Milestone 3
TECHIN 517 SP 2024

Orchestration

Haochen Zeng, Sam Wong, Yeary Yuan,
Kaiwen Men, Jiawen Chen, Ravinder Gulla,
Ankit Shaw

Problem

How might we use a robot system for healthcare facility patients to easily request an item and have it delivered to the room?

Problem

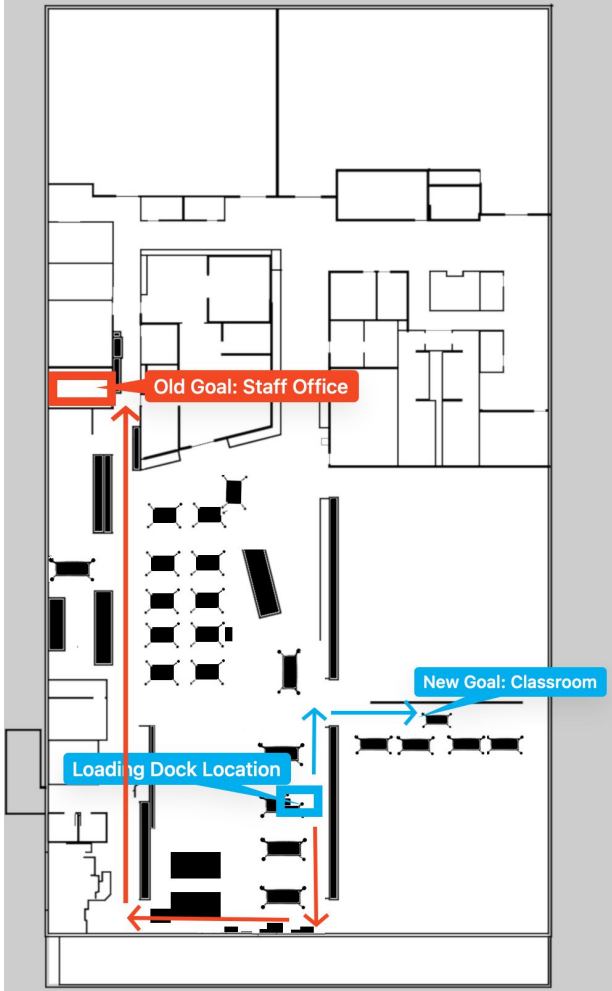
Efficiently delivering essential items can greatly enhance patient satisfaction and well-being.

By **automating this process with robots**, we can ensure **timely and accurate delivery**, **reducing the workload on healthcare staff** and allowing them to focus on more critical tasks.

This will help **improve operational efficiency** and ensure that patients receive the care and attention they need promptly, thereby enhancing overall healthcare outcomes.

Simulated Environment @ GIX

- Navigation
  - From Storage Room (Robot Desk)
    - Desks to place items
    - No need to move arm
  - To Patient's Room (Classroom Lectern)
    - Closer for testing purpose
    - No need for faculty to leave
  - To Staff Office (Old Destination)
    - The most populated office on second floor
    - Request the most items
    - Takes too long to reach
    - Huge empty space on path, Create got lost.



Simulated Environment @ GIX

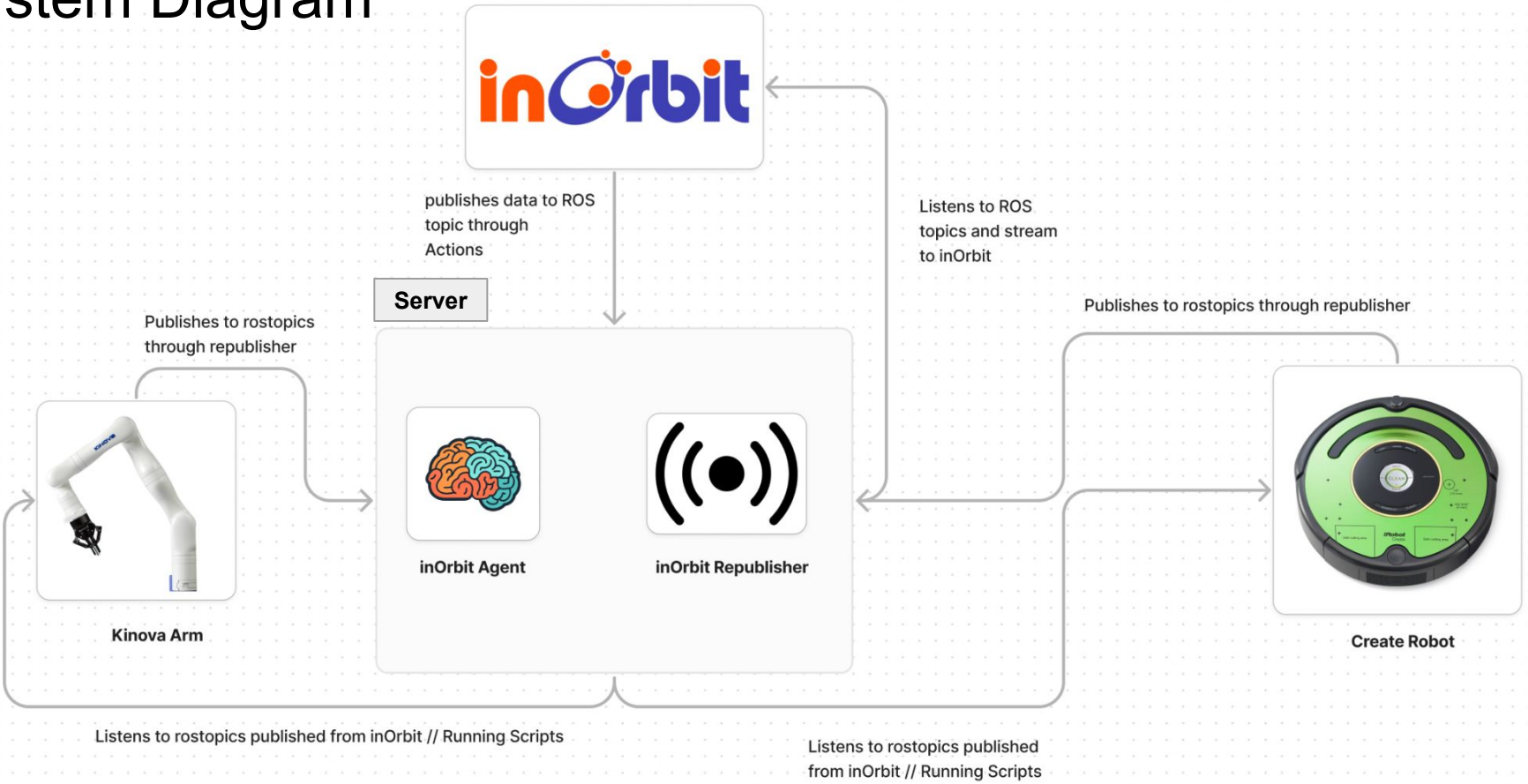
- Manipulation
  - Requested Items for grab
    - Water bottle: soft or firm, various weights
    - Blankets: soft, thin, heavy
    - Sock ball: soft, various sizes

	Requested Item
0	Water
1	Medication
2	Blankets and Pillows
3	Food and Snacks
4	Personal Hygiene Items
5	Socks and Slippers
6	Assistance with Mobility
7	Television or Entertainment Devices
8	Telephone or Communication Devices
9	Comfort Measures

Orchestration - InOrbit

- Agent Installation
  - **Computer** controlling Kinova + Camera
  - **Raspberry Pi** controlling **Create 2** + LiDAR
- Setting up missions and actions
  - Kinova: receive requested item, pick up item, drop off item
  - Create: go to loading dock next to Kinova, wait until item is loaded, go to classroom for delivery

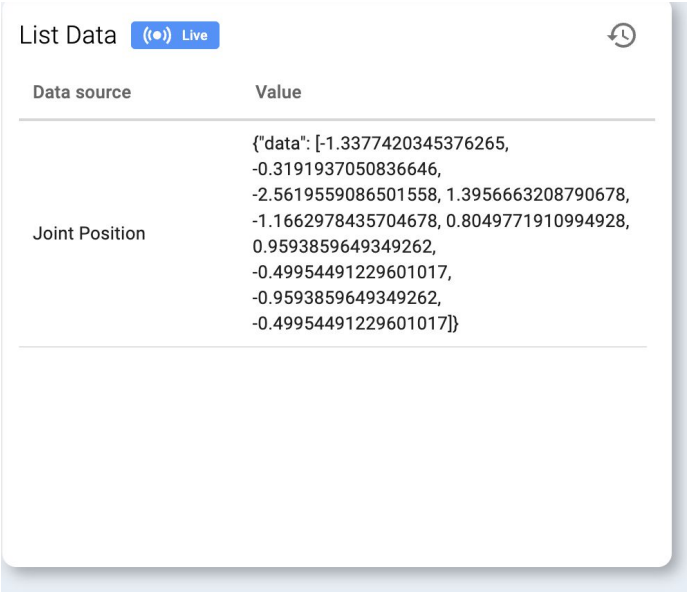
System Diagram



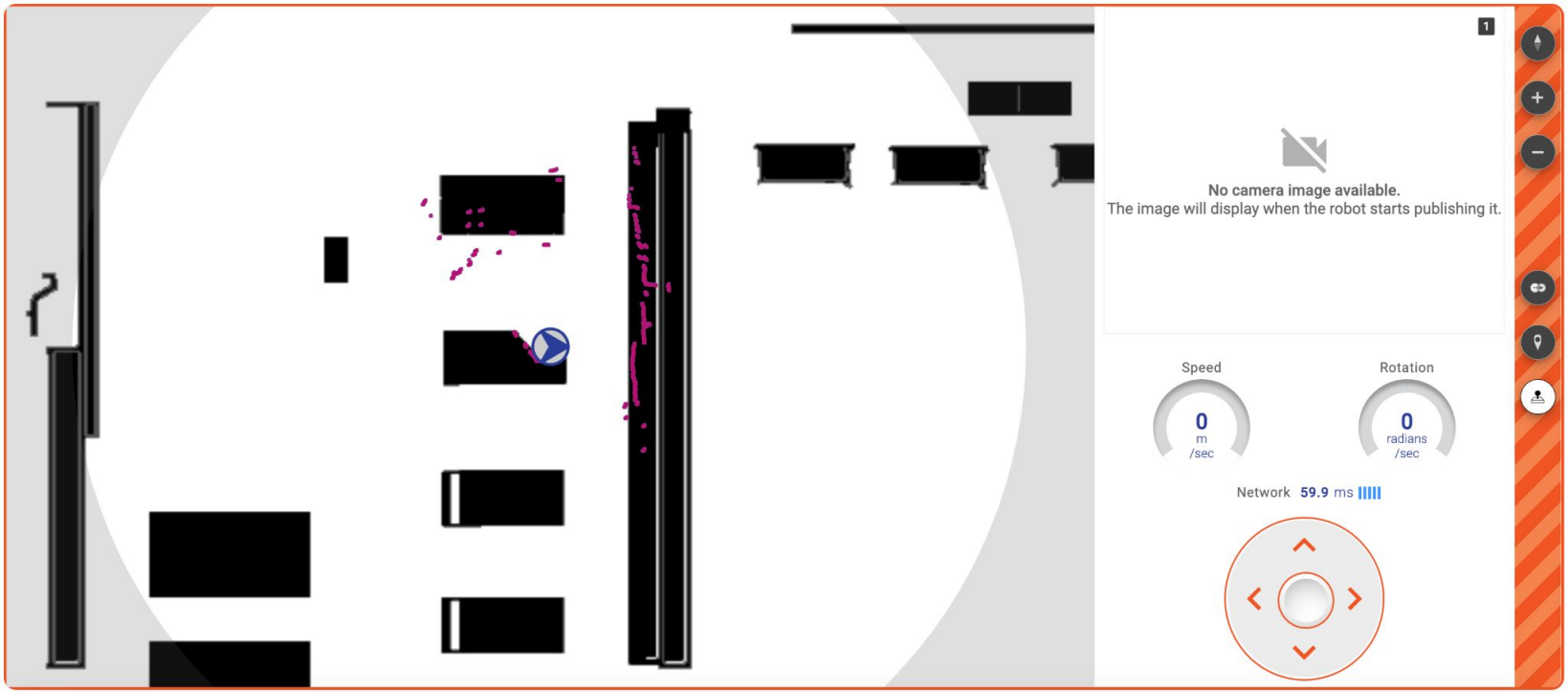
Inorbit Republisher

Subscribing to “/my_gen3_lite/joint_states”

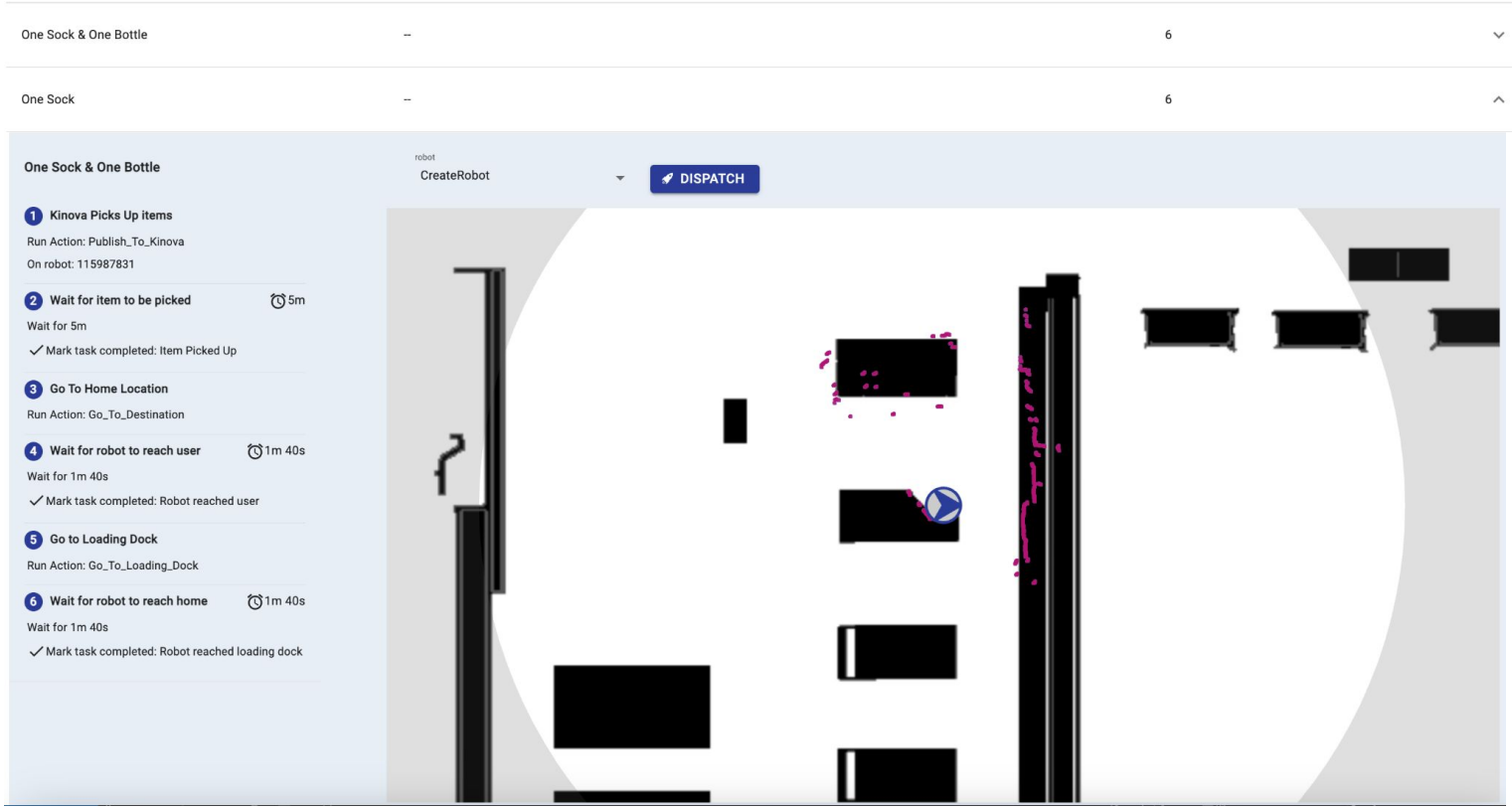
```yaml
republishers:
  - topic: "/my_gen3_lite/joint_states"
    msg_type: "sensor_msgs/JointState"
    mappings:
      - field: "position"  # Field from the ROS message
        mapping_type: "array_of_fields"  # Mapping type for an array of fields
        out:
          topic: "/inorbit/custom_data/0"
          key: "joint_positions"  # Key for the republished data
```



User Interface - Teleoperation

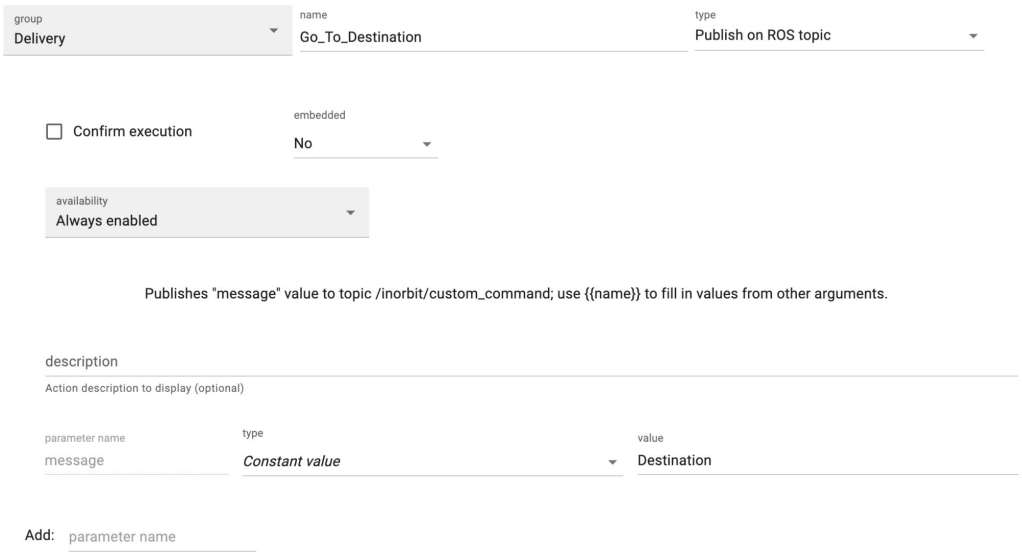


User Interface - Mission



Action Definition

Publish on ROS Topic Action

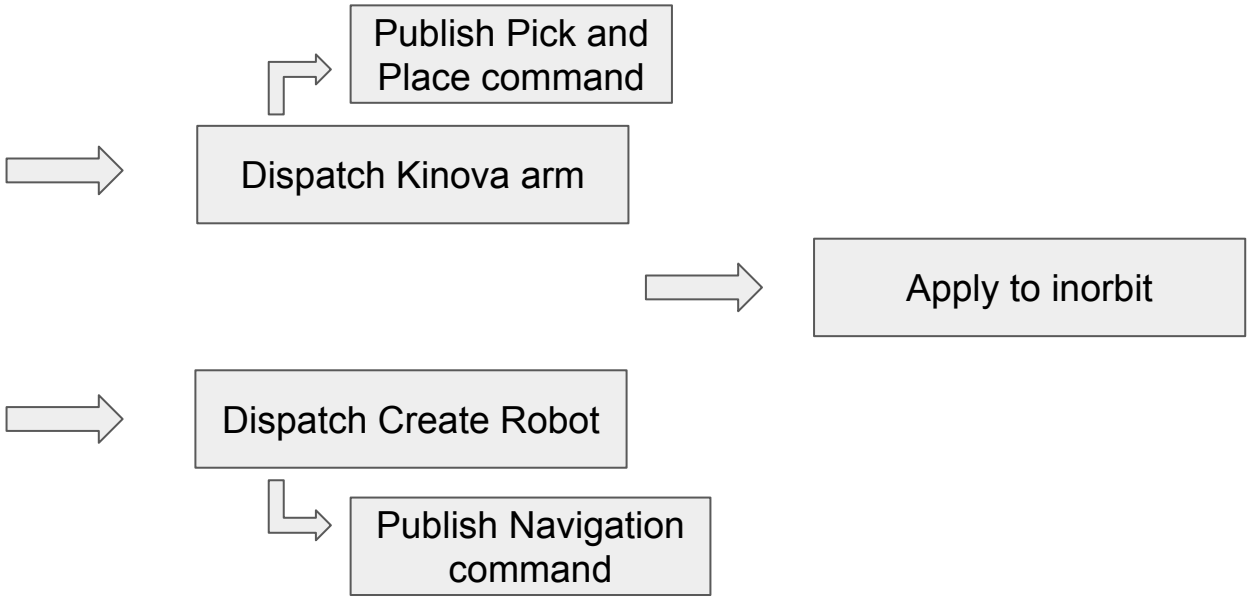



```python
def callback(data):
    print(data.data)
    if data.data == 'Load Dock':
        result = send_goal(16.07, -32.44, 0.0, 1.0)
    elif data.data == 'Destination':
        result = send_goal(25.57, -25.95, 0.0, 1.0)
    else:
        return
    if result:
        rospy.loginfo("Navigation success!")
        pub = rospy.Publisher('arrived', String, queue_size=10)
        pub.publish('Finished!')


if __name__ == '__main__':
    try:
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber('/inorbit/custom_command', String, callback)
        rospy.spin()
```

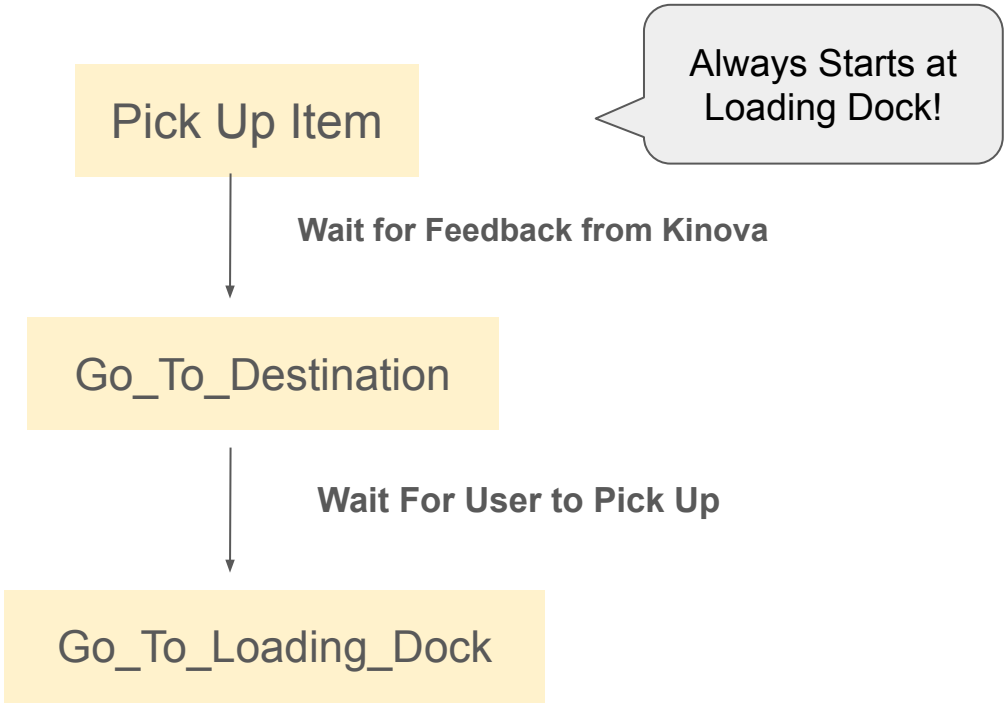
Mission Definition

```yaml
apiVersion: v0.1
kind: MissionDefinition
metadata:
  id: fetch_sock_bottle
  scope: account/2LxBXQ2qQyL6o2eHH
spec:
  label: One Sock & One Bottle
  steps:
  - label: Kinova Picks Up items
    runAction:
      actionId: PublishToKinova
      target:
        robotId: "115987831"
  - completeTask: Item Picked Up
    label: Wait for item to be picked
    timeoutSecs: 300
  - label: Go To Home Location
    runAction:
      actionId: PublishToTopic-0KWqOk
  - completeTask: Robot reached user
    label: Wait for robot to reach user
    timeoutSecs: 100
  - label: Go to Loading Dock
    runAction:
      actionId: PublishToTopic-kjeIdn
  - completeTask: Robot reached loading dock
    label: Wait for robot to reach home
    timeoutSecs: 100
```



Mission Definition

Ex. Deliver 1 sock and 1 bottle to user



One Sock & One Bottle

1 Kinova Picks Up items

Run Action: Publish_To_Kinova

On robot: 115987831

2 Wait for item to be picked 5m

Wait for 5m

✓ Mark task completed: Item Picked Up

3 Go To Home Location

Run Action: Go_To_Destination

4 Wait for robot to reach user 1m 40s

Wait for 1m 40s

✓ Mark task completed: Robot reached user

5 Go to Loading Dock

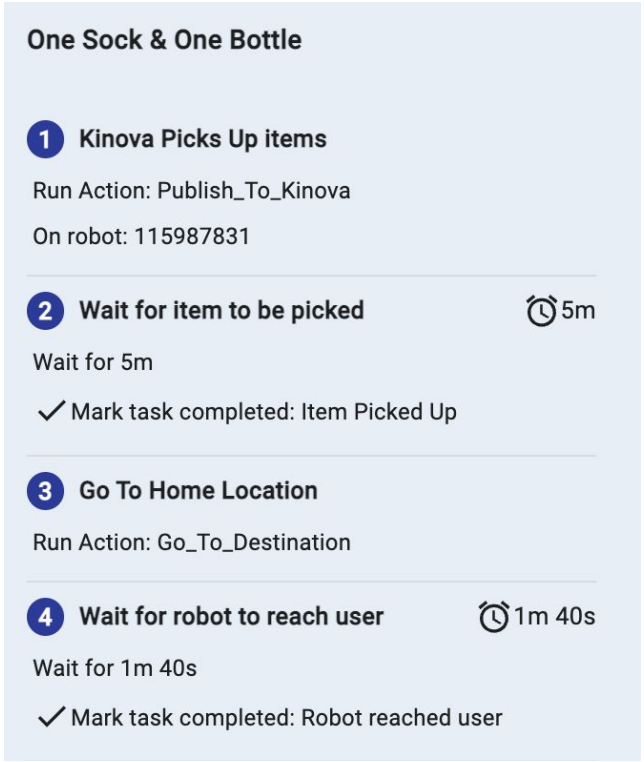
Run Action: Go_To_Loading_Dock

6 Wait for robot to reach home 1m 40s

Wait for 1m 40s

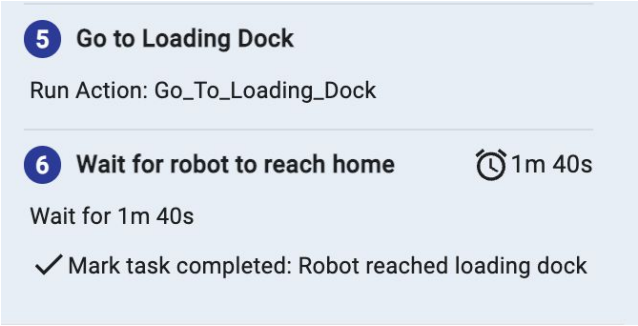
✓ Mark task completed: Robot reached loading dock

Actions



- Kinova Pick Up Items
  - Action: Runs Publish_To_Kinova action to signal the Kinova robot to pick up items. This publishes a message to the ROS Topic with the items that the arm needs to pick.
  - On Robot: Specifically executed on the Kinova robotID: 115987831.
  - Wait 5 minutes for Kinova to put items into Create basket.
- Create Goes to Destination Location
  - Action: Go_To_Destination action executed on Create. This will send the robot to the destination location (lectern).
  - Wait 100 seconds for Create to arrive at Location.

Actions

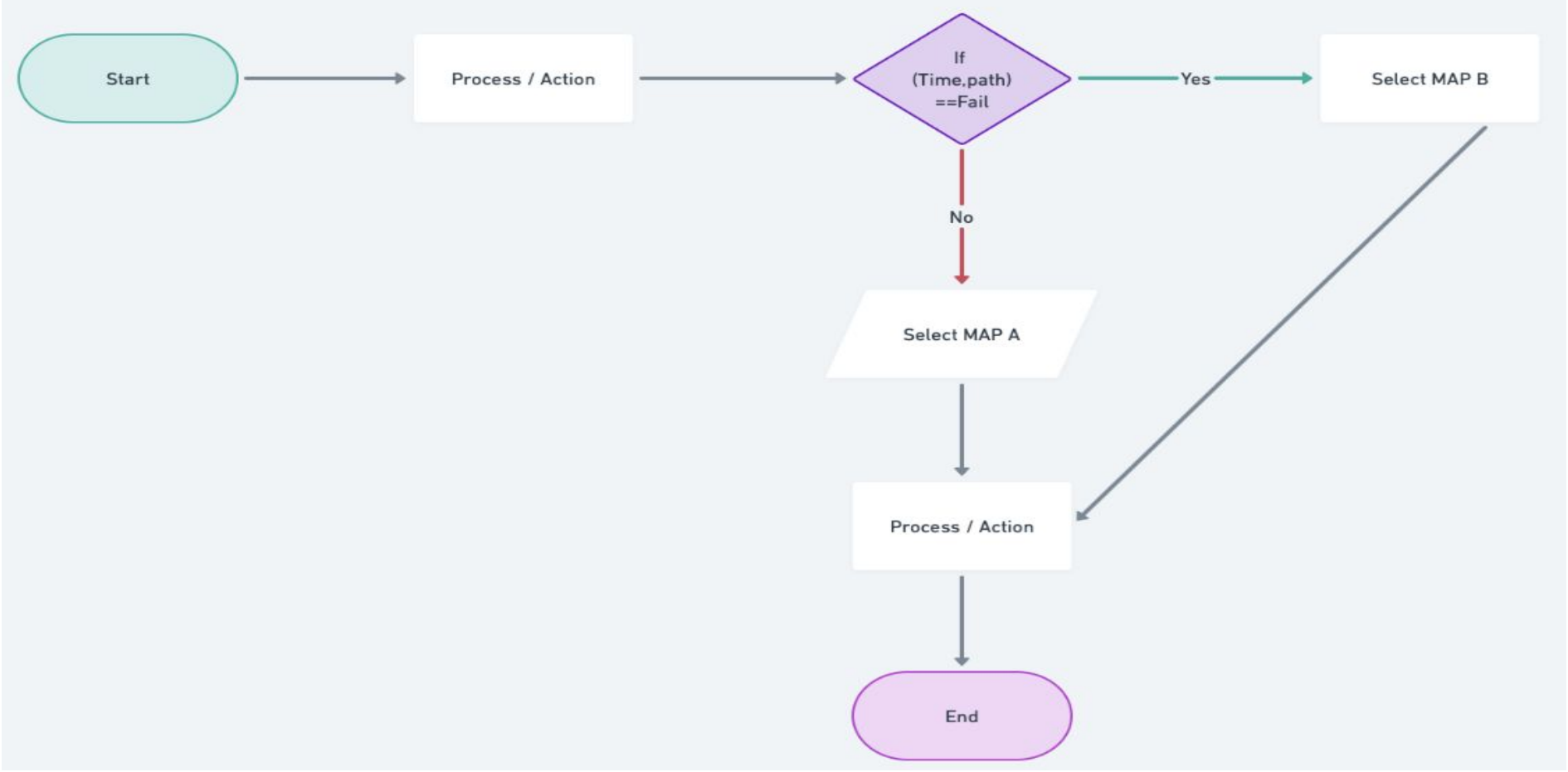


- Create Goes to Loading Dock
  - Action: Go_To_Loading_Dock action executed on Create. This will send the robot back to the loading dock by the Kinova.
  - Wait 100 seconds for Create to arrive at Loading Dock.

Next Steps

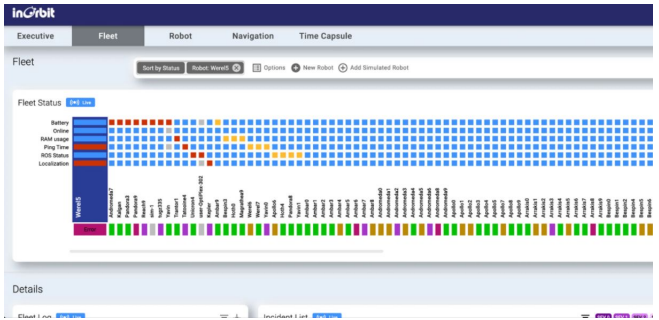
- Immediate: User Experience Improvements
  - Collect feedback from initial users, based on the time taken, path followed and items delivered.
  - Automation: ensure the system can dynamically adapt to new user inputs and operational environments, specially for daily/weekly scheduled task.
- Future: Technical Enhancements
  - Use reinforcement learning algorithms to optimize task execution paths and performances, based on historical performance data.

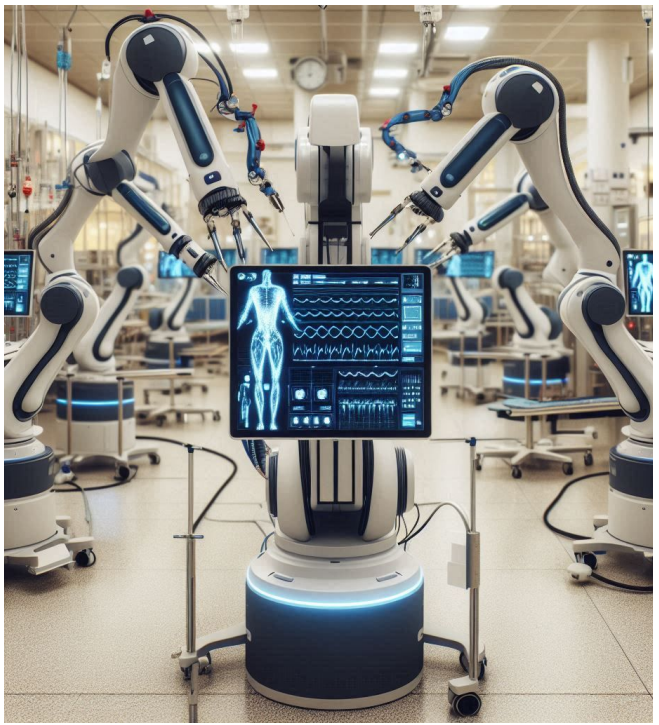
Next Steps


Start
Process / Action
If
(Time,path)
==Fail
Yes
Select MAP B
No
Select MAP A
Process / Action
End

Next Steps

- Scalability & Fleet Coordination
  - Implement behavior trees to manage robot tasks dynamically.
  - Develop criteria for selecting the best-suited robot for each mission based on availability, battery level, and proximity.





Manipulation Team Milestone 3

Sam Cole
Lakshita Singh
Daithy Ren
Kendra Yang
Ziqi Gao
Aayush Kumar
Louisa Shi

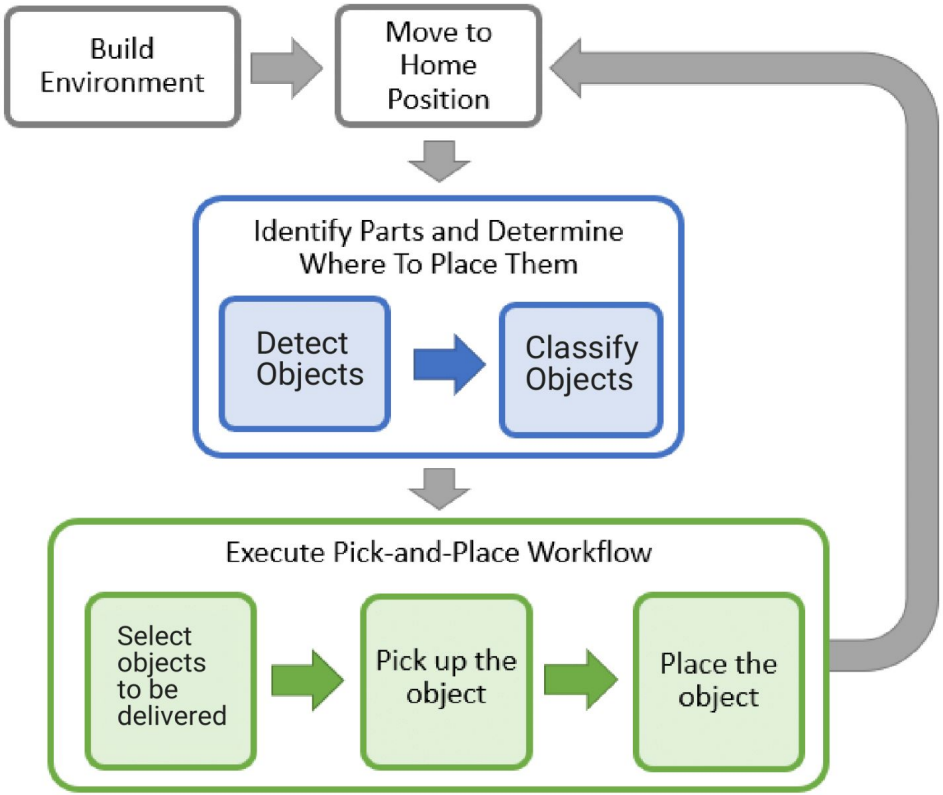
Link

Link

Stateflow Chart

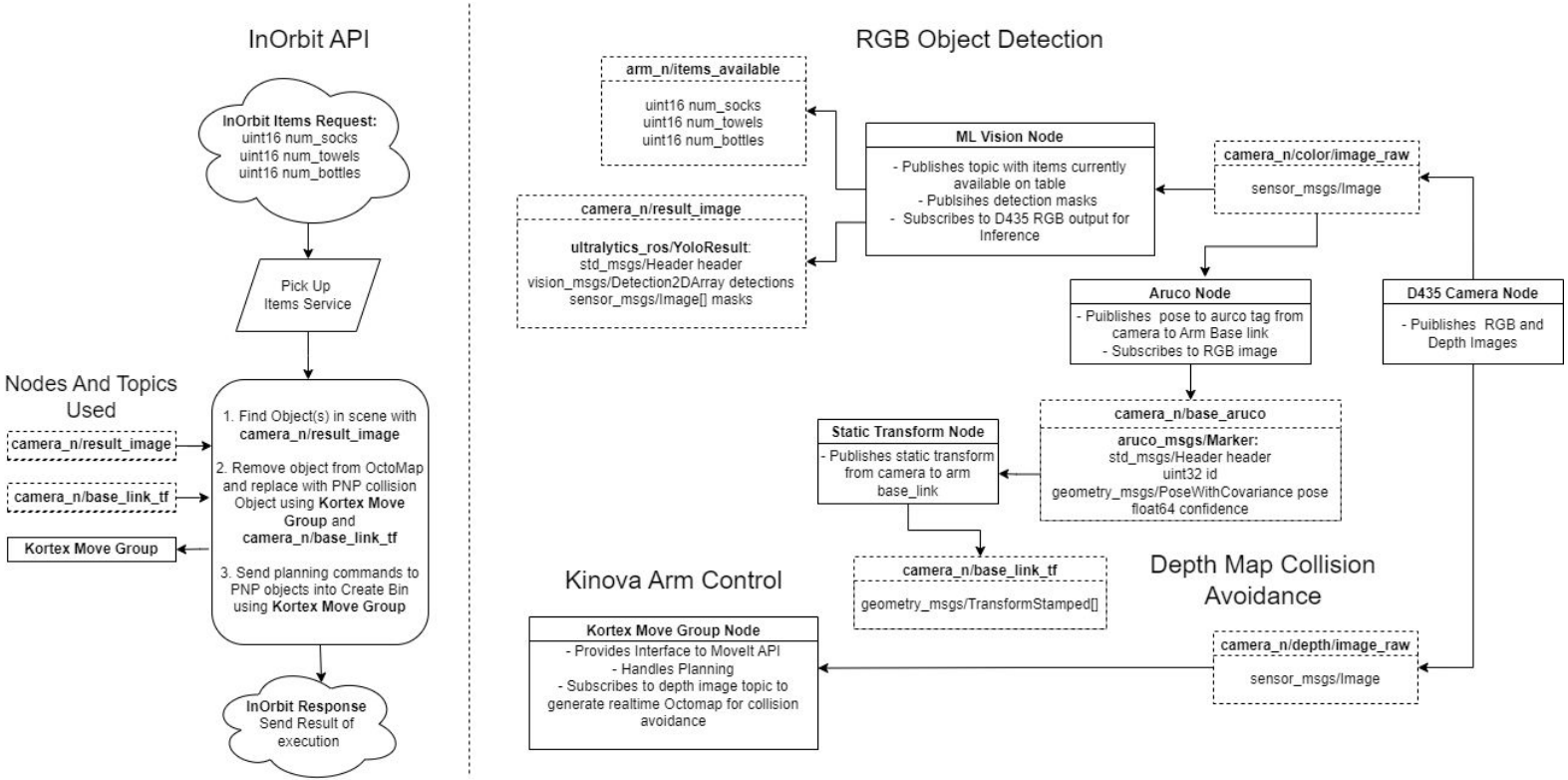
The chart dictates how the manipulator interacts with the objects. It consists of basic initialization steps, followed by two main sections[1]:

- Identify Objects and Determine Where they are and Where to Place Them
- Execute Pick-and-Place Workflow



[1]MathWorks. "Pick and Place Workflow in Gazebo Using ROS." MathWorks, n.d., https://www.mathworks.com/help/robotics/ug/pick-and-place-workflow-in-gazebo-using-ros.html.

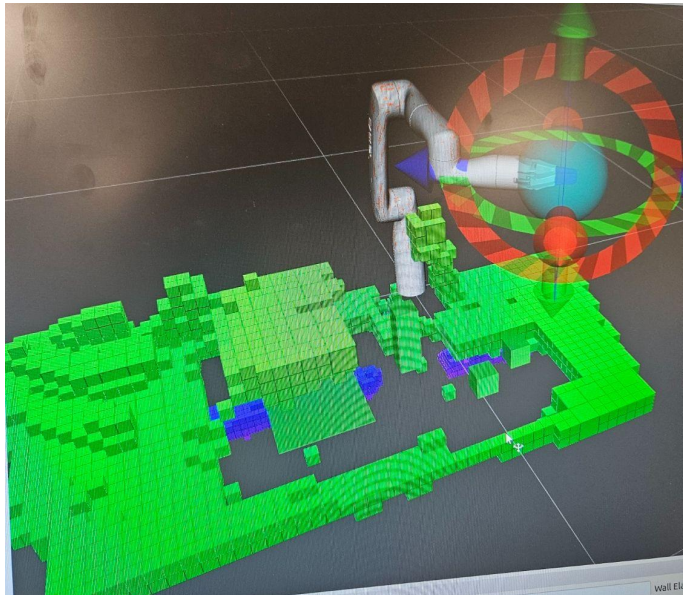
Software Architecture



Planning with MoveIt

> Depth Map Collision and Avoidance

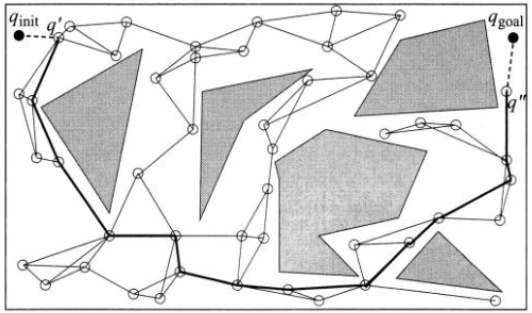
Intel D435 Depth Camera



Moveit Octomap from D435 depth data

Planning with MoveIt

> Path Planning & Grasping Technique



PRM Planner: Finding a Path

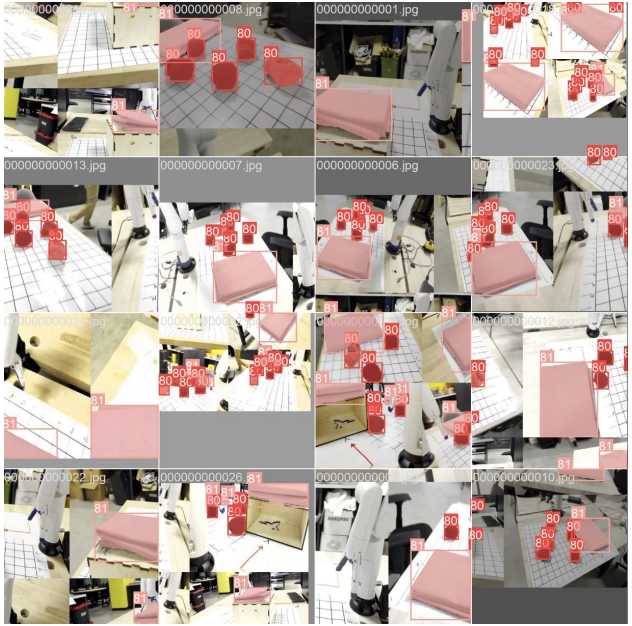
- OMPL(Open Motion Planning Library) consists of many state-of-the-art sampling-based motion planning algorithms.

- Probabilistic Roadmap Method (PRM): sampling-based algorithm. It attempts to connect states to a fixed number of neighbors, PRM gradually increases the number of connection attempts as the roadmap grows in a way that provides convergence to the optimal path.

- Constraints for Gripper
  - MoveItGoalBuilder includes a method called add_path_orientation_constraint(orientation_constraint), which takes in a constraint
  - For the bottle to not flip upside down and cause placing failure, we added orientation constraints to keep the arm upright.

Detecting and Classifying Objects

- Trained an Yolo-V8 model (yolov8l-seg) for object detection and segmentation.
- Calculate and return segmentation masks, bounding boxes, and centroids to other modules.
- Accuracy: ≈ 95%



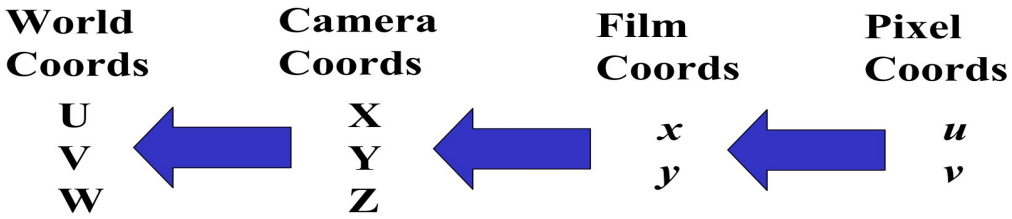
Transformation Pipeline

> With Kinova Arm

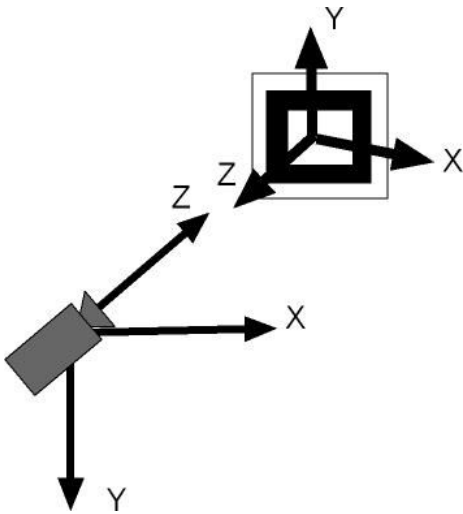
- Fixed position and orientation of the ArUco marker relative to the base link of the Kinova robotic arm.
- Determining the static_transform_publisher node for broadcasting the transformation from the ArUco marker frame to the base link of the Kinova robotic arm.
- Obtaining offset between the ArUco marker frame and the base link of the Kinova robotic arm.
- Using Aruco ROS(Perspective-n-Point (PnP) problem-solving approach) package for converting Coordinates from Camera frame to Aruco frame.

Transformation Pipeline

> With YOLO



- Transformation Steps:
  - Obtain pixel coordinates (x_pixel, y_pixel) from YOLO object detection results
  - Extract depth value at pixel coordinates from aligned depth image
  - Convert pixel coordinates to normalized image coordinates:
    - x_normalized = (x_pixel - cx) / fx
    - y_normalized = (y_pixel - cy) / fy
  - Convert normalized image coordinates to camera coordinates:
    - x_camera = x_normalized * depth
    - y_camera = y_normalized * depth
    - z_camera = depth



Pick and Place Evaluation

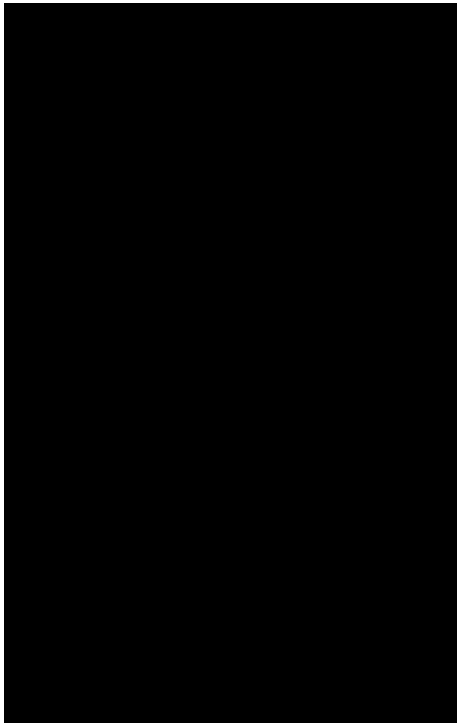
Command parameters: <object1_name> <Quantity 1> <object2_name><Quantity 2> …

Item	Pick Success Rate (%)	Pick Success Rate with GT position (%)	Place Success Rate (%)	Place Success Rate with GT position (%)
Socks	**66**	**100**	**100**	**100**
Bottle	**100**	**100**	**100**	**100**
Blanket	**66**	**100**	**66**	**100**

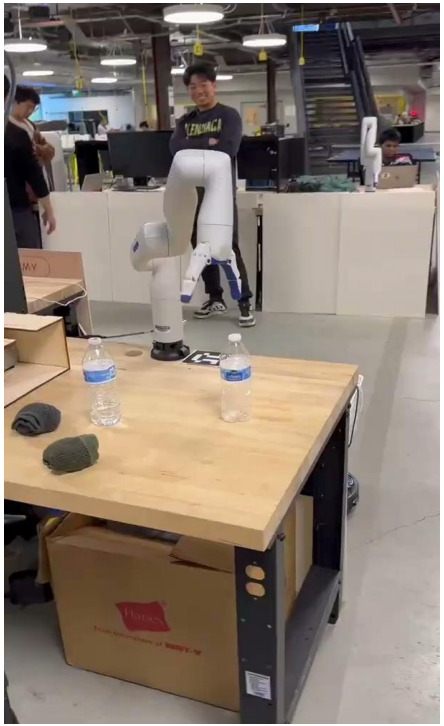
The evaluation is performed using (1) 3 pairs of socks, 3 bottles within one command; (2) 6 times blanket

- **Error in positional estimation** by vision recognition is the major cause for failed “Pick” action
  - Center bias because of tilted bounding box
- Sometimes the blanket holder overlaps with the octomap, causing the pick planning to fail

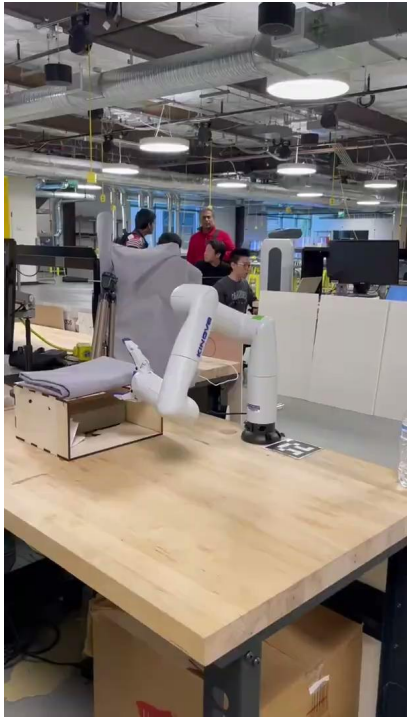
Demo



Socks



Bottle



Blanket

Navigation

Haokun Feng, Mingheng Wu, Hengjun Zhang, Xinyi Zhou,
Kate Chen, Matthew Zhang

Link

Navigation Goal

Senario:

Accept commands from the Inoribit to autonomously transport items grabbed by the manipulation team to a designated location, and wait for the receiver to take the items.

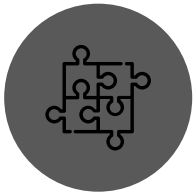
Goal 1:

Achieve accurate GIX mapping through lidar scanning basing on the robot movement.

Goal 2:

Enable the robot to perform smooth navigation and cargo transportation tasks.

Challenge



Hardware

Have never operated or programmed a create robot before, and there is relatively limited information available online.

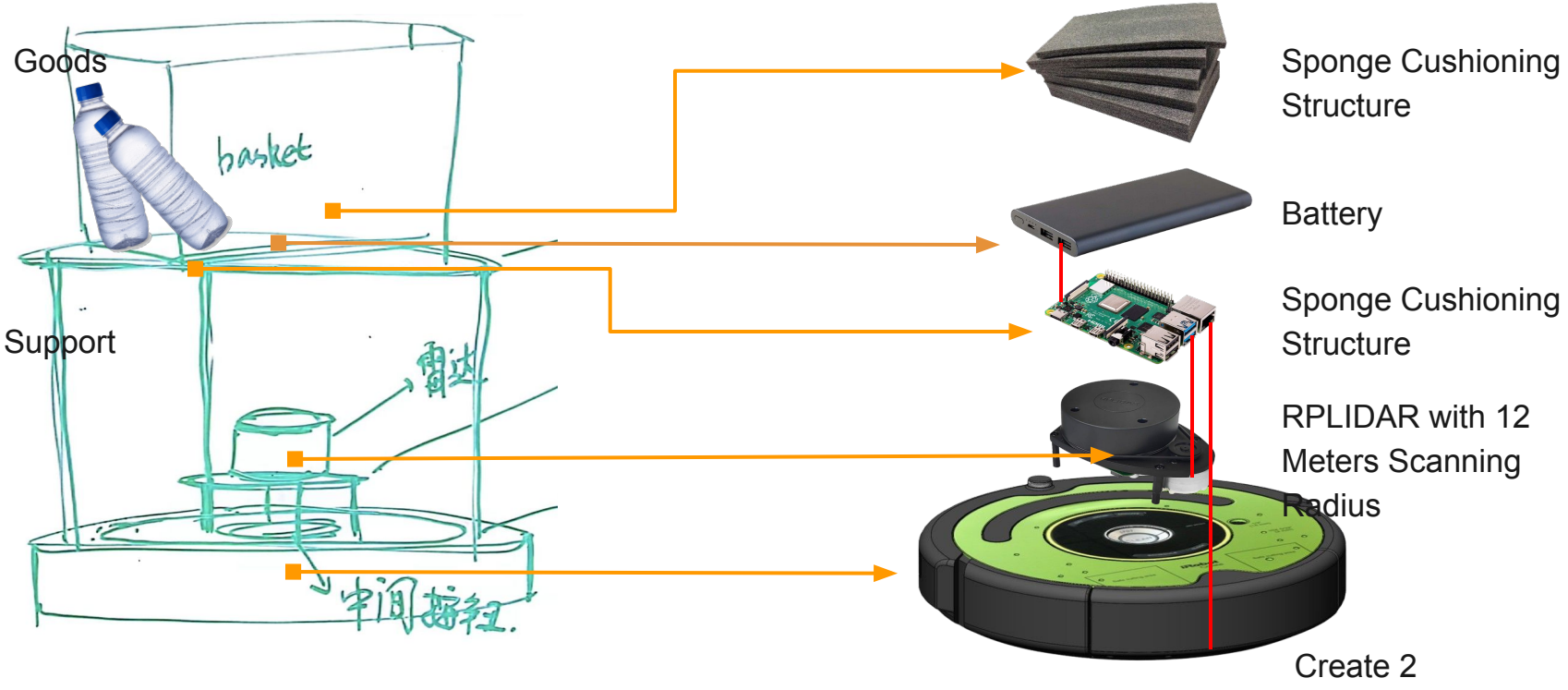
Precise Mapping

Precision control of lidar for creating mapping, also, The robot and lidar need to operate at low altitudes, navigating a relatively complex environment.

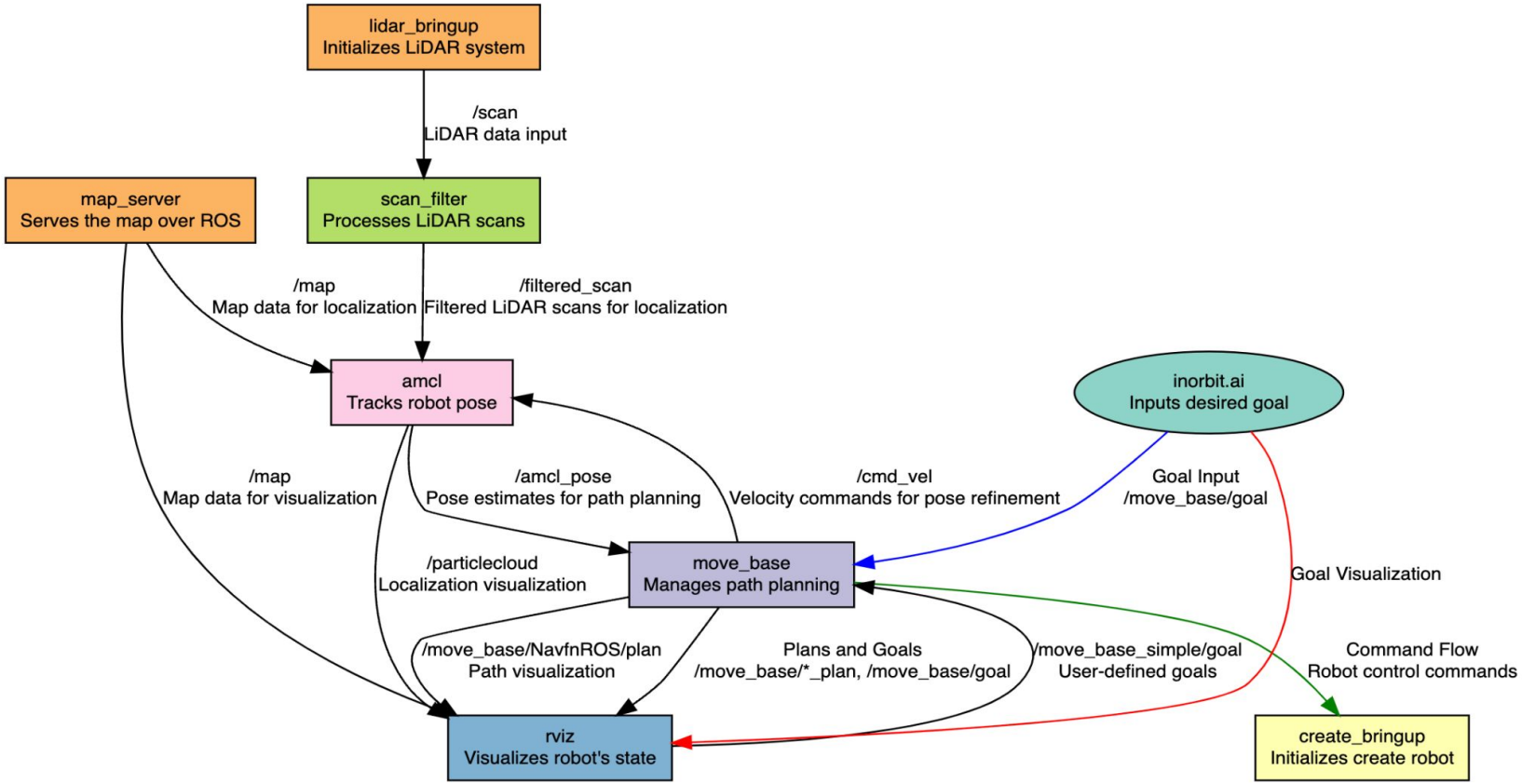
Parameters Tuning

Navigation parameters such as costmap Amcl and planner have a significant impact on the navigation performance of robots, but there is no clear guideline to determine which parameter is effective. Therefore, the impact of different parameters can only be measured repeatedly in experiments. Ultimately determine the parameter with the highest success rate.

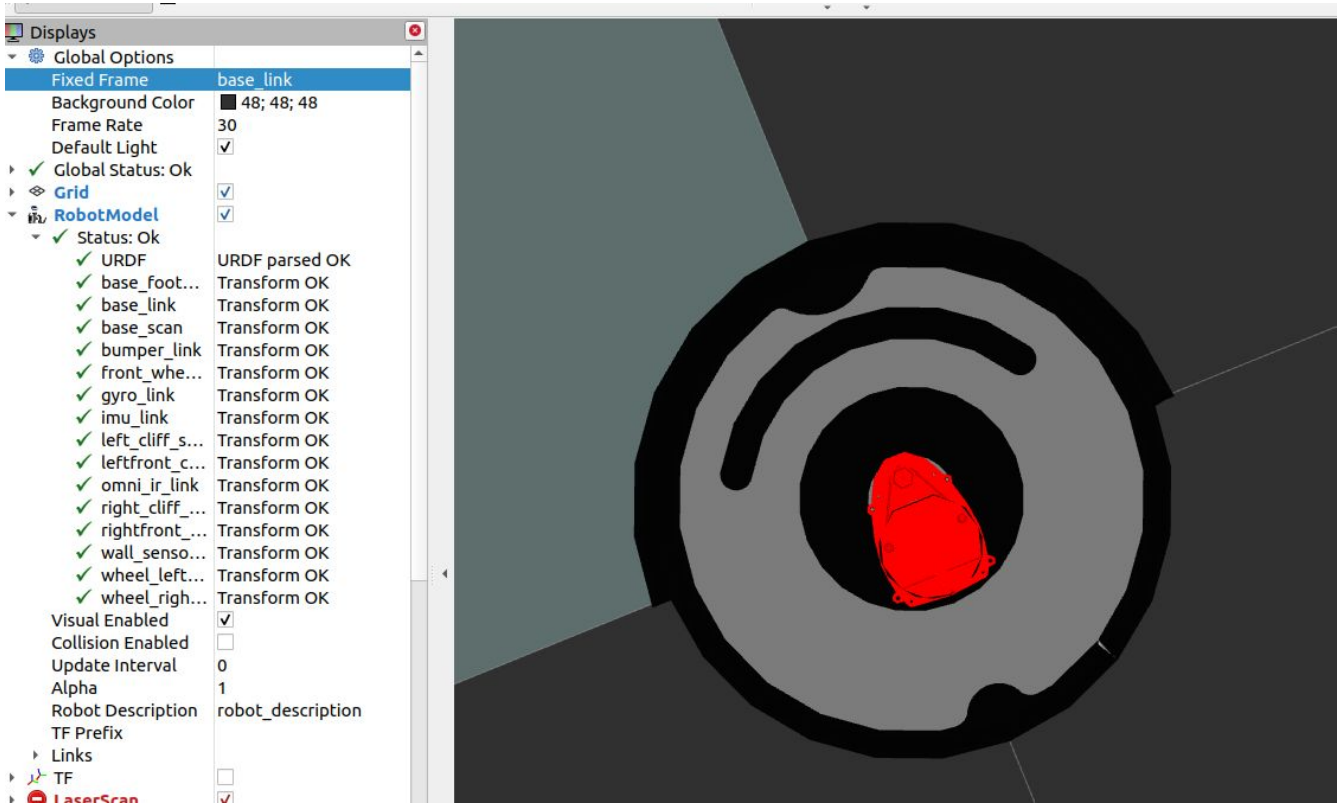
HW & Product Design



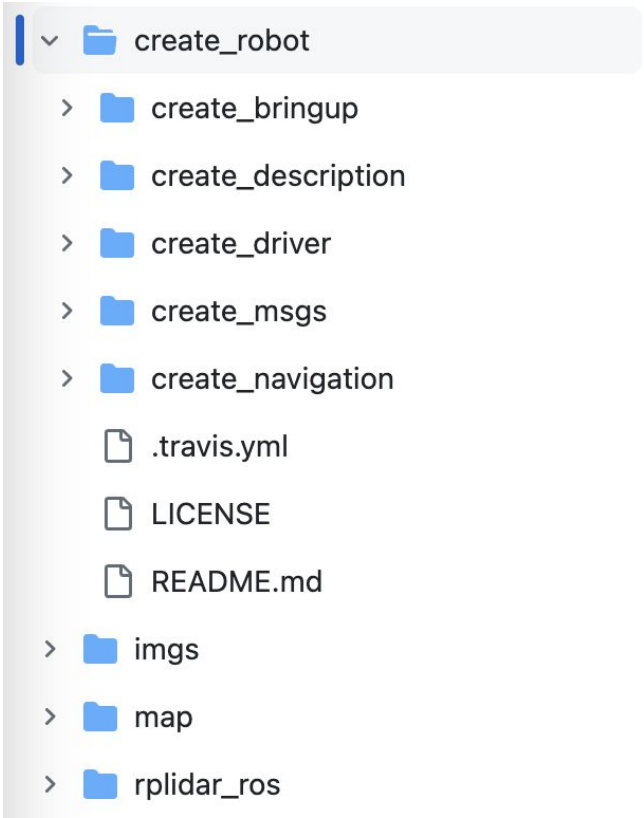
SW Architecture

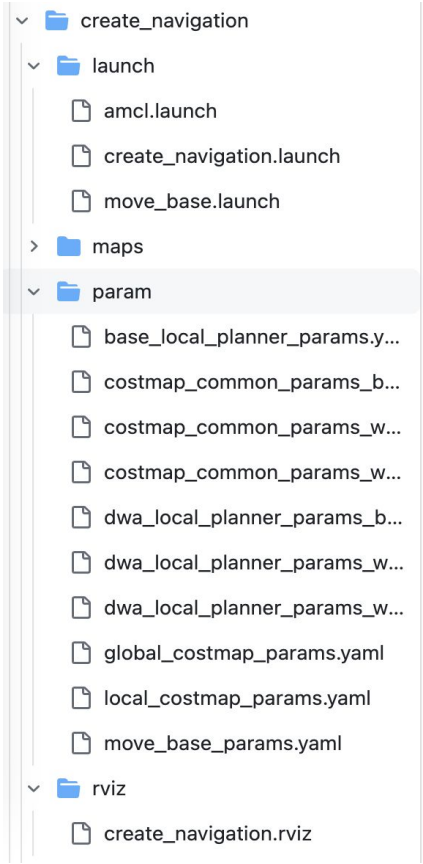


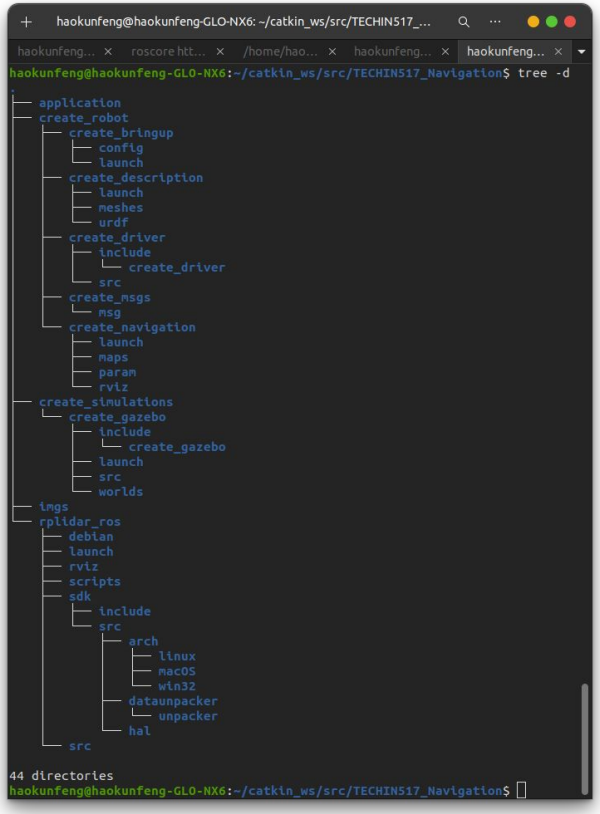
SW Architecture: Physical visualization



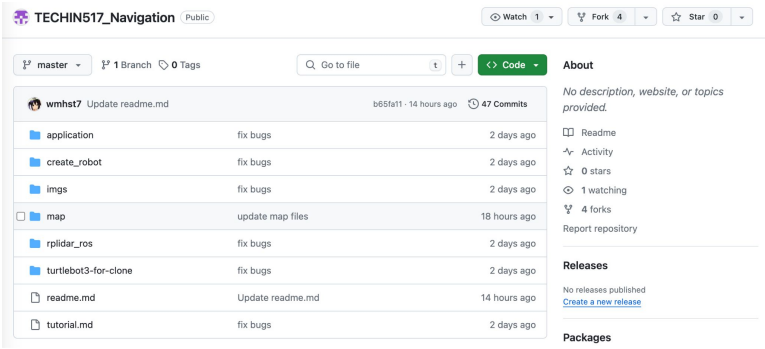
SW Architecture: Implement package for Create2 and Lidar for navigation







SW Architecture: Open source package for other users



https://github.com/SwxTemp/TECHIN517_Navigation/tree/gixlaptop

README

Navigation Stack for Create Robot and Rplidar

Usage Instructions

System Requirements

- Ubuntu 20.04
- ROS 1 Noetic

Installation Steps

1. **Install Turtlebot3 Packages:** Ensure that the Turtlebot3 packages are installed on your system.
2. **Clone Repository**

Clone the repository to your ROS workspace (`src` folder).

```
git clone https://github.com/SwxTemp/TECHIN517_Navigation.git
```

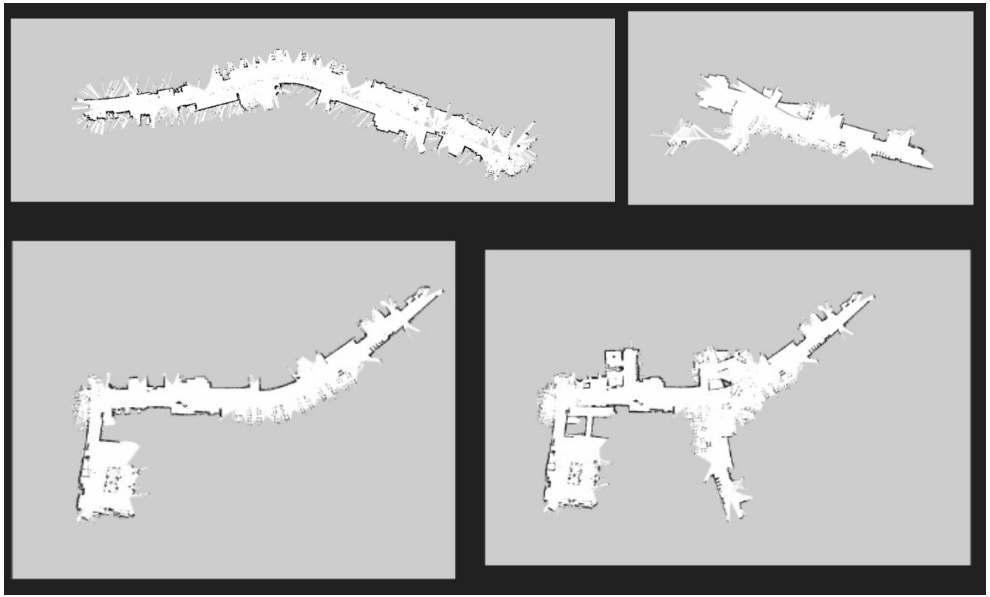
3. **Update USB Serial Port Names:**

- Navigate to the following files:
  - `create_robot/create_bringup/config/default.yaml`
  - `rplidar_ros/launch/rplidar_a1.launch`
- Update the USB serial port names as needed. Typically, one is named USB1 and the other is named USB0.

4. **Build Workspace:**

Map Stage I

Initial Mapping with Gmapping



Method:

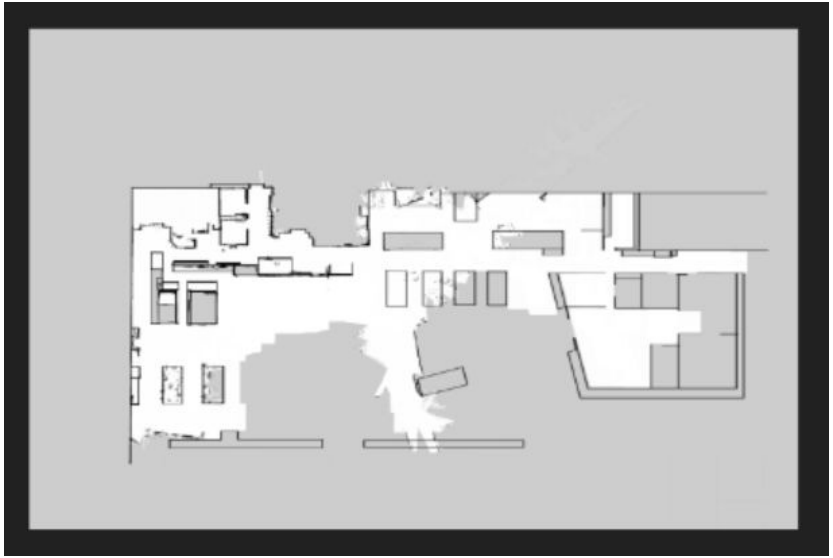
- Utilizes gmapping primarily relying on odometry data for Simultaneous Localization and Mapping (SLAM).
- Employs LIDAR scans for initial map creation.

Challenges:

- Inherent hardware inaccuracies lead to cumulative errors.
- Resulting maps may suffer from distortions due to error accumulation.

Map Stage II

Small-Scale Mapping and Manual Stitching



Method:

- Maps are drawn on a smaller scale using LIDAR, followed by manual stitching of these segments.

Outcomes:

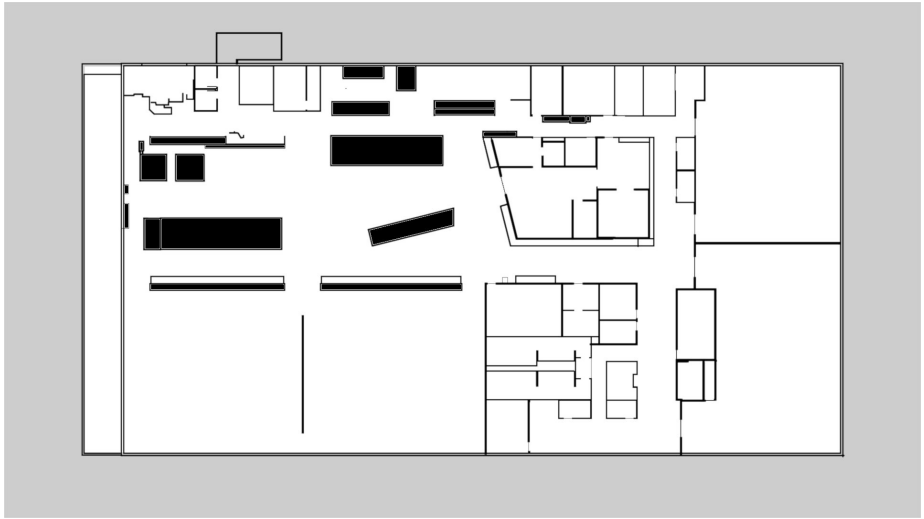
- Reduced overall distortion, bringing the map closer to real-world accuracy.

Challenges:

- Low efficiency in the mapping process.
- Proportional errors can occur during the stitching of map segments.

Map Stage III

Utilizing Construction Blueprints for Reference



Method:

- Maps are refined by comparing and integrating them with construction blueprints.

Outcomes:

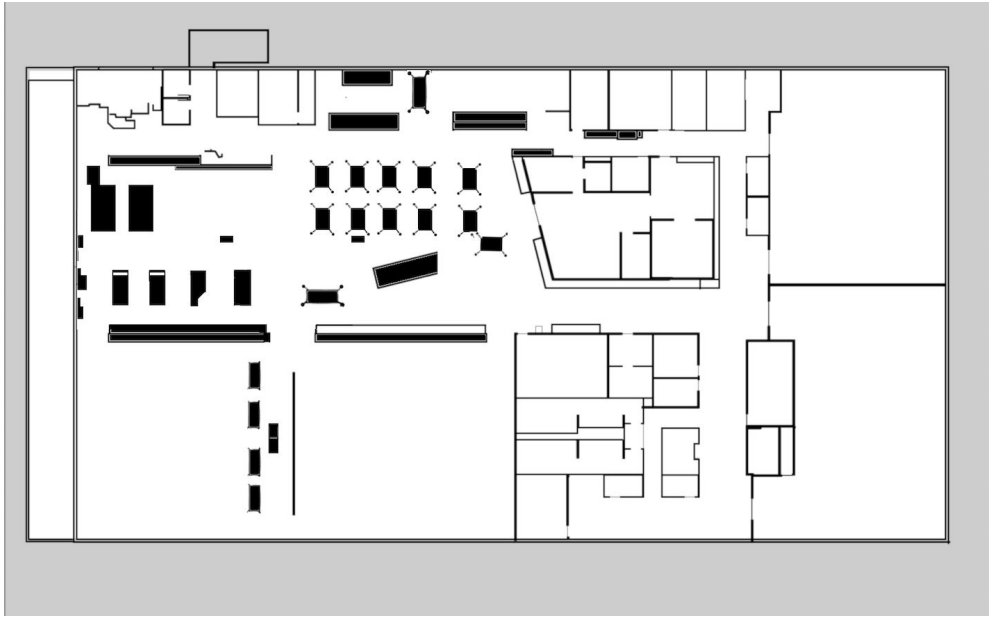
- Significantly improved overall accuracy, especially in the positioning and relationships between walls.

Challenges:

- Inaccuracies in depicting the relationship between movable objects and fixed structures.

Map Stage IV

Combining Methods for Enhanced Accuracy



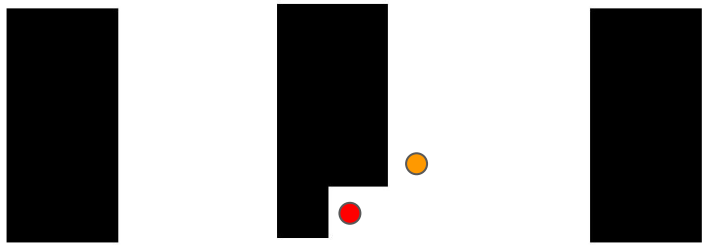
Method:

- Integrates LIDAR scanning with manual measurements to optimize the positioning and dimensions of movable objects like furniture.

Outcomes:

- Highly accurate maps that enhance navigation success and precision.

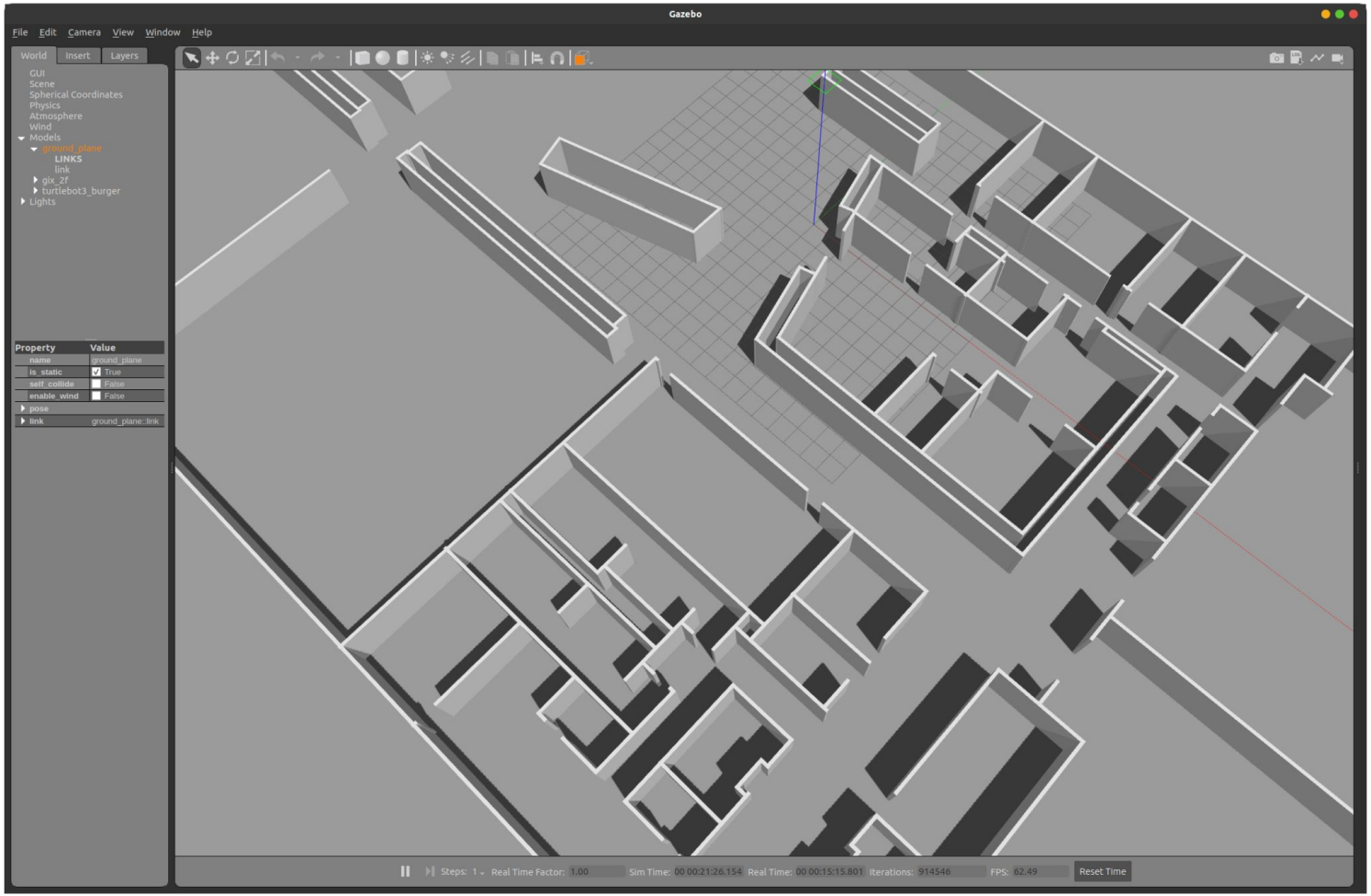
Change of Pickup Station



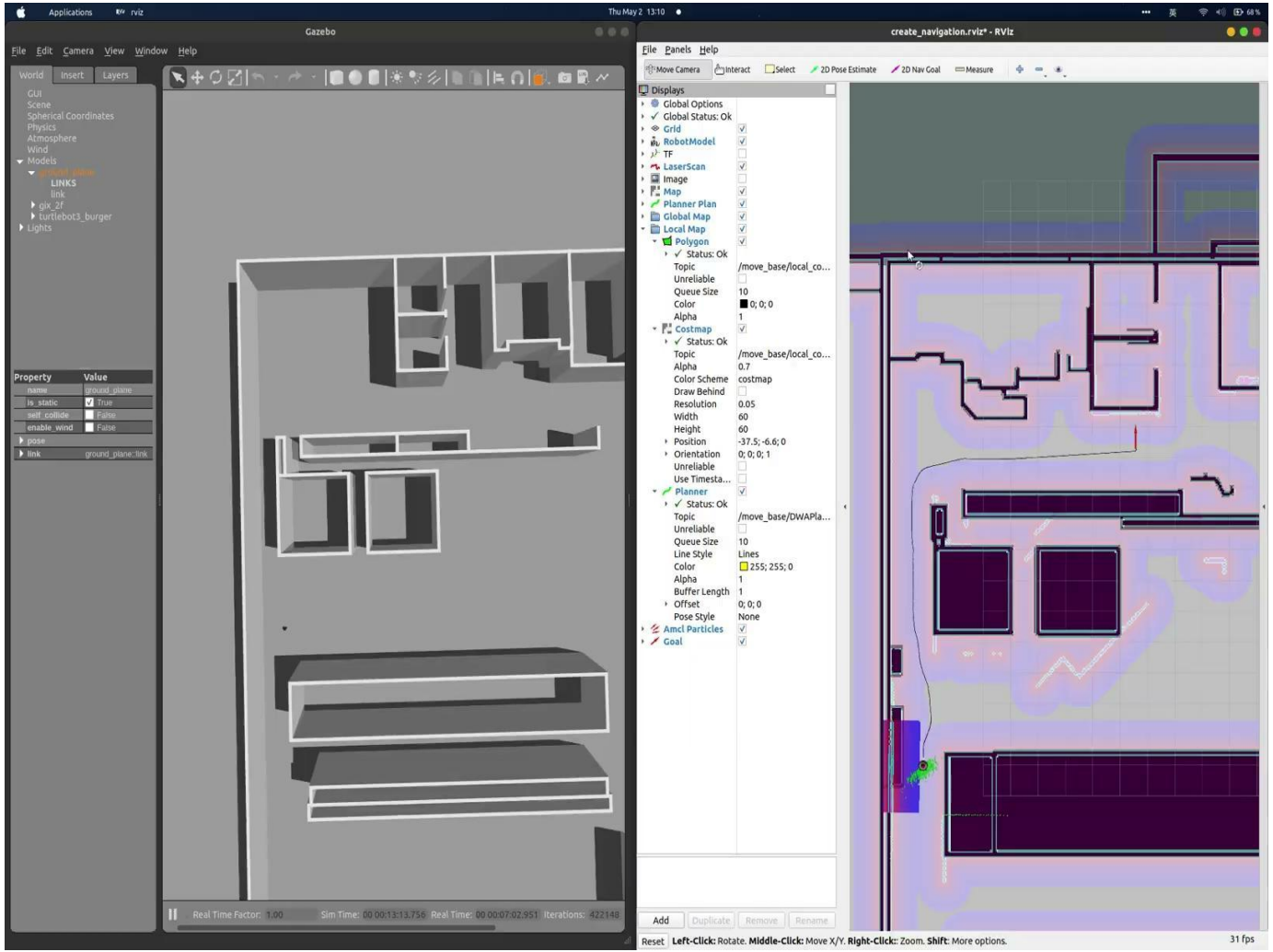
Old Version

New Version

High-Precision World Model with Simulation Package



Simulation Demo



Physical Robot Experiments

Start Point: Kinova Arm (x=16.07, y=-32.44)
Destination 1: Classroom (x=25.57, y=-25.95)
Destination 2: Kinova Arm (x=16.07, y=-32.44)

Experimental Procedure: After calibrating the starting point of the robot on the map, send a fixed coordinate of destination 1 to make it self navigate and navigate to the set coordinate 1; If the target point 1 is successfully reached, send the fixed coordinates of target point 2 to navigate to it on its own. After arriving at the destination, the final error and success rate are determined by the displacement from the fixed coordinates.

Key indicators: success rate, time spent, accuracy of target location

Physical Robot Experiments

Destination	AVG Time Spent	Total Trips	Successful Trips	Success Rate
To Teaching Station	1'46"	10	6	60%
To Kinova Arm	1'52"	10	2	20%

Before Parameters Tuning

Inferred reasons for low success rate:

1. The **map is not accurate** enough, and there is an error in the relative position between the table and the wall;
2. **Tables and miscellaneous items have errors** in capturing spatial features by robots;
3. There is an issue with the **navigation parameters** of the robot itself.

Parameters Tuning

- **base_local_planner_params.yaml**
  - xy_goal_tolerance=0.1, yaw_goal_tolerance:0.2
- **costmap_common_params_burger.yaml**
  - inflation_radius=0.2, cost_scaling_factor=4.0
- **dwa_local_planner_params_burger.yaml**
  - xy_goal_tolerance=0.1, yaw_goal_tolerance:0.2

In the experiment, we attempted to modify many parameters, and after testing, the above parameters were the most useful ones

Parameters Tuning

xy_goal_tolerance: This parameter defines the tolerance of the robot's navigation process towards the target position. Specifically, it determines the allowable error range for the robot to reach the target position during navigation tasks. The smaller the parameter, the more accurate the relative target position of the robot after reaching the destination

yaw_goal_tolerance: yaw_goal_tolerance defines the allowable range of orientation error (in radians) for a robot to reach its target position. The smaller the parameter, the more precise the orientation will be when reaching the target position. However, counterintuitively, if this parameter is smaller, the robot will take a long time to adjust after reaching its destination, so this parameter should be adjusted larger.

inflation_radius: infliction_radius is a key parameter used to set the radius of the obstacle expansion layer. The smaller this parameter, the closer the robot can approach the boundary of the obstacle.

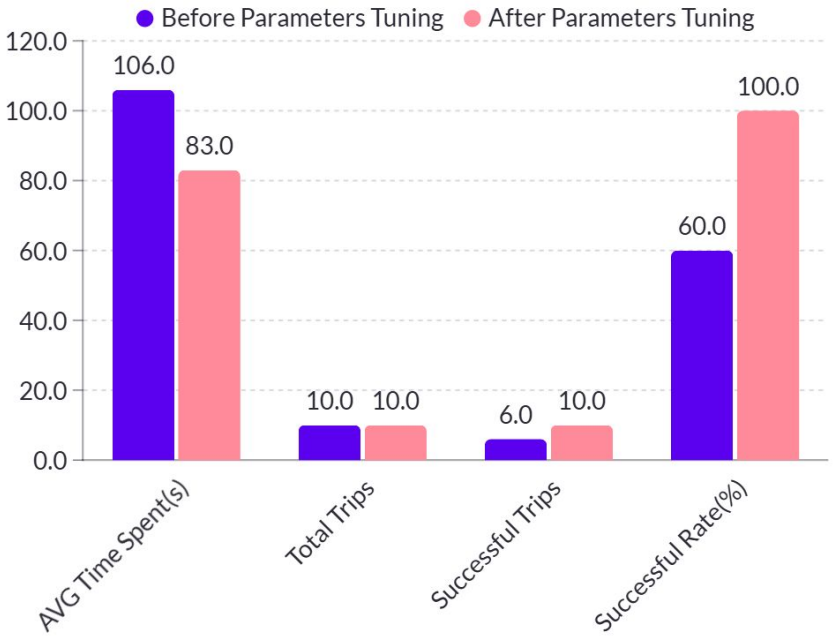
cost_scaling_factor: cost_scaling_factor is used in robotic navigation to regulate the growth rate of Cost Value in Inflation Layer.It defines how the value of the expansion area is attenuated from the edge of the obstacle.The lower Cost_Scaling_factor will increase the value of the generation, and the robot may choose the path closer to the obstacle

Physical Robot Experiments

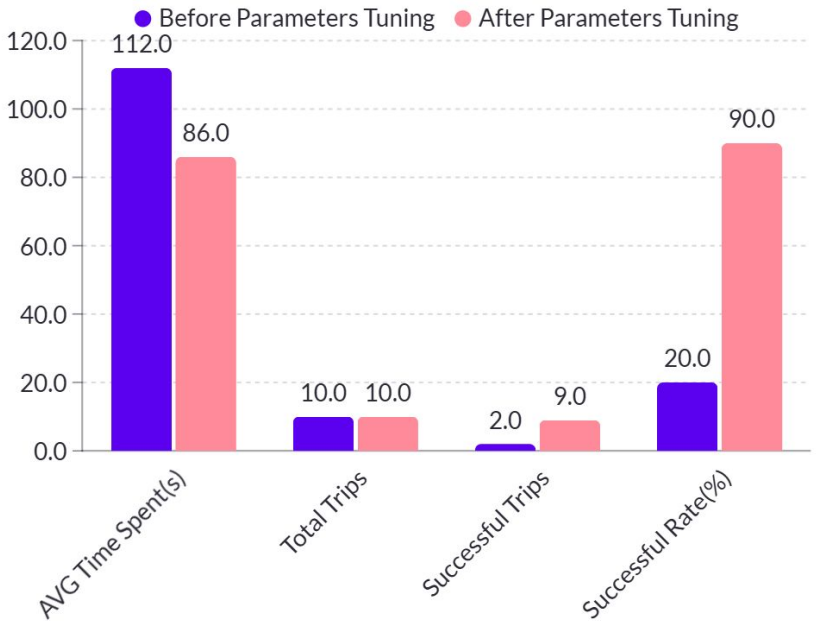
Destination	AVG Time Spent	Total Trips	Successful Trips	Success Rate
To Teaching Station	1'23"	10	10	100%
To Kinova Arm	1'26"	10	9	90%

After Parameters Tuning

Physical Robot Experiments

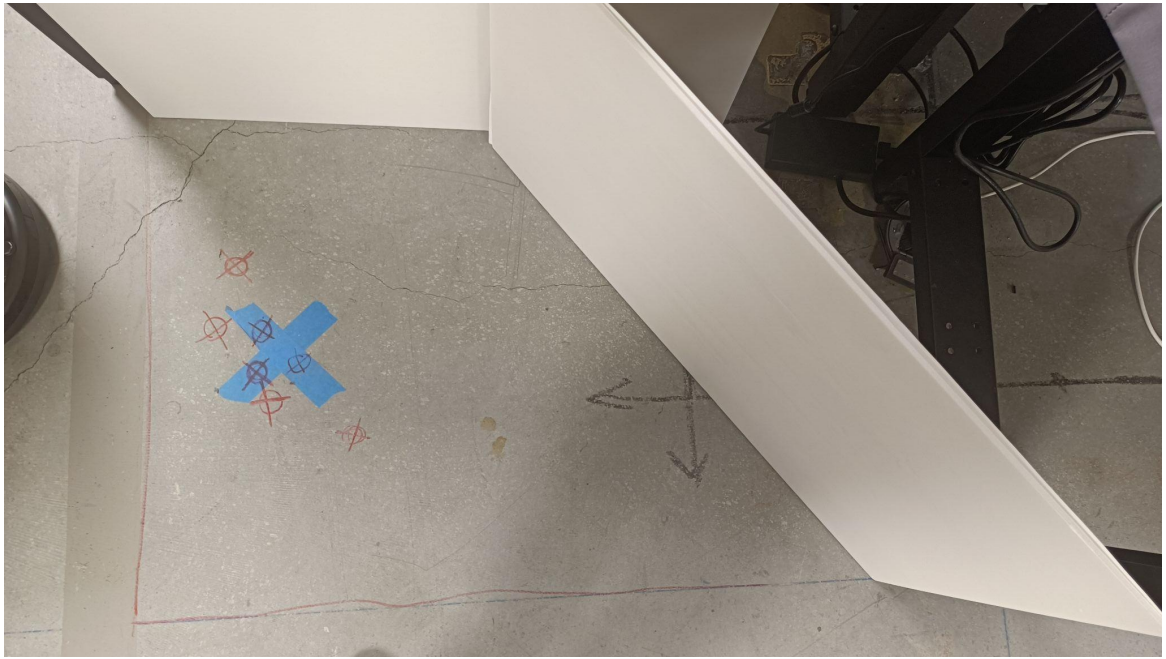


Comparison Experiment To Classroom Teaching Station



Comparison Experiment To Kinova Arm

Physical Robot Experiments



After parameter tuning and testing, the robot reaches the final target position with an error within the allowable range

Physical Robot Demo