$$

where d is the distance from the robot position to the next waypoint. Once the robot gets within $0.05m$ of the waypoint, it starts tracking next waypoint in the path. The final

waypoint is set to place the robot within the possible griping range as described in Section 3.3.

3.2.2 Reorient base for block pickup

Once the final waypoint is reached, the robot is reoriented to face away from the block, ensuring that the angle between the block and forward direction are not within the restricted zone as described in Section 3.3. When the robot base is in the final orientation, ALANDR stops any further movement and the system transfers to the *Grasp Block* module.

3.3 Grasp Block

The *Grasp Block* module requires that the robot base is situated with an appropriate orientation and distance with respect to the block. Figure 5 illustrates the region that the block can be in for the *Grasp Block* module to correctly operate. Once these conditions are met, the module proceeds by carrying out the following steps:

1. Roughly position camera over block
2. Update block location estimate
3. Position gripper on block
4. Actuate gripper
5. Return to transit configuration

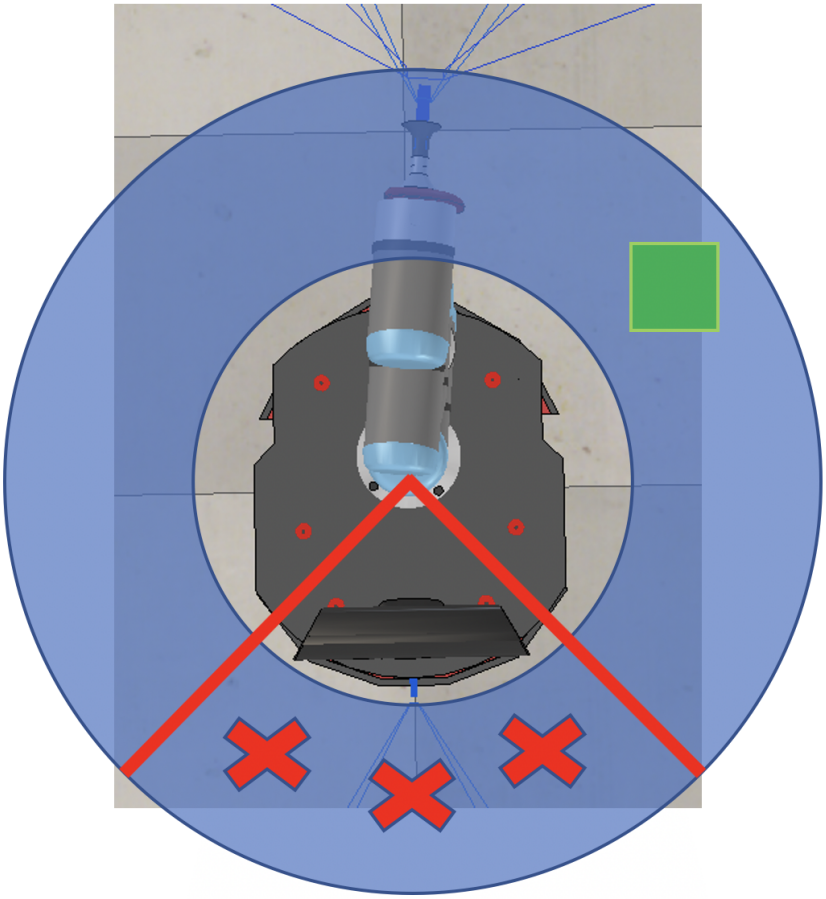


Figure 5: The angle between the front of the robot base and direction vector from the robot base to the block must be between -130 and 130 degrees. The distance of the block from the robot base must be between 0.2 and 0.375 meters.

3.3.1 Roughly position camera over block

It is assumed that there is some uncertainty in the position of the block with respect to the base when this module begins. To overcome this, the camera attached to the end-effector is placed face down over the expected location of the block. The desired end-effector position for this step is given by equation 4. The angle, θ, is defined in Figure 6.

$$P_{camera} = \begin{bmatrix} -\frac{1}{2}cos(\theta) & \frac{1}{2}sin(\theta) & 0.2 \end{bmatrix}^T \tag{4}$$

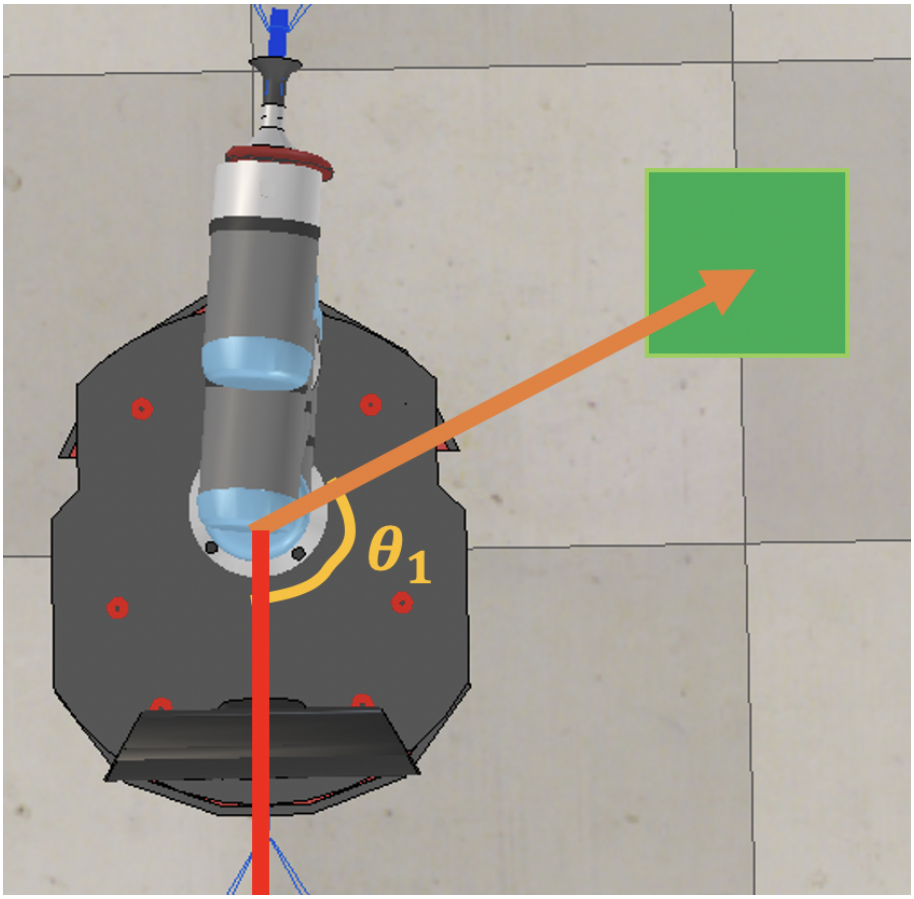


Figure 6: The angle between the front of the robot base and direction vector from the robot base to the block is used for initial camera positioning.

To move the end-effector into the desired position, the team uses inverse kinematics to obtain a series of UR3 joint angles that move the gripper to into position. The series of joint angles are sent to V-REP as set-points for the built-in PID controllers that control each of the UR3s joint angles. Since the inverse kinematics rely on parameters from the forward kinematics, a brief description of the forward kinematics are given first.

Forward kinematics: The zero position and joint coordinate frames are showed in Figure 7. The forward kinematic formulation leads to the screw axes, S, and zero transformation matrix, M, as shown in the following equations.

$$S = \begin{bmatrix} 0 & -1 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 8.5577e^{-5} & 0 & 0 & 0 & 8.5233e^{-5} & 0 \\ -1.246e^{-4} & -1.0887e^{-1} & -3.5252e^{-1} & -5.6577e^{-1} & 1.1222e^{-1} & -6.5112e^{-1} \\ 0 & 5.4415e^{-5} & 1.3034e^{-4} & 8.5248e^{-5} & 0 & 8.5187e^{-5} \end{bmatrix} \quad (5)$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2.2554e^{-1} \\ 0 & 1 & 0 & 6.7162e^{-3} \\ 0 & 0 & 1 & 6.5112e^{-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

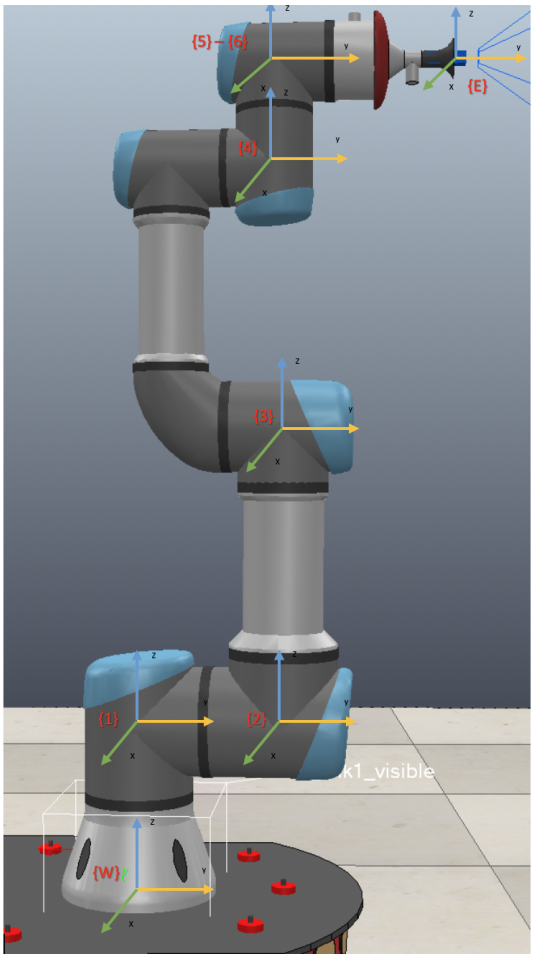


Figure 7: Zero position and coordinate frames of the UR3 used for forward kinematic formulation.

Inverse Kinematics: A slightly modified version of the Newton-Raphson method is used to numerically solve the inverse kinematics, and generate a sequence of joint angles that will smoothly move the end-effector from an initial pose to a desired pose. The modification comes in the form of normalizing the Jacobian and scaling it by the the number of attempts that have been made to create a path within a certain number of iterations. Another modification to the process is the use of the magnitude of the error in the transformation matrix as its success criteria.

3.3.2 Update block location estimate

The rough positioning of the robotic arm over the block places the block in view of the camera attached to the end effector. V-REP's built-in blob detection is used to find the

coordinates of the block with respect to the end-effector camera. The forward kinematics formulation from section 3.3.1 is then used to transform the block position into UR3 base coordinate system. Figure 8 shows the masked view of the end-effector camera while roughly positioned over the block. The green square is the blob representing the block to be picked up.

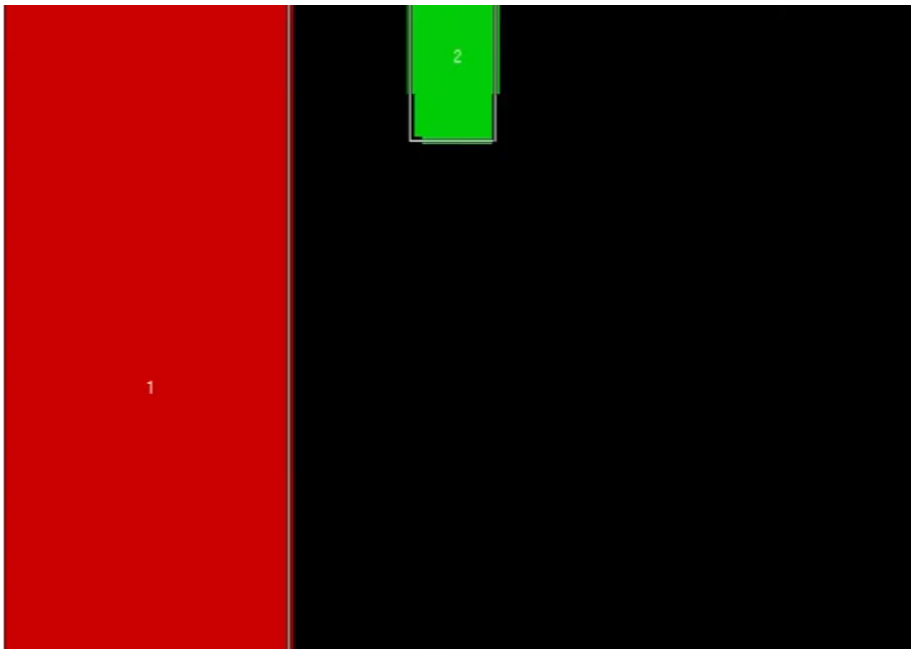


Figure 8: Masked image view of the end-effector camera after being roughly positioned over the block. The green is the blob used to determine the block coordinates.

3.3.3 Position gripper on block

As in Section 3.3.1, inverse kinematics are used to convert the position of the block found in Section 3.3.2 into a series of UR3 joint angles and move the gripper to the block position.

3.3.4 Actuate gripper

Once the end-effector is placed in contact with the block, the suction is enabled. The controller waits until the suction sensor returns a successful suction event before proceeding.

3.3.5 Return to transit configuration

Once the block has been confirmed to be gripped, the robotic arm is moved to its zero position as show in Figure 7. Upon reaching the transit configuration, the *Grasp Block* module is complete and the controller transitions into the next module.

3.4 Travel to Drop-off

The method implemented in this module is same as Section 3.2, but with a different path. The module proceeds according to the following the steps:

1. Move through path waypoints to reach deposit location
2. Reorient base for block deposit

3.4.1 Move through path waypoints to reach deposit location

The path for this module moves the robot through a series of waypoints that place it close enough for the robot arm to be able to deposit the block on the deposit platform.

3.4.2 Reorient base for block deposit

Once the base is close enough to the deposit location, the robot base is reoriented so that its side is facing the deposit location. This final position and orientation was selected to simplify the implementation of the *Release Block* module.

3.5 Release Block

The *Release Block* module requires that the deposit location is within $0.Xm$ and that the robot's right side is facing the deposit location. The process for is carried out in the following steps:

1. Position the block over the deposit location
2. Actuate the gripper

3.5.1 Position the block over the deposit location

The required conditions of this module trivializes the positioning of the block over the deposit location. The UR3 joint angles are simply set as follows:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\frac{\pi}{2} \\ 0 \end{bmatrix} \tag{7}$$

3.5.2 Actuate the gripper

Once the joint angles reach their set positions, the suction is disabled. The block falls onto the deposit location and the robots task is complete.

4 Experimental Setup

4.1 Model

ALANDR uses both pre-installed packaged models in conjunction with V-REP Lidar and blob detection sensors. The team began with the model titled **Pioneer-p3dx**, this model was classified as the "Base" model during exporting. The Pioneer model was chosen for its low profile and sufficient surface area needed for the addition of sensors and a robotic arm. The robotic arm chosen was the pre-installed **UR3** model. Having used the **UR3** robotic arm in labs the team had experience with scenarios that were similar to the overall goal of the project. The name of the two chosen sensors are **fastHokuyo** and **blobTo3dPosistion** and correspond to the Lidar and blob detection sensors respectively. The placement of the **fastHokuyo** sensor was chosen to successfully localize the robot , detect obstacles, and build a 3d map in real time. The **blobTo3dPosition** sensor and **BaxterVacuumCup** placement was chosen so that ALANDR would be capable of locating a block within a given uncertainty region and transporting said block to a new location. **Pioneer_p3dx_caster_free** wheels were added to ensure ALANDR was stable and could perform all the required tasks while maintaining the **fastHokuyo** on a parallel plane with the floor to aid better map generation. Figure 9 illustrates the placement of all the models used within ALANDR.

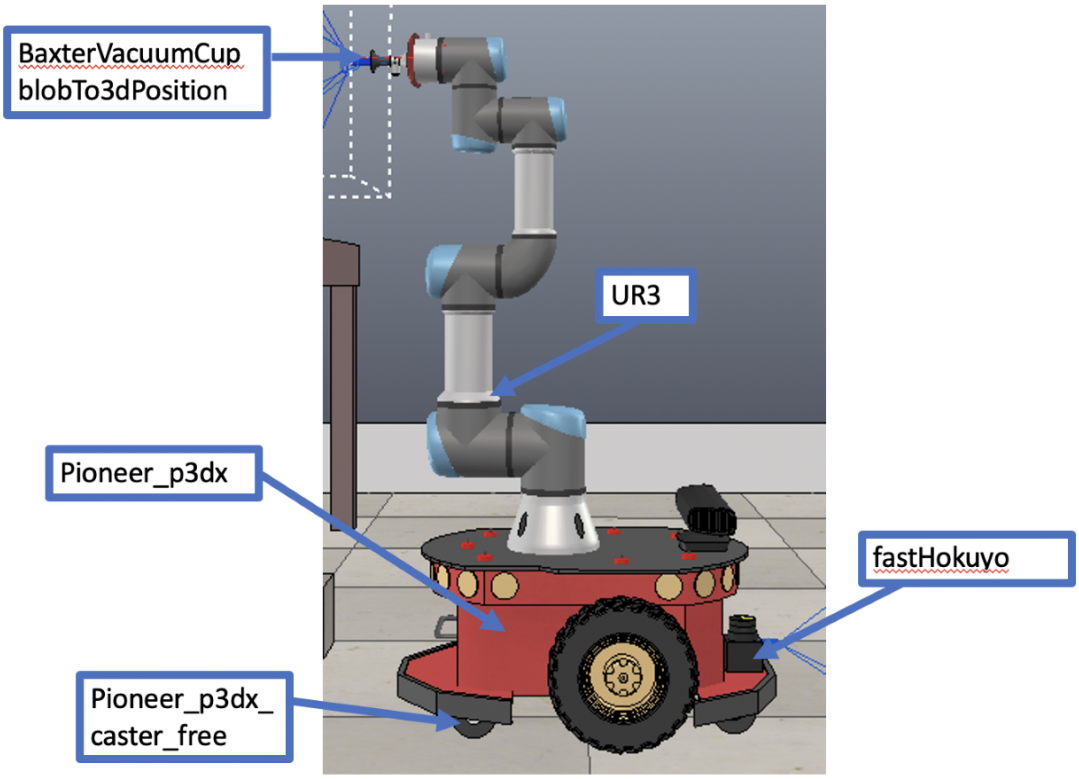


Figure 9: V-REP models used in ALANDR

4.2 Scene

A scene was created to test the capabilities of ALANDR. The team's aim when constructing the scene was to keep it grounded in reality by using models and shapes that may be found in a normal indoor environment. Scene #1 as seen in figure 10 was used to test the re-positioning of the block and Hector SLAM map generation implementation. The scene was constructed using existing models and shapes within V-REP by manipulating size and density to achieve the desired properties.

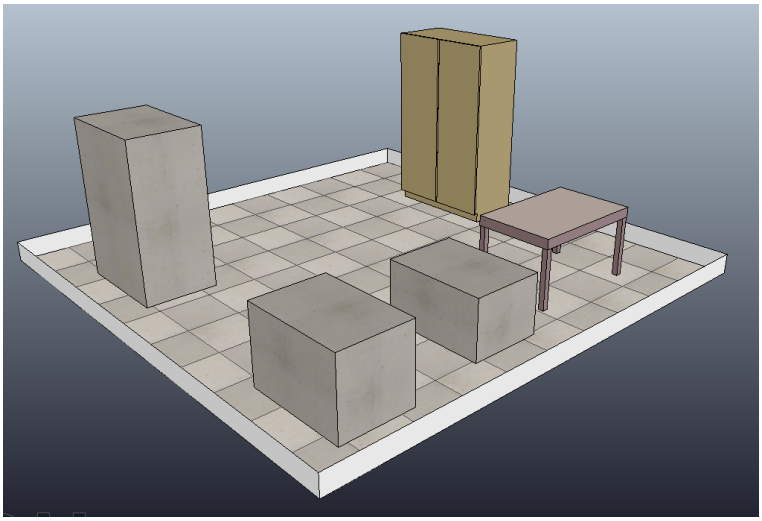


Figure 10: Scene #1 Used for Experimentation

4.3 Task/Objective

The overall goal of ALANDR as described earlier is that of an autonomous vehicle that is capable of locating itself and an item then moving said item to a new predetermined location. The experiment must therefore be defined to ensure this overall goal is tested. Firstly, a map was created using Hector SLAM by instructing ALANDR to travel around the map. This was used to ensure the teams Hector SLAM implementation was working properly. ALANDR's movement was either done manually using keyboard inputs or by implementing a simple script for the initial map creation. The team then tested the second large portion of ALANDR functionality, picking up and transporting a block to a new location. This test was performed by randomly placing ALANDR and a block in the locations shown in figure 16. For the purpose of testing and data collection the path and end location were defined by the team. Success of the task was defined to be when the item was transported successfully to the deposit location.

4.4 Simulation Interface

In order for the team to implement desired existing packages, communication between V-REP and python had to be established. Robot Operating System (ROS) was used to communicate between the two, the following figure illustrates the implementation.

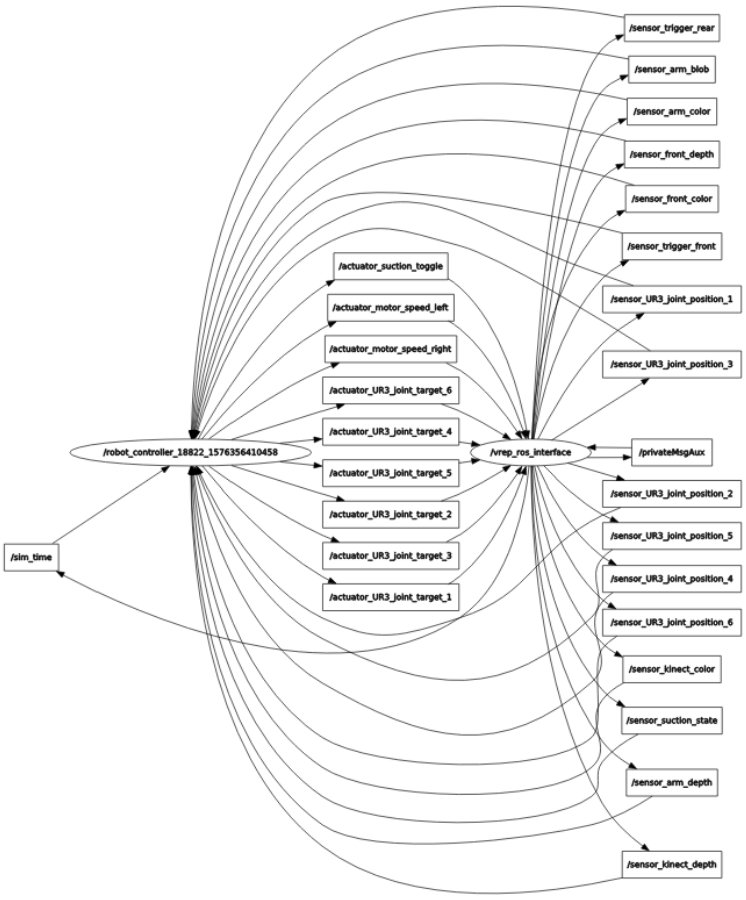


Figure 11: ROS Nodes and Topics (not all inclusive)

5 Data and Results

5.0.1 Hector SLAM Results

The team setup a Pioneer robot with arm and Lidar in the scene of Figure 10. Then the Hector SLAM was launched in the terminal and the Lidar scan result was shown in Figure 12. Robot base was controlled by keyboard to move in the experiment scene. During the movement, Lidar continued scanning, and all Lidar scan frame will be processed by scan matching algorithm to generate a complete occupancy grid map.

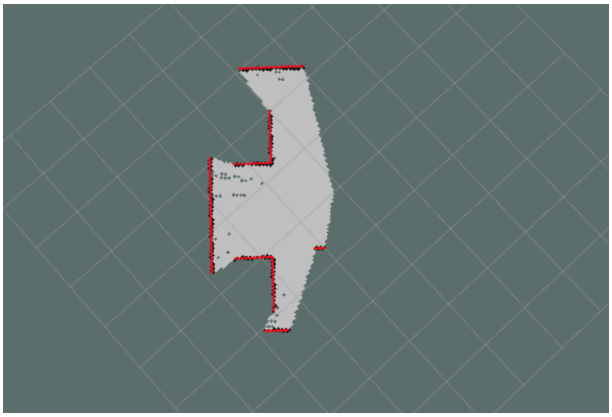


Figure 12: 2D Lidar scan point cloud

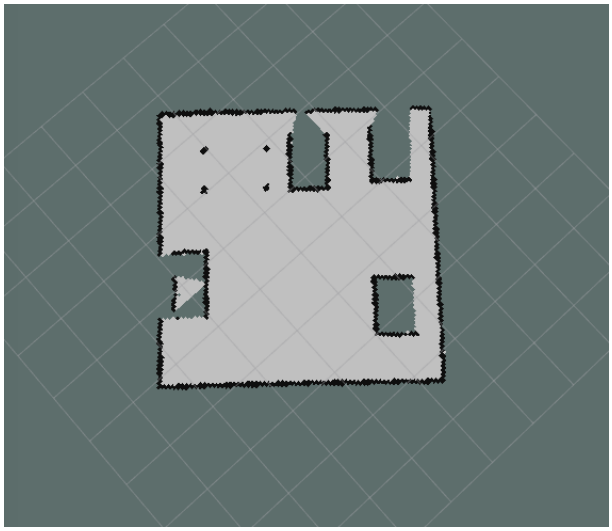


Figure 13: Occupied grid map generated by Hector SLAM

Then, map-server was used to save the map and load the navigation package with this map. In the Rviz software, a mouse was used to set the initial position of the robot as well as its initial pose. Also, the goal was set using the same approach. A path was created after setting two points as shown in Figure 14 indicated by the green line.

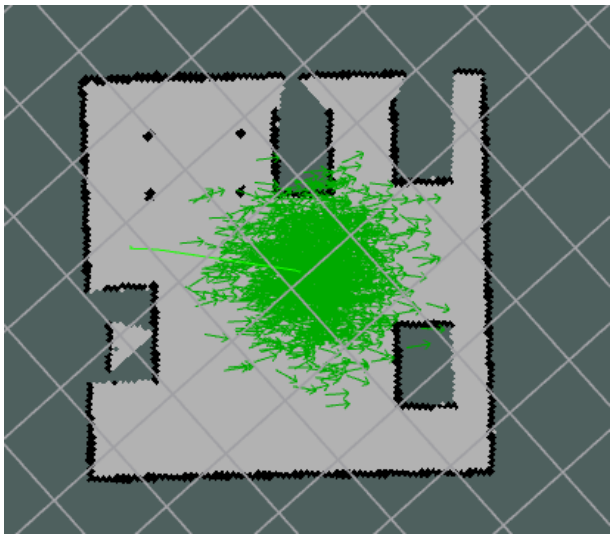


Figure 14: Path generated by ros navigation package

However, in the path following section for the robot, it was found that the robot didn't exactly follow the expected path to the end point. After checking all the setup and map building process, it was found that there existed an offset between map coordinate and V-rep world coordinate. As Figure 15 shows, the Lidar scan point cloud cannot match the pre-defined map after we selected the initial position. Unfortunately, the team still needs more time to fix this problem. Hence, to guarantee the completeness of the project, pre-defined

paths were used in the experiment.

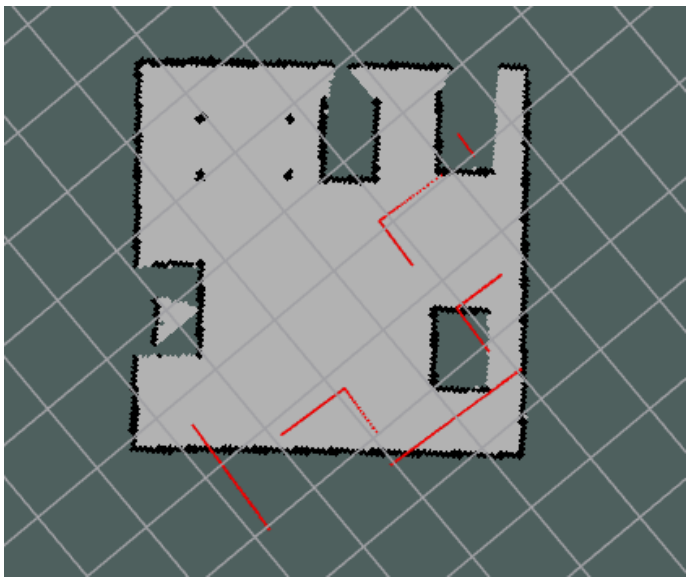


Figure 15: Lidar scan and map offset

5.0.2 Block Movement Results

The following scenario was performed within V-REP to obtain data and quantify the success of ALANDR. A predetermined area where the block may be located was defined and a script within V-REP was written to place the block randomly within the region. The pose and orientation of ALANDR was also altered. Figure 16 illustrates the valid regions for the block (green) and ALANDR (blue) positions. The final desired location of the block was constant throughout all the runs and located on top of the table.

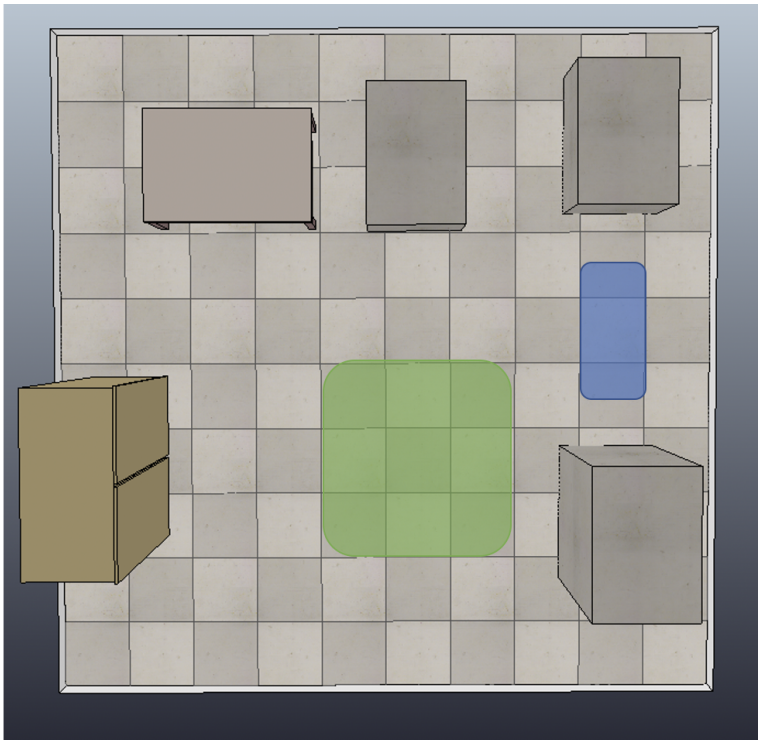
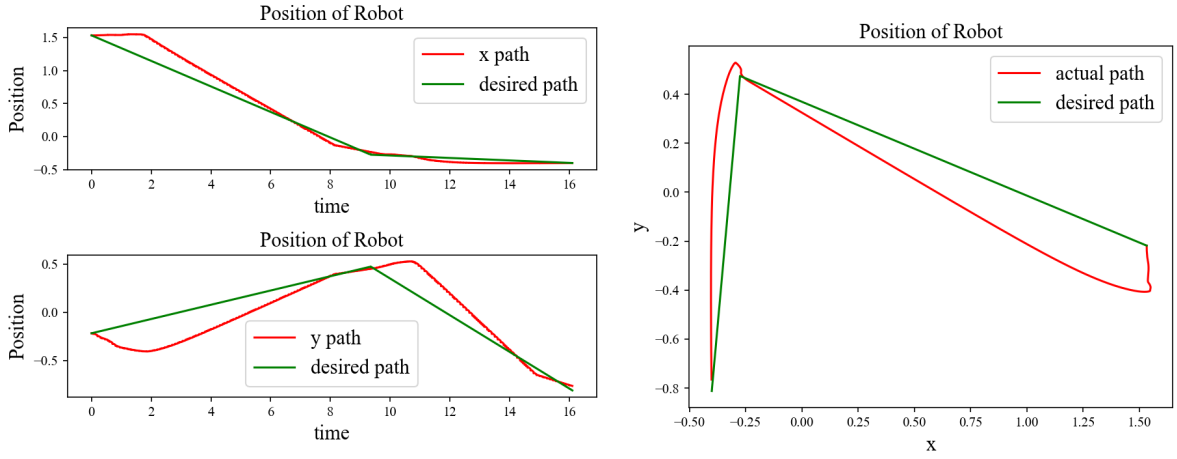
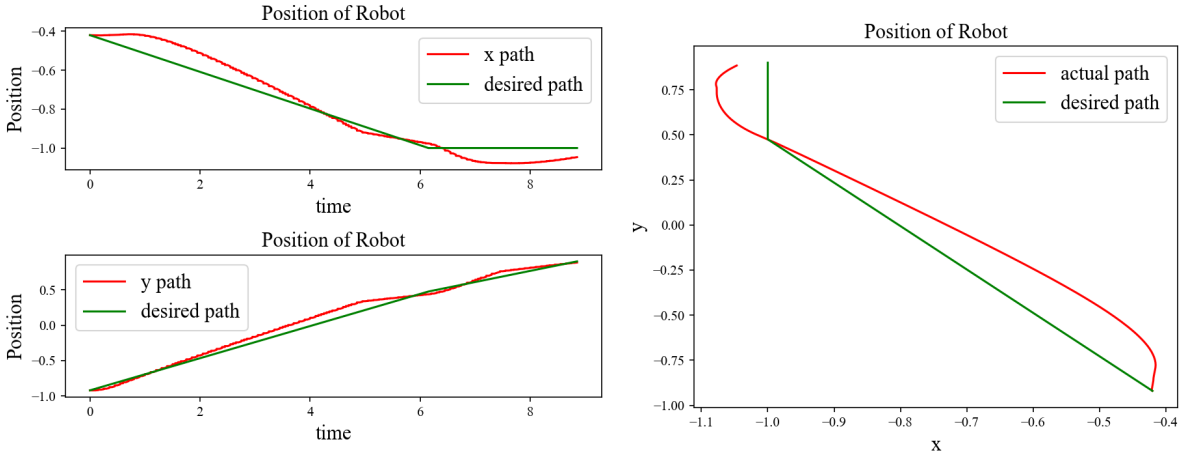


Figure 16: Valid regions used for testing, ALANDR (blue) block (green)



((a)) x and y path **((b)) 2D path**

Figure 17: Graph of the actual and desired path moving to the block position



((a)) x and y path

((b)) 2D path

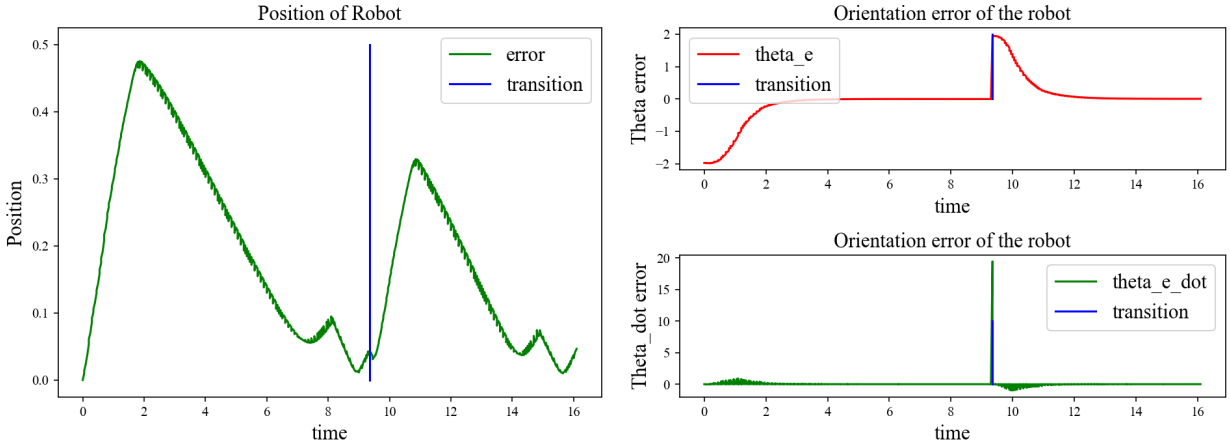
Figure 18: Graph of the actual and desired path moving to the drop off position

The path in both the x and y directions was then plotted with respect to the desired path for two tests as seen in figure 17 and 18. As seen by each set of graphs ALANDR does a relatively good job on staying on path, however, some overshoot does occur in both tests. If obstacles where an issue then the controller may need to be modified so that overshoot is reduced to ensure ALANDR doesn't collide with obstacles. Both tests resulted in ALANDR delivering the block to the desired location and deemed as successful.

5.0.3 Error Analysis

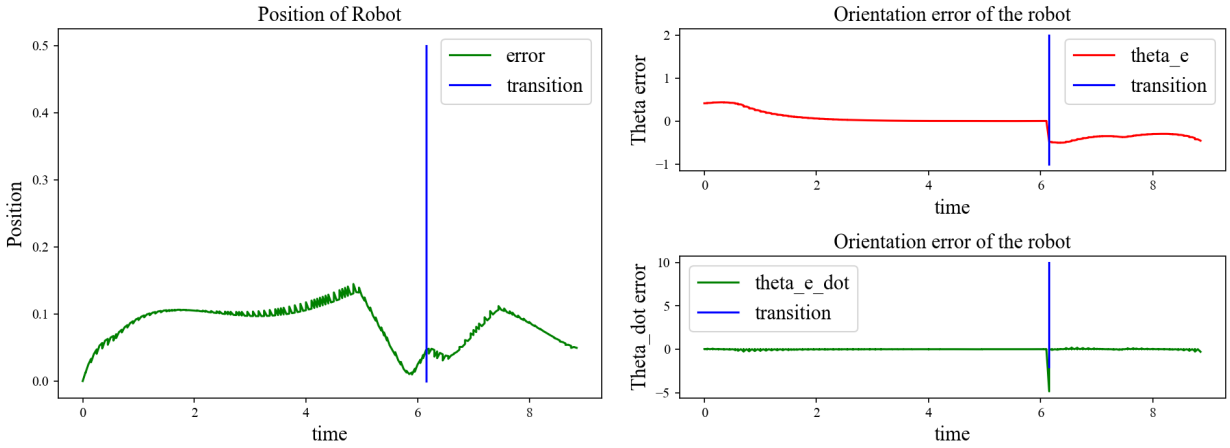
In hopes of better understanding the overshoot seen in the previous section an error analysis was performed. The error analysis consisted of three plots for each test, figure 19 and 20. As seen by figure 19 the position error increases sharply twice and then slowly decreases with time. The decrease of error to almost zero when approaching the transition location is attributed to the fact that ALANDR decreases speed to minimize overshoot when approaching. Similarly, the large increase in error aligns with the transition phase and the initializing phase for test 1. When analysing the path taken by ALANDR it is clear the current controller prefers slow gradual turns rather than turning in place. A simialr effect can be seen in test 2 but with less effect due to the more linear path taken to achieve its goal. If reduction of this error was desired a new controller that penalized large errors should be used to ensure ALANDR spins in place when possible.

The orientation error for both tests was centered around zero and spiked during transition, this is to be expected and doesn't require further optimization.



((a)) Position error **((b)) Orientation error**

Figure 19: Graph of the corresponding position and orientation error



((a)) Position error **((b)) Orientation error**

Figure 20: Graph of the corresponding position and orientation error

5.0.4 Success Rate/Failure Case

ALANDR proved to be successful in all 30 tests performed. Issues might arise however after full implementation of Hector SLAM is performed. A possible location for failure that may need further development is the workable area for the UR3 robotic arm. If the block happens to be located at a location where the UR3 arm can't travel to, mission failure may occur. Current ALANDR modules do not account for such a scenario. Another failure case that might occur is when ALANDR path error is too large and it collides with a boundary or obstacle. This scenario did not occur during testing, however, if the scene was more restrictive this could prove to be challenging for ALANDR. All these failure cases should be considered further moving forward.

Table 1: Results for full task attempts. A successful trial requires the robot to move to the block, pick it up, move to the deposit location, and drop it off.

Number of Trials	Number of Successes	Percent Success
30	30	100

6 Conclusion

The team was successful in creating ALANDR, a system that is capable of picking up a block and re-positioning said block on a table with 100 percent success rate. ALANDR also has the ability to map a room in real time using Hector SLAM while being controlled by a user. Errors in the overall path were analyzed and the team believes the error can be reduced substantially moving forward. Issues arose when implementing all the envisioned functionality, however, every distinct system operates independently and the team is hopeful that full implementation would have been possible with a bit more time.

By completing the project the team learned how to utilize publishers and subscribers to allow communication between python scripts and V-REP using ROS. The team members did not have V-REP experience prior to the completion of the project. V-REP while powerful and a great tool has some learning curves that the team did not anticipate. In addition, the team decided to use pre-existing packages for the implementation of SLAM that proved to be challenging when linking with V-REP. The team also learned the importance of carefully defining work areas for the UR3 robotic arm. Due to it being a simulation the UR3 robotic arm would on occasion perform maneuvers that are not physically possible, therefore, to keep with the idea of this being possible in the real world the team had to consider further physical constraints for ALANDR.

An argument can be made that the team was overly ambitious at the beginning of the project, however, the team strongly believes that if given a bit more time the two main portions of the completed project can be linked creating the overall initial vision. Currently angular and linear velocity for each motor can be accessed that would perform the required movement within V-REP, however, the team believes an offset is being applied between the Hector SLAM map coordinates and the V-REP global coordinates. Moving forward the team would focus on fixing the current issue and believes that if this issue is resolved the vehicle should perform as initially designed.

References

[1] Ros hector slam. http://wiki.ros.org/hector_slam. Access Jul. 18, 2019.

[2] Ros navigation. http://wiki.ros.org/navigation. Access Jul. 18, 2019.

[3] Kui-Hong Park, Yong-Jae Kim, and Jong-Hwan Kim. Modular q-learning based multi-agent cooperation for robot soccer. *Robotics and Autonomous Systems*, 35(2):109–122, 2001.