RoboCup@Home - Carry My Luggage Final Report

Margaux Edwards, David Martinez, Xuhao Jin
[1]Practical Course RoboCup@Home Winter Semester 2023
[2]Institute for Cognitive Systems, Technical University of Munich (TUM), Karlstraße 45/II, 80333 Munich, Germany
✉ go98baq@mytum.de
February 14, 2024

Abstract — This report presents the development process for the *Carry My Luggage* task within the 2024 RoboCup@Home competition, employing the TIAGo platform. It encompasses the integration of advanced robotic capabilities such as perception, manipulation, navigation, and human-robot interaction. Key achievements include the autonomous detection and navigation to pick up luggage (bag), obstacle avoidance, and following the operator to a designated location. The project's innovation lies in its comprehensive approach to addressing the competition's challenges, demonstrating significant advancements in robotics applied to real-world scenarios. Future work aims to expand the robot's capabilities for broader applications in upcoming RoboCup seasons. The github repository for this project can be found at GitHub RoboCup@Home Repo and full video Final Demonstration

1 Introduction

The RoboCup@Home league serves as an influential arena within the RoboCup competition, hosting an annual international contest aimed at pushing the boundaries of service and assistive robotics. This platform facilitates the development and benchmarking of robots in a range of tasks designed to simulate real-world scenarios. Robots are assessed on capabilities such as human-robot interaction, autonomous navigation in changing environments, computer vision, object manipulation, and more, thereby promoting advancements in robotics technology and its application in daily life [1]. This report documents the TUM RoboCup@Home Team's contribution to the *Carry My Luggage* challenge. Using the TIAGo robot platform developed by Pal Robotics [2], this project focuses on enhancing the robot's ability to assist a human by carrying luggage, a bag to a designated location. This task not only showcases the integration of various robotic functionalities—such as perception, navigation, and manipulation—but also emphasises the importance of human-robot collaboration, a critical aspect of future robotics applications in service and assistance domains.

Fig. 1 TIAGo Robot Platform used for the Open Platform RoboCup@Home competition [2]

1.1 Task Definition

Carry My Luggage involves a robot helping an operator carry a bag to a car parked outside, encapsulating the essence of service robotics in assisting with everyday tasks. This challenge is divided into distinct stages, each focusing on a critical aspect of the task:

1. **Stage 1: Picking up the bag**, where the robot must identify and collect the luggage indicated by the human operator.
2. **Stage 2: Following the operator** to the destination, requiring robust person tracking and navigation capabilities.
3. **Stage 3: Navigating through obstacles**, testing the robot's ability to deal with real-world environmental challenges.
4. **Stage 4: Re-Entering the Arena** (as a bonus task), where the robot must find its way back to the starting point.

The team's approach to this challenge emphasises the development of autonomous bag detection and retrieval in Stage 1, and effective human following in Stage 2, presenting a comprehensive solution that pushes the boundaries of what service robots can achieve in real-world scenarios. Unlike previous iterations of the carry my luggage challenge, it was decided that as a stretch goal, stage 1 would be modified to:

Stage 1: The robot autonomously detects, navigates to (while avoiding obstacles) and picks up the bag, rather than being handed the bag as seen in previous RoboCup@Home demonstrations [3].

Furthermore, as indicated in the rules, the main focus for this challenge was completing the followign goals: *person following, navigation in unmapped environments, social navigation* [1] consequently, these became the criteria for the engineering challenge for the team and informed the creation of the task's scoresheet as illustrated in Figure 2.

Action	Score
Main Goal	
Picking up the correct bag	*100*
Following the person to the car	*300*
Avoiding the crowd of people obstructing the path	*50*
Avoiding the small object on the ground	*50*
Avoiding the hard-to-see object	*50*
Avoiding the area blocked with retractable barriers	*50*
Bonus rewards	
Re-entering the arena	*100*

Fig. 2 The RoboCup@Home 2024 Scoresheet [1]

1.2 Task Breakdown

To tackle the complexity of the *Carry My Luggage* challenge within a limited timeframe, the task was systematically divided into key areas, each contributing significantly to the overall goal:

1. **Perception**: The foundation of the robot's ability to interact with its environment, focusing on the identification of bags, humans, and obstacles. This subsystem is crucial for enabling the robot to understand its surroundings and make informed decisions.
2. **Navigation**: This involves the robot's movement towards identified targets and its capacity to map and traverse unknown environments while avoiding obstacles. Efficient navigation is essential for the robot to follow the operator accurately and reach the destination successfully.
3. **Human Interaction and Manipulation**: Encompassing speech interaction for intuitive communication with humans and the physical manipulation of objects like bags. This aspect bridges the gap between technical functionality and user-friendly operation, facilitating seamless human-robot cooperation.

2 System Overview

2.1 Robotic Platform

The TIAGo platform is an off-the-shelf service robot as illustrated with its main features in Figure 1. Important to the developed solution were the 7 DoF arm with gripper, RGB-D camera and 2D LIDAR. For software development, ubuntu 20.04 [4] was used with ROS Noetic [5] as the middle-ware with Python and C++ utilised to build additional capabilities. For simulation, Gazebo [6] was used to test and develop the system before being deployed on the physical hardware using the PAL open source TIAGo tutorials and model [7]. The TIAGo platform has featured successfully before in the RoboCup@Home competition with CATIE [8] and homer@UniKoblenz [9], both placing teams in past RoboCup@Home competitions. The following sections explain the technologies or frameworks used for perception, manipulation, navigation, and interaction for the final proposed solution to the *Carry My Luggage* challenge.

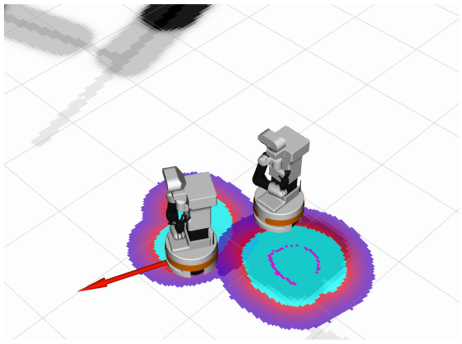

Fig. 3 Example Human_Follow in Gazebo environment with two TIAGo platforms

2.2 Perception

2.2.1 Vision Node

The vision system setup encompasses the integration of depth and RGB modules on the robotic

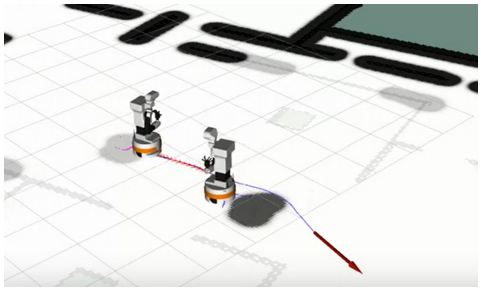

Fig. 4 Example Path Planning with obstacle avoidance in Gazebo environment with two TIAGo platforms

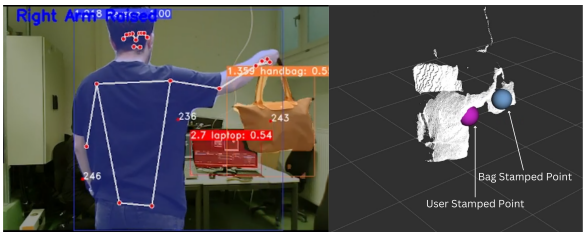

Fig. 5 Bag, person and obstacle detection via YOLACT plus arm raise detection (left image), published cropped point cloud and stamped points (right image).

platform, essential for object detection and human following tasks. Depth images, obtained from the raw depth frame topic, provide crucial depth information, while RGB images, captured from the raw RGB image topic, facilitate human, obstacle and pickup target detection.

Within the obstacle detection node, the YOLACT CNN model [10] serves as a pivotal component, enabling the identification of obstacles, user persons, and items designated for pickup. YOLACT generates per-class bounding boxes, which are utilized to compute the centroid of all detections, along with per-class masks that facilitate the cropping of the depth frame, isolating depth information pertinent to detected objects. The depth information derived from the masked frames aids in detecting the person closest to the robot, while other obstacles are republished as a new depth frame after being cropped with masks. Additionally, a secondary model from the Mediapipe library [11], dedicated solely to the closest person (user), detects human body poses, specifically identifying the raised arm. This arm detection mechanism assists in determining the luggage the person intends to pick up, distinguishing between the left and right sides, see Fig.5.

The purpose of this vision node extends beyond mere detection and tracking; it orchestrates a comprehensive perception framework aimed at facilitating human-robot interaction and task execution

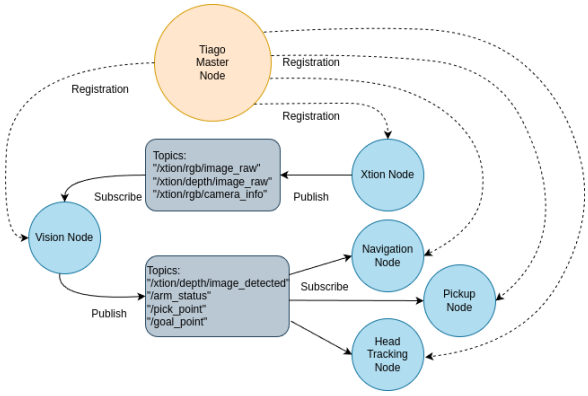

Fig. 6 ROS vision node architecture.

seamlessly. Subscribing to the RGB frame, depth frame, and camera parameters, the vision node processes incoming data streams to extract valuable insights. Following processing, the node publishes several key components essential for the robot's decision-making and action execution. These include the 3D centroid of the bag item, crucial for navigation and interaction; the 3D centroid of the user person nearest to the robot, serving as a goal for navigation tasks; a new masked depth image containing only the detected objects, streamlining subsequent processing steps; and a string denoting the status of the user's raised arm (left/right/no arms raised), pivotal for task prioritization and execution sequencing (Fig.6).

In essence, the vision node serves as the sensory hub of the robotic system, harnessing advanced computer vision techniques to perceive and comprehend its surroundings effectively. Through intricate object recognition algorithms, depth analysis, and body pose detection mechanisms, the system enhances its ability to interact intuitively with users, discerning their intentions, and executing tasks collaboratively and efficiently.

2.2.2 Head Tracking Node

The Head Tracking node is a key component in the robotic system, designed to ensure continuous visual engagement between the robot and the user by directing the robot's gaze toward the centroid of the individual. Upon initialization, the node sets up crucial parameters such as the ROS node name, camera frame, and topics for image acquisition and centroid tracking. It also establishes a point head action client, enabling precise control over the robot's head movements to maintain visual contact with the user.

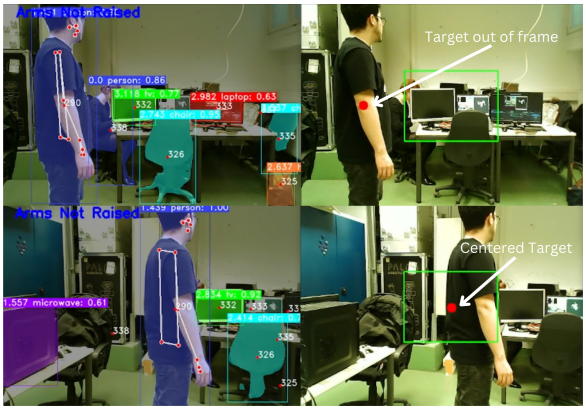

Fig. 7 Head tracking node following a human target.

Subscribing to the image topic, the node receives RGB image data and continuously tracks the centroid of the user person. As new image frames arrive, the node updates the displayed image to visualize the centroid's position and allows manual selection of a new centroid point through user interaction. The look to centroid method, a core component of the node, calculates the normalized coordinates of the centroid pixel, facilitating dynamic adjustment of the robot's head orientation to keep the user within its field of view Fig.7.

Throughout operation, the node remains responsive to system events, gracefully handling shutdown requests and maintaining an efficient processing loop. By encapsulating these functionalities within a modular framework, this node enhances the robot's human-robot interaction capabilities, enabling seamless communication and intuitive engagement with users across diverse environments.

2.3 Manipulation

For the manipulation tasks, particularly object grasping, our system leverages MoveIt! [12], an advanced motion planning framework as shown in Figure 8. This choice is driven by its robust capabilities in handling complex manipulator geometries and its integration with ROS [13, 14]. The initial step in our manipulation pipeline involves acquiring the target object's coordinates within the 'xtion_rgb_optical_frame' coordinate system. This is achieved through our vision subsystem, which precisely identifies and localises the object using depth and RGB data. Once the object's position is determined, we employ the 'tf' (transform) library to convert these coordinates into the 'base_footprint' coordinate system, which is essential for relating the object's location to the robot's reference frame [15].

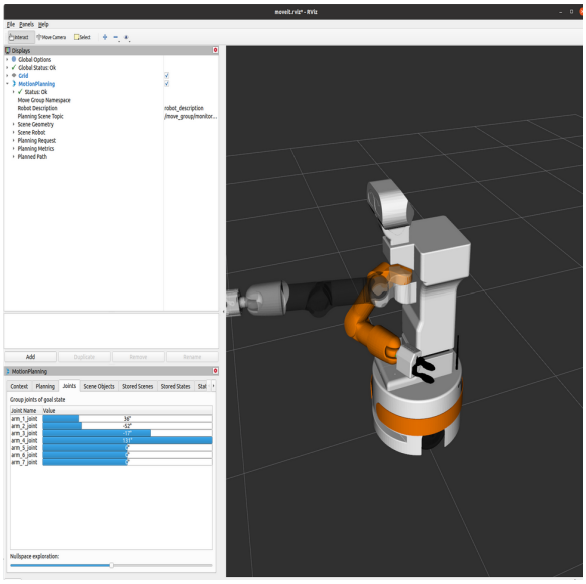

Fig. 8 TIAGo in MoveIt!

Having accurately translated the object's coordinates into the 'base_footprint' frame, we now possess the necessary XYZ positional data to proceed with the manipulation task. MoveIt! is then utilised to perform Cartesian space motion planning. This approach allows for direct manipulation paths to be calculated, taking into account the spatial relationship between the robot's end-effector and the target object [13]. By leveraging Cartesian planning, we ensure that the robot's movements are both efficient and optimised for the task at hand, significantly reducing the complexity of reaching and grasping operations. This integration of vision-based object localisation with MoveIt!'s motion planning capabilities forms manipulation strategy. It enables the robot to execute precise and reliable grasping actions, crucial for the successful completion of object interaction tasks within our robotic system.

2.4 Navigation

With the main goal for autonomous navigation within a competitive field towards an unspecified location, our methodology was focused around not using a pre-determined map by using simultaneous localisation and mapping (SLAM) principles. As expected, the utilisation of SLAM proved to be significantly more straightforward with a pre-existing map however, with the available sensors on TIAGo (2D LIDAR and Depth Camera) planning and navigation with a limited field of view was possible. Initial experiments were conducted using TIAGo's li-

Fig. 9 TIAGo Picking up Bag based on Moveit! instructions

braries in the Gazebo environment as illustrated in Figures 3, 4 and 10 which laid the groundwork for subsequent development on the physical platform. Included are results of the navigation system demonstration videos of the working Human_Follow `https://youtu.be/AmJvWMp4egE` and the Bag_Follow `https://youtu.be/DT3_9TTFRBQ`

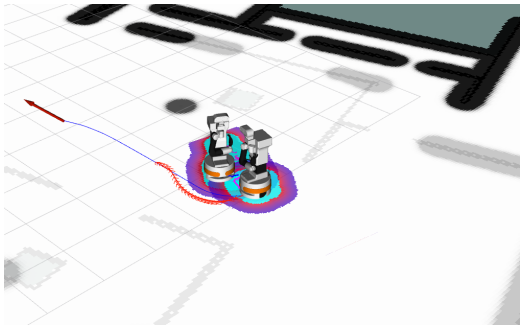

Fig. 10 Example Path Planning with obstacle avoidance in Gazebo environment with two TIAGo platforms

2.4.1 Navigation Stack

The navigation stack created for this task system ultimately used Gmapping [16] for visual odometry mapping, move_base [17] for cost map creation and ACML [18] for localisation as shown in the navigation stack diagram in Figure 11. The move_base and AMCL localisation nodes were run directly on the robot and, to prevent mapping issues, the map_creator node was killed before launching the navigation stack. As illustrated in Figure 11, the created

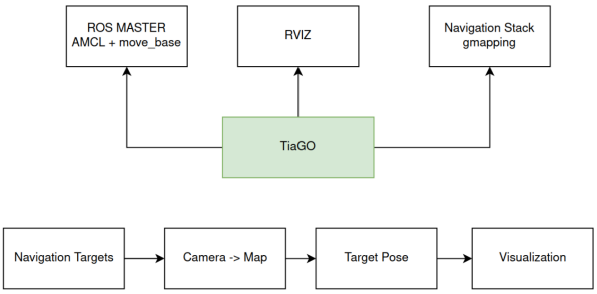

Fig. 11 Navigation Stack

navigation stack comprised of four stages: receiving the navigation targets from the perception stack (human_pose, bag_pose, the spheres in Figure 13), performing the transformation from camera to map frame, calculating the navigation target using the methodology described in section 2.4.3 and publishing a goal for the move_base (human_goal, bag_goal, the pose arrows in Figure 13). This process was complimented by a custom RVIZ [19] environment (Figure 12) as shown in Figures 13-17 with visualisations of the robot, control, localisation, mapping, planning and goals. Additionally, if the pose of the human was lost during running, the pose would automatically return the robot's current pose to prevent the robot attempting to navigate to an illegal pose. The system was also designed to use move_base's route planning to cancel and recalculate a path if an illegal path was detected. The re-routing was given a set number of opportunities to replan if the goal point was lost mid-navigation. While not fully implemented, the ability to return to the home position was planned with waypoint storage as a text file to record the inputs to move_base for the return to home functionality.

2.4.2 Mapping and Navigation

With the ultimate goal of using Macenski's Slam_Toolbox [20], there were substantial challenges in adapting TIAGo's pre-fabricated scripts without inducing critical issues, particularly concerning the definition of transforms. This was heightened by the fact that on the robot's startup, navigation and localisation nodes are automatically launched and functionality of the robot is dependent on these nodes. Moreover, the scarcity of comprehensive documentation, especially distinguishing between

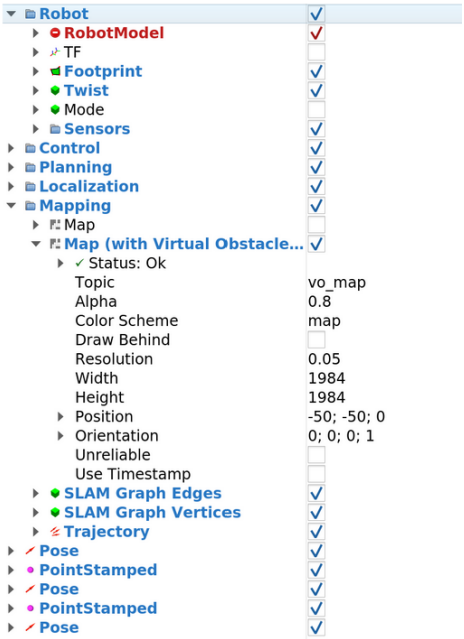

Fig. 12 Custom Developed RVIZ Environment which launches with Navigation Stack

ROS and ROS2 frameworks including configuration files, markedly impeded development.

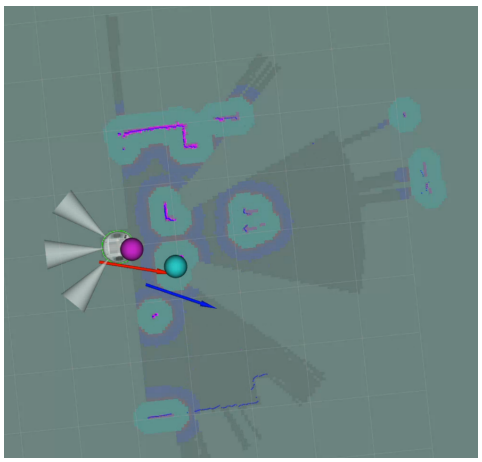

Fig. 13 Robot's Map on launch. Shown here are the robot's pose, the local and global costmaps, the poses of the bag (in frame) and the human goal (not in frame so at robot's pose). The human is clearly shown in the cost map as an obstacle.

2.4.3 Obstacle Avoidance

Incorporating obstacle avoidance into the path planning framework was a crucial step in our project to enhance autonomous navigation capabilities. Utilising both local and global costmaps as calculated by move_base, the the desired navigation pose could be calculated, enabling the robot to effectively navigate through the created map in real time. However, there needed to be a clear distinction between human and bag goals and obstacles. This differentiation was pivotal in navigating through environments populated with dynamic obstacles, without crashing into the objects and creating achievable goals for the path planning. By fine-tuning the offset distances for the goal "obstacles," we successfully minimised collision risks and facilitated adaptive planning around the costmaps. Given the task of incorporating obstacle avoidance into the path planning of a robot, the calculation of a goal point (x^*, y^*) that accounts for both the robot's radius and an object's radius was crucial. For this the original coordinates of the point be (x, y), the robot radius be r_{robot}, and the object radius be r_{object}. The total offset distance d needed to maintain a safe clearance from the object is the sum of the robot's and the object's radii, $d = r_{robot} + r_{object}$. The angle θ between the original point and the origin, necessary for adjusting the goal point, is calculated using the atan2 function:

$$\theta = \text{atan2}(y, x)$$

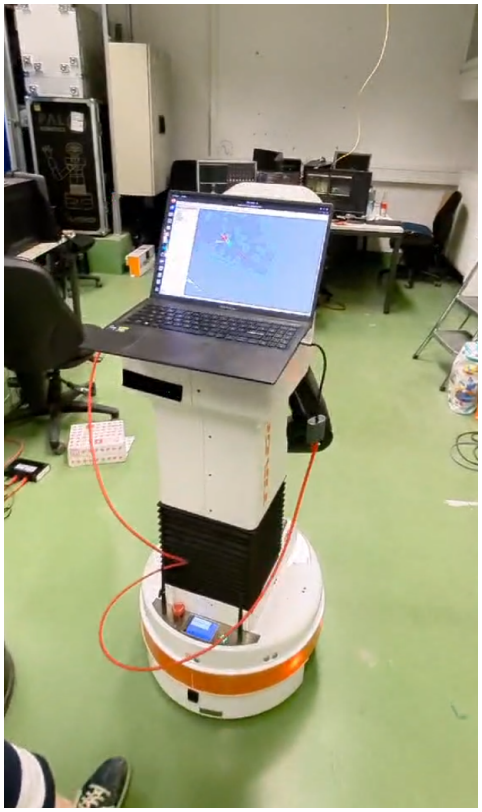

Fig. 14 TIAGo SLAMming

The adjusted coordinates (x^*, y^*) for the goal point, ensuring a safe distance from the obstacle, are derived as follows:

$$x^* = x - d \cdot \cos(\theta)$$

$$y^* = y + d \cdot \sin(\theta)$$

$$z^* = 0.0$$

where $d \cdot \cos(\theta)$ and $d \cdot \sin(\theta)$ respectively adjust the x and y coordinates to offset the goal point by distance d from the obstacle, considering the direction of θ. This methodology was used for both the calculation of goal points for the bag and the human navigation and guaranteed that the robot maintains an adequate buffer from obstacles during navigation, taking into account both its dimensions and those of potential obstructions. Future work includes incorporating visual obstacle detection to further refine navigation strategies, especially for identifying tables and freestanding obstacles that are not adequately represented in cost maps. Interestingly, we observed that the planning algorithms implemented were sufficiently robust to navigate around low-lying or small obstacles, indicating that our current focus could shift towards enhancing obstacle recognition and avoidance for larger and more complex obstacles.

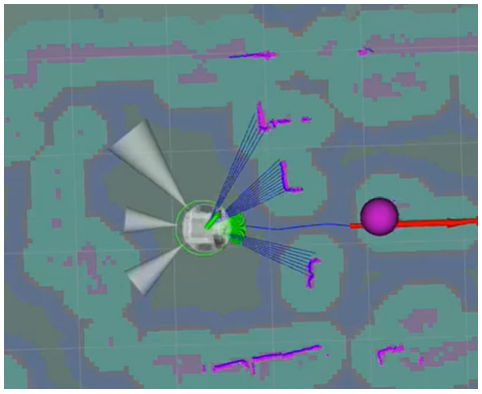

Fig. 15 Active Obstacle Avoidance during human_follow. The human_goal (red arrow) is located in front of the human's pose (pink sphere) and in a free space outside of the costmap buffer allowing for navigation. The obstacles in frame include small boxes, bags, tables, other humans and walls

2.5 Human Interaction

Human-robot interaction is a critical component of our robotic system, ensuring smooth operation and user engagement. We have divided this section into three main parts: arm gesture recognition, navigational interaction, and voice feedback, each designed to enhance the overall usability and functionality of our robot[21]. For the *Carry My Luggage* task, a script was created to outline how the human robot interaction should proceed, particularly introducing the task as shown in figure 16.

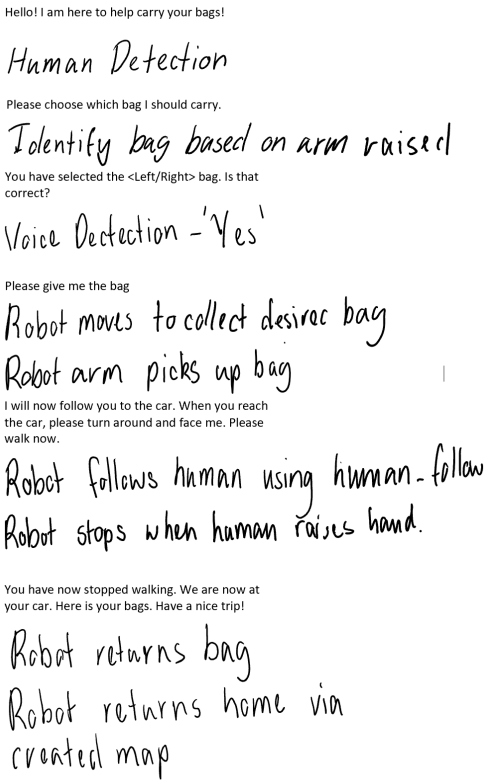

Fig. 16 Human Interaction Script for *Carry My Luggage Task*

2.5.1 Arm Gesture Recognition

The first layer of our interaction model utilises advanced arm gesture recognition to interpret human commands. This process begins with the visual identification of arm postures, leveraging previous developments in gesture recognition technology [21] [11]. Upon recognising a specific gesture, the system issues commands to determine the direction of action, effectively deciding which object—left or right—should be interacted with. This mechanism involves the precise control of head joints to direct the robot's gaze towards the intended target, enabling visual tracking of the object. This level of interaction allows the robot to interpret human intentions and execute tasks with higher accuracy and penalisation.

2.5.2 Navigational Interaction

Following successful manipulation, the robot's head returns to a neutral position, shifting its focus towards the human operator. Utilising its navigation capabilities, the robot then engages in person tracking. This is achieved through the initiation of a follow-me feature, where the robot maintains a visual lock on the person. To halt the robot's movement, the operator can signal by raising both hands, a universally recognised gesture for stop. Upon detecting this signal, the robot immediately ceases all navigational activities. This interaction method not only ensures safety but also provides a seamless way for users to control the robot's movement in a dynamic environment.

2.5.3 Voice Feedback

The final aspect of our interaction model encompasses voice feedback. By integrating a voice interface with the robot's system, we enable the machine to communicate using spoken language through speakers. This feature significantly enhances the robot's interactivity, making the human-robot interaction more intuitive and engaging. The voice system is tightly coupled with the state machine, providing users with audible feedback that reflects the robot's current operational state. This allows for more accurate command issuance and better understanding of the robot's actions, thereby improving the overall user experience.The integration is accomplished through the utilisation of the 'SimpleActionClient' from the ROS framework, specifically targeting the ''/tts'' (Text-to-Speech) action server [14]. This setup allows for dynamic speech synthesis based on the robot's state and interactions. Here is a brief example of how this integration is implemented in the context of our robot, which in this case is TIAGo:

```
# Initialize the SimpleActionClient
client = actionlib.SimpleActionClient('/tts', TtsAction)

# Wait for the server to start
client.wait_for_server()

# Create a goal to send to the server
goal = TtsGoal(text="Your message here.")

# Send the goal to the server and wait to be executed
client.send_goal_and_wait(goal)
```

3 Methodology

For each task undertaken, describe the task, the unique challenges it presents, and the strategies employed by your team. Highlight any innovative solutions or algorithms developed.

3.1 State Machine

Our robotic system employs a state machine to orchestrate task flows, ensuring a robust and adaptable response during human-robot interaction and object manipulation tasks. The state machine's design allows for handling of dynamic scenarios and unexpected events, which is critical in real-world applications.

3.1.1 Task States

The initial state, **INIT_POSITION**, prepares the robot by positioning it in a predefined starting location. Upon successful initialization, the state transitions to **Find_Human**, where the robot searches for and identifies a human operator. If a human is not found, the state machine proceeds to **Remind_People_to_come**, a state designed to attract human attention, possibly through auditory or visual cues. Once a human is detected, the **Arm_Dection** state is triggered, during which the robot observes and interprets the human's arm gestures. Successful gesture recognition leads the robot to the **Find_Bag** state, where it locates the bag intended for manipulation. Should the bag not be detected, the robot enters the **Bag_Helper** state, which involves seeking assistance, possibly through interactive queries or visual feedback to the human operator. Upon locating the bag, the robot transitions to **Move_to_Bag**, navigating towards it. In the event of navigation failure, the robot will re-attempt the process or seek assistance as necessary. Successful navigation to the bag leads to the **BagGrasp** state, where the robot attempts to grasp the bag. Following a successful grasp, the robot's gaze shifts to the human operator in the **Look_Human** state, preparing for the next interaction. The **Move_to_Human** state is subsequently activated, wherein the robot navigates back to the human. If this process is interrupted or aborted, the robot re-engages with the human in **Remind_People_to_come_2**, ensuring continuous engagement. Upon successful navigation to the human, the robot enters the final **PUT_DOWN** state, where it safely places the bag down at the target location.

3.1.2 Handling Aborted States

One of the key features of our state machine is the ability to handle **aborted** states. An **aborted** state occurs when an action fails to complete success-

fully, which can happen for various reasons such as Time_out when crawling or no person or target is recognized.

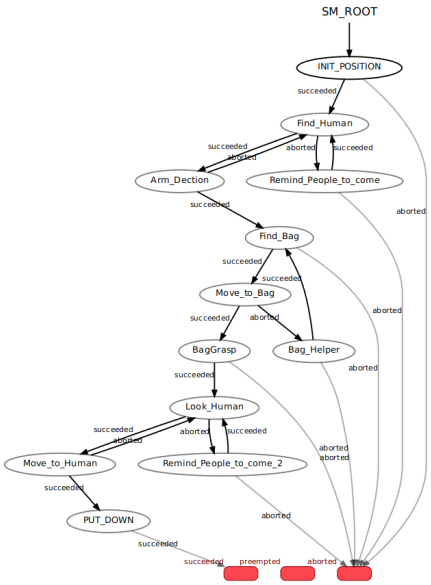

Fig. 17 Full State Machine for *Carry My Luggage* Task

4 Experiments and Results

For testing and the final demonstration, a test environment was created as illustrated in Figure 18. The boxes were placed under the bags to prevent collisions with the robot gripper and ground and provide a stable surface for the two bags used for the task. The chosen bags were used due to their dimensions and were stable and lightweight for manipulation. These boxes also acted as small obstacles for the Robot to avoid. For the final demonstration, the robot was able to complete all tasks however, issues where an audience were involved did make human_follow challenging. As indicated in 1.1 the project's goals as per the rules were defined as: person following, navigation in unmapped environments and social navigation. This project met these goals as all required goals were demonstrated in the demonstration. In terms of the RoboCup@Home Scoresheet in Figure 2, the following distribution of marks was achieved:

- Collecting the correct bag (100)
- Navigating to car (300)
- Avoiding Small Obstacles (50)
- Avoiding Other People (50)

Consequently, the final score for this challenge was 500 of a possible 700 marks.

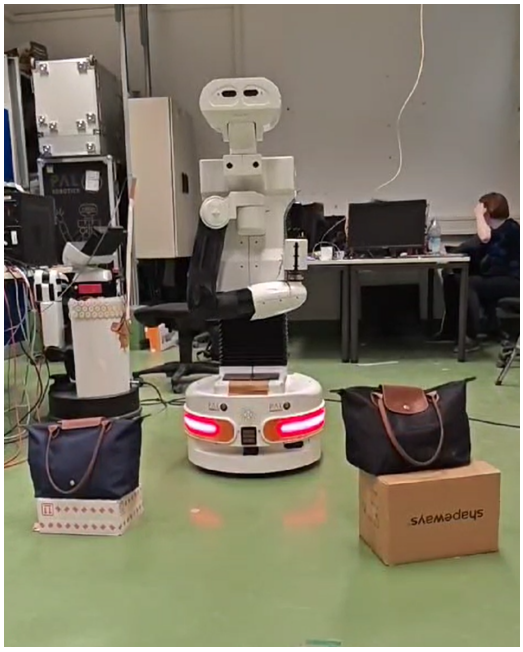

Fig. 18 Experiment Setup from Human's perspective

5 Conclusion and Future Work

The RoboCup@Home project discussed in this report outlines the process for creating the *Carry My Luggage* Task for the 2024 RoboCup@Home competition. It provides a complete overview of how to complete a stage in a team's RoboCup@Home competition using the TIAGo platform and, by using releatively known models including systems already deployed on the robot, can be used as a base solution for further research into navigation, vision systems and human interaction. Future work would include adding additional capabilities for the robot to complete other challenges in the RoboCup@Home competition to all for entry into future RoboCup seasons.

References

[1] J. Hart, M. Matamoros, A. Moriarty, H. Okada, M. Leonetti, A. Mitrevski, K. Pasternak, and F. Pimentel, "Robocup@home 2024: Rules and regulations." `https://athome.robocup.org/rules/2024_rulebook.pdf`, 2024.

[2] PAL Robotics, "TIAGo Robot." `https://pal-robotics.com/robots/tiago/`, Nov 2023. Accessed: 2024-02-14.

[3] C. T. Center, "2023 robocup@home - open platform - carry my luggage," Jul 2023.

[4] "Ubuntu 20.04 LTS." https://releases.ubuntu.com/20.04/. Accessed: 2024-02-14.

[5] "ROS Noetic." http://wiki.ros.org/noetic. Accessed: 2024-02-14.

[6] "Gazebo: Robot simulation made easy." http://gazebosim.org/. Accessed: 2024-02-14.

[7] "Ros wiki: Tiago tutorials." http://wiki.ros.org/Robots/TIAGo. Accessed: 2024-02-14.

[8] R. Fabre, B. Albar, C. Dussieux, L. Joffroy, Z. Li, C. Pinet, J. Simeon, and S. Loty, "Centre Aquitain des Technologies de l'Information et Electroniques (CATIE)." http://robotics.catie.fr/, 2023. Email: r.fabre@catie.fr; Accessed: 2024-02-14.

[9] D. Müller, N. Y. Wettengel, and D. Paulus, "Winning Team of the RoboCup Virtual @Home Open Platform League 2021." http://homer.uni-koblenz.de, http://agas.uni-koblenz.de, 2021. Active Vision Group, Institute for Computational Visualistics, University of Koblenz-Landau, 56070 Koblenz, Germany. Emails: {muellerd, niyawe, paulus}@uni-koblenz.de; Accessed: 2024-02-14.

[10] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," 2019.

[11] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," 2019.

[12] "MoveIt!." http://moveit.ros.org/. Accessed: 2024-02-14.

[13] S. Chitta, I. Sucan, and S. Cousins, "Moveit! [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2016.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[15] T. Foote, "tf: The transform library," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2007.

[16] "GMapping." http://wiki.ros.org/gmapping. Accessed: 2024-02-14.

[17] "move_base." http://wiki.ros.org/move_base. Accessed: 2024-02-14.

[18] "Adaptive Monte Carlo Localization (AMCL)." http://wiki.ros.org/amcl. Accessed: 2024-02-14.

[19] "RViz." http://wiki.ros.org/rviz. Accessed: 2024-02-14.

[20] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[21] S. S. Rautaray and A. Agrawal, "A review of gesture recognition techniques and applications," in *International Journal of Computer Applications*, vol. 975, p. 8887, Foundation of Computer Science, 2013.

[22] "play_motion." http://wiki.ros.org/play_motion. Accessed: 2024-02-14.