ROS Navigate Pick and Place

Student	Christy Jo Manthara - christyjo.manthara@studenti.unipd.it
	Maulik Rupeshbhai Sompura - maulirupeshbhai.sompura@studenti.unipd.it
Professor	Emanuele Menegatti
Project Repository	https://bitbucket.org/ir2324-group-30/assignment2/src/main/
VideoLink	https://drive.google.com/drive/folders/1RhaeZrJ6Qr4cn0sTnF6AyT7s0P7QpOwv?usp=drive_link

Introduction

The goal of this project is to implement a fetch and delivery behavior in an assistance robot Tiago, within a simulated environment. The task involves picking up colored objects from a table and delivering them to corresponding colored cylindrical tables, while avoiding collisions with gold hexagons which act as obstacles.

Task Requirements

- Develop a routine using MoveIt! to pick and place objects according to a given sequence.
- Use Tiago's camera and AprilTag library to detect objects and their poses.
- Define collision objects for the table and the obstacles to generate collision-free trajectories.
- Implement a grasping routine that can handle objects from the top or side as necessary.
- Use the Gazebo_ros_link_attacher plugin to virtually attach objects to the gripper.

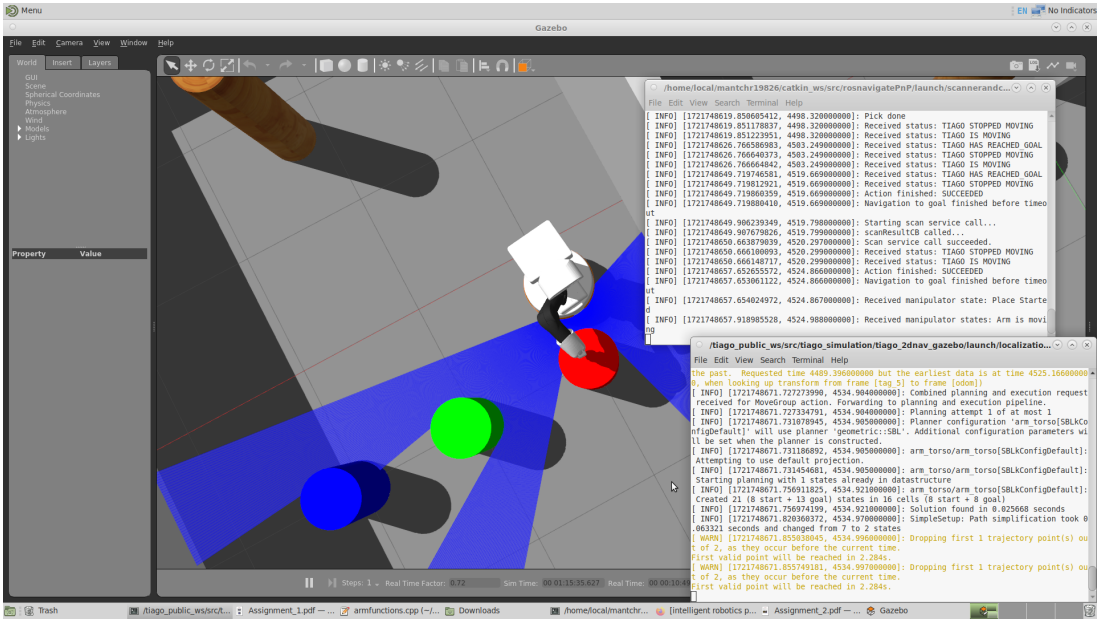
Project Overview

The main functionality of the code is divided into several nodes and functions, each responsible for a specific part of the fetch and delivery behavior. Here is a detailed explanation of the approach and algorithm used in the project:

Algorithm Steps

1. **Initialization**:
    - All the nodes are started by using 2 launch files.
    - Initialize the human node inside the tiago_client and use it to get the objects to be identified to the user. The entire operation fails and shuts down ros if the human_client is not operational.
2. **Running nodes**:
    - The Navigation is handles by the Tiago_server
    - manipulation of the arm and torso is handled by the manipulation_node.
    - The detection of april tags is done by the detection_node which uses the aprildetect.srv file
    - The scanning of the barrels and the environment is done using the scan_node which uses the barrelscan.srv file
3. **Object Detection**:
    - Use the AprilTag library to detect objects on the pick-up table.
    - Retrieve the poses and IDs of the detected objects.
    - Transform object poses from the camera frame to the robot frame.
4. **Adding Collision Objects**:
    - Add collision objects to the planning scene for the table and obstacles.
    - Create appropriate collision shapes (cylinders and boxes) based on detected object properties.

5. **Grasping Objects**:
    - Open the gripper as a safety measure before approaching the object.
    - Plan and execute the arm movement to approach the object.
    - Plan and execute the arm movement to grasp the object, ensuring the gripper closes securely around the object.
    - Attach the object to the gripper using the Gazebo_ros_link_attacher plugin.
6. **Placing Objects**:
    - Plan and execute the arm movement to move the object to the corresponding colored cylindrical table.
    - Plan and execute the arm movement to place the object on the table.
    - Open the gripper to release the object.
    - Detach the object from the gripper.



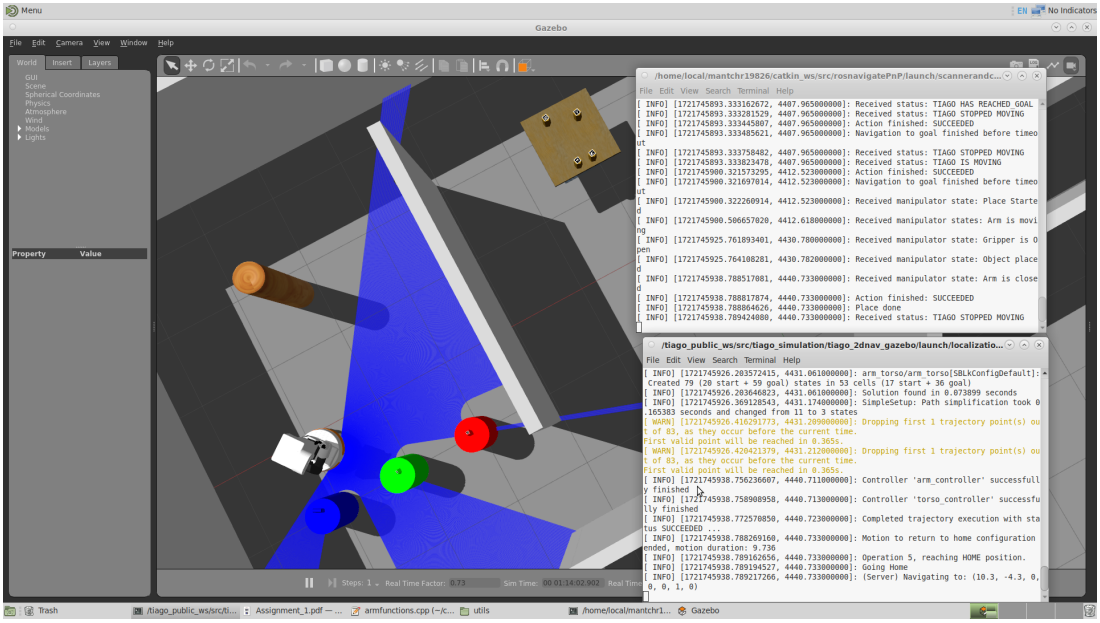
7. **Navigation**:
    - Navigate Tiago to the pick-up table and delivery tables using a predefined navigation stack.
    - Implement collision-free paths for moving objects.

Detailed Functionality

- **Object Detection** (`apriltagdetected.cpp`):
    - Moves the head up and down before the detection and after the detection using the control_msgs::PointHeadAction.
    - Detects AprilTag markers on the objects.
    - Calculates the pose of each object relative to the robot.
- **Arm Manipulation** (`arm_manipulator.cpp`):
    - **addCollisionObjects**: Adds the table and obstacles as collision objects in the planning scene.
    - **pick**: Handles the sequence of opening the gripper, approaching the object, grasping it, and attaching it to the gripper. Pick follows the following sequence of actions: plan approach/depart,create the goal pose, opening the gripper, create plans for approach, attach the object, close the gripper, depart to home. We have used moveit::planning_interface::MoveGroupInterface for the movement of the arms and the grippers of the tiagobot. Further we have used moveit::planning_interface::PlanningSceneInterface interface allows for the addition, removal, and updating of objects in the planning scene.

  - **place**: Handles the sequence of moving to the delivery table, placing the object, and detaching it from the gripper. It follows the same action ad before except for the fact that it opens the gripper on goal. Then it goes back home.

- **Node Communication** (`tiago_client.cpp` and `tiago_server.cpp`):
  - `tiago_client`: Sends requests to perform pick and place actions. Sends requests to the armmanipulator, detector and scanner when necessary. These are done with the help of srv files which are populated from the client in a requestresponse format. The human client is called from here.
  - `tiago_server`: Handles the execution of pick and place actions based on requests received. Adds waypoints for each location on the map where the Tiago can move to execute the functions as called in the sequence.
- **Barrelscanner** (`barrelscanner.cpp`):
  - Uses the /xtion/rgb/image_raw to detect the colors from the RGB camera of the Tiago.
  - The function processImage converts incoming images to OpenCV format using cv_bridge.
  - Function detectColorOrder divides the image into regions of left, center and right and processes each region to find the dominant color.
- **Services** (`Aprildetect.srv` and`barrelscan.srv`):
  - `Aprildetect.srv`: The requests are a boolean to check the readiness for the detection and the apriltag id which is an integer. The response and an apriltagdetection array called detections.
  - `barrelscan.srv`: The requests are a boolean scanready and a Laserscan message. The responses are the cylinder poses which is a vector of geometry poses and the integer array to determine the cylindercolor.
- **Launch files**:
  - `RosnavigatePnP.launch`:This launch file is used to run the world, apriltag launch, navigation launch, human node, manipulation node and detection node.
  - `scannerandclient.launch`: This launch file launches the scanner and tiago_client nodes.



Handling Specific Objects

- The blue object (ID 3) required specific adjustments to the gripper closing values to ensure it is securely grasped without slipping.
- The code was reviewed and fixed to ensure the correct attachment and feedback during the grasping process.

Challenges and Solutions

- **Object Slipping**: The blue object was dropping after being picked up. This was fixed by adjusting the gripper closing values and ensuring the object was securely attached to the gripper. Furthermore, the velocity of the arm movements was reduced down to 0.5 using the function setMaxVelocityScalingFactor().
- **Collision Avoidance**: Adding accurate collision objects for the table and obstacles ensured collision-free paths during the pick and place actions.
- **Order of nodes execution**: The order in which the nodes are to be run caused multiple problems. This was solved by adding necessary checks to each of the nodes and necessary logging with the files. In particular the scanner node failed to launch multiple times so we shifted it to another launch file and ensured that it ran before the client node started so that the process runs smoothly.
- **Service request and response**: Managing the service request and response was a difficult task especially in choosing the parameters to be passed and ensuring that they reach the required nodes. Further we made use of the "rosnode list" and "rosservice list".

Conclusion

This project successfully implemented a fetch and delivery behavior in Tiago using ROS and MoveIt!. The approach involved detecting objects using AprilTag markers, adding collision objects, planning and executing arm movements for grasping and placing objects, and navigating the robot to avoid collisions. Adjustments were made to ensure the blue object was securely picked up and placed, addressing initial issues with object slipping.

Project Work Division

Christy Jo Manthara's Responsibilities

- Development and implementation of the server-side architecture, including `tiago_server.h` and `tiago_server.cpp`.
- Path planning and execution of navigation commands on the server side.
- Development and integration of the object detection component, including `detector.h` and `detector.cpp`.
- Apriltag detection with the file apriltagdetected.cpp
- Continuous updating and refining of the navigation path based on real-time feedback from the detection system.
- Handling callbacks and configuring the working of the files in the proper sequence as well as the launch files.
- Creation of srv files, the way they are connected to various components and the manner in which they are populated.

Maulik Rupeshbhai Sompura's Responsibilities

- Development and implementation of the `TiagoMove.action` and file.
- Implementation of the client-side architecture, including the development of `tiago_client.h` and `tiago_client.cpp`.
- Establishing and testing ROS actions for client-server communication.
- Managing and handling the feedback mechanisms within the client component.
- Working with the scanner node which can detect the barrels and decide which one to go there.
- Using the `/xtion/rgb/image_raw` topic to publish the images captured by the RGB camera of the tiago