GestureCommand: A Simulated Camera-Based Gesture Recognition System for Autonomous Table-Specific Delivery Robotics

Chit Lee (2070900), Juni Katsu (2246810), Cheuk Yu Lam (2250728), Abbas Mandasorwala (2342319), Kozerenko Elizaveta (2395147)

***Abstract*—This document details the development of robotic simulation software, tailored to improve service in cafe environments. Central to this development are two features: gesture control and advanced navigation. The gesture control module enables cafe staff to command the robot to specific tables, and additionally, allows customers to signal the robot to return to the till using straightforward hand gestures. Meanwhile, the navigation feature ensures an accurate and efficient path finding of the robot from the till to the designated table and back. This combination aims to enhance operational efficiency, reduce service duration, and elevate the overall customer experience within cafe settings. This report includes technical parts of the development of gesture recognition algorithms and navigation strategies. It also addresses the various challenges encountered throughout the software's development phase. Furthermore, the report outlines potential enhancements to refine the software's functionality, aiming for optimal performance in real-world applications.**

***Index Terms*—Hand gesture control, gesture-based communication, path-finding algorithms, robotic navigation**

I. Introduction

IN the ever-evolving landscape of the service industry, technological innovations play a pivotal role in shaping customer experiences and operational efficiencies. This report introduces a visionary robot software, specifically designed for application in cafes, which incorporates two features: gesture control and navigation. The objective of this software is to streamline service processes, minimize waiting time, and provide unique and engaging customer interaction. Gesture control allows for a novel and user-friendly method of communication between the staff, customers, and the robot, where hand gestures are translated into specific delivery instructions. On the other hand, navigation equips the robot with the ability to steer autonomously and accurately within the cafe's environment, ensuring timely and precise delivery of orders. This report will outline the motivation behind this technological venture, the objectives of the software development, and the expected outcomes in terms of enhancing the cafe service experience. Following this, we will delve into the detailed methodology of the software's development, including the technical components, user interface design, and the integration of gesture control and navigation systems.

II. Related Work

Gesture recognition emerges as a pivotal communication mechanism in human-robot interaction. Extensive research has been conducted in this domain, followed by the work of H. K. Kaura et al. [1], who developed a camera vision-based gesture recognition system. This system displays hand-palm gestures utilizing the Convex Hull technique and finger counting to transmit commands. In contrast, K. N. Lavanya et al. [2] implemented a color-based recognition approach. This method employs gloves with distinct colours (red, green, and blue) to identify finger coordinates, using the spatial separation between these colours to activate specific commands. These methodologies underscore the variety of strategies in leveraging gestures for facilitating human-robot communication, with potential implications in robotics and interactive systems.

Regarding localization, the tracking of vehicle motion presents several methodological avenues, with odometer data being a notable option. Reference [3] provides a comprehensive evaluation of various tracking methods in comparison to odometer data. The Kalman filter emerges as a viable candidate for robot localization, particularly under the presumption of known initial robot positioning, as discussed in references [4] and [5]. Given that our simulation robot is equipped with a camera, localizing through camera data appears to be the most accurate method, capturing extensive environmental information, a concept further elucidated in [5].

The research gap identified pertains to the development of a gesture-controlled delivery robot. Numerous delivery robots have been designed and implemented, such as the coffee delivery robot in South Korea referenced in [6], which operates via QR code scanning, and another delivery robot described in [7], which utilizes a touchscreen for command input, the integration of gesture control in this context remains unexplored. Prior research, including [1] and [2], demonstrates the feasibility of gesture-based control in other applications. However, the application of such technology in the realm of delivery robots has not been documented, thus presenting an innovative research opportunity.

III. Problem Statement

This research focuses on three core hypotheses regarding the functionality of a sophisticated robotic system.

Accuracy of Gesture Recognition Mechanism: The foremost hypothesis posits that the robot will be capable of precisely recognizing and distinguishing a predefined set of human gestures. This assumption hinges on the clarity and distinctiveness of these gestures for the robot's vision system,

along with its ability to differentiate these gestures under expected environmental conditions in a cafe.

Obstacle Avoidance Capability: The second hypothesis centres on the robot's ability to navigate within a predefined area without incurring collisions. This skill will be facilitated by the robot's array of sensors, which are expected to provide accurate and consistent data regarding its immediate environment, including potential obstacles in its path. A critical aspect of this capability is the robot's proficiency in dynamically recalculating its route in response to environmental changes, such as an obstructed pathway due to the displacement of objects like a table.

Localization Accuracy: The final hypothesis involves the robot's use of its sensory apparatus, in conjunction with the Particle filter algorithm, for precise localization. The robot is anticipated to accurately determine its position within a mapped area and retain this capability even when manually relocated within the environment.

Experimental Approach: To validate these hypotheses, experimental trials involving human participants will be conducted. These participants will test the robot's gesture recognition system under expected conditions in a cafe. Subsequent testing phases will focus on the robot's navigation and obstacle avoidance within a known mapped area using a simulator, we created a map for a hypothetical cafe environment and tested our code there, followed by an integration of gesture recognition with navigational tasks, checking in the robot's ability to recognize a gesture, execute the corresponding action, and await subsequent instructions.

IV. Algorithms & Framework

This section presents a comprehensive overview of the algorithms and frameworks employed in developing an advanced robotics system, specifically focusing on gesture recognition and navigation capabilities. Utilizing ROS Noetic and Python 3.8 on Ubuntu 20.04, these systems are integral to our goal of creating an autonomous table-specific delivery robot, equipped to operate efficiently in service environments. The report explores the synergy between cutting-edge software and robotic functionality, emphasizing our system's robustness and adaptability in interactive service scenarios.

A. Gesture

The gesture control system is designed to interpret hand gestures using a camera feed and translate them into specific actions or commands. The system is built around a series of algorithms and techniques that allow it to detect hands, recognize gestures, and quantify these gestures in real-time. The main libraries used with Python are Mediapipe Machine Learning library developed by Google and OpenCV Open source Computer Vision library (for real-time hand detection). Below, the core algorithms and processes are explained in a generalized manner, including some pseudo code for clarity.

Hand Detection and Analysis

Initialization: A hand detection system is initialized with specific settings like the maximum number of hands to detect and sensitivity levels.

Processing Each Frame:

Each frame from the camera is converted to a format suitable for hand detection.

The system then scans the frame to identify any hands present. For each hand detected, it calculates the position of key points (landmarks) on the hand. It also identifies attributes like the bounding box of the hand, whether it's a left or right hand and its orientation.

```
for each frame from the camera:
    convert the frame to the required format
    detect hands in the frame
    for each hand detected:
        find landmarks
        calculate hand attributes
        (type, bounding box, orientation)
```

Finger State Determination

Algorithm:

The system checks the position of each finger. Special attention is given to the thumb, as its position interpretation varies based on the hand's orientation and type. The state (up or down) of each finger is determined based on the relative positions of specific landmarks.

```
for each hand detected:
    for each finger:
        determine if the finger is up or down
    special handling for thumb position
```

Gesture Recognition

Algorithm:

The system defines specific gestures based on which fingers are raised. It includes a timing mechanism to ensure the gesture is held for a certain period before being recognized. Once a gesture is recognized and confirmed, it triggers a specific action or command.

```
for each detected gesture:
    if a gesture is held for the required
    duration:
        confirm gesture
        trigger the corresponding action
```

Distance Measurement Between Points

Algorithm:

Select pairs of landmarks (e.g. fingertips) and measure the distance between them. This information can be used for advanced gesture recognition or other purposes.

```
for selected pairs of landmarks:
    calculate the distance between them
```

Main Loop

Process:

Continuously captures frames from the camera.

Applies hand detection on each frame.

Determines the number of fingers raised and recognizes gestures.

Triggers actions based on confirmed gestures.

Pass the command to the robot and send the goals.

Once arrive at the goal, the loop repeats.

Limitations and Improvements for gesture

Encountered limitations when initially using only one hand for finger detection, as incorporating both hands led to inaccurate and unreliable outcomes. During their trials, the detection system struggled to differentiate between a normally placed hand and a flipped one.

To enhance the results, the team decided to integrate CV Zone, a Computer Vision package, with Mediapipe and OpenCV. This combination led to more precise outcomes, enabling them to implement advanced features such as hand detection and finger counting on two hands. Additionally, this integration allowed for the detection of finger counts at various angles, including scenarios where hands are flipped. This approach significantly improved the robustness and versatility of their hand-detection system.

B. Navigation

The localisation and path-finding system is designed to navigate the robot to the corresponding table identified by the gesture control system. This system employs a particle filter algorithm for precise self-localization. Concurrently, a waypoint navigation system is integrated to enable the robot to efficiently traverse to various tables and subsequently return to the till.

Localisation

Accurate navigation to the designated table is a critical requirement for the robot, necessitating precise positional determination. In addressing this need, we have integrated a particle filter algorithm, recognized for its proven effectiveness in such applications [8]. Initially, the implementation of a Kalman filter was considered. However, given the availability of a particle filter code from a previous project, coupled with its good performance in past tests, the decision was made to utilize the particle filter approach.

For the parameters of the particle filter, a decision was made to adopt a low standard deviation of 1 for the initial particle distribution, since the robot's starting position is known. Furthermore, in consideration of the relatively compact nature of the operational environment, we have chosen a minimal standard deviation of 0.03, along with a particle count of 200, for the filter's subsequent update processes.

An explanation of the particle filter's operational mechanism follows.

```
initialization:
    generate a sample set N of particles from
    the initial state distribution
for each step loop:
    for each particle loop:
        prediction: Predict the new state of
        each particle based on the
        previous state.
        updating: Update each particle's
        weight based on the sensor data and
        complete normalization.
    resampling: generate a new resampled set
        with the replacement of particles based
        on their weights.
    estimation: Estimate the current robot's
        state by taking the average
        particles' weight.
```

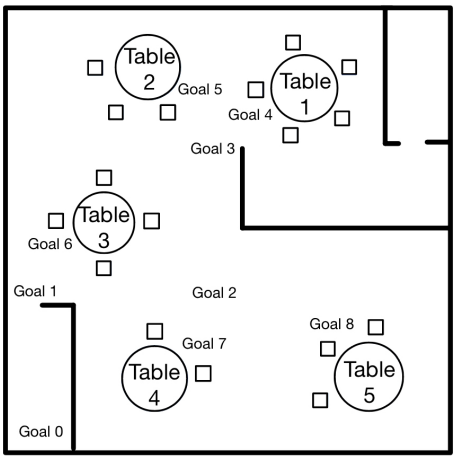

Fig. 1: The chosen cafe map to simulate, we have 5 tables, and Goal 0 is the Till area

For the estimated position, our approach involves calculating the average of all positions and orientations within the particle set. This method is particularly effective due to the anticipated small size of the particle cloud, which allows less concern about the presence of multiple particle clouds and ensures that the estimated pose (position and orientation) accurately reflects the robot's true physical location.

Navigation

For the navigational framework of our robotic system, a waypoint-based strategy has been employed. This method involves the strategic pre-definition of multiple waypoints on the map showing figure 1, thereby establishing designated routes for the robot's journey to any given table and its subsequent return to the till. The paths connecting these waypoints are linear, facilitating straightforward navigation from one waypoint to the next. A significant advantage of this approach is its relative simplicity, especially with the complexities associated with implementing a full navigation stack [9]. Typically, a comprehensive navigation stack comprises a global planner, a local planner, and both global and local cost maps, along with recovery behaviour algorithms. However, such an intricate system is rendered superfluous in our application due to the known map layout.

In our specific implementation, the global planner functions to keep track of the robot's current waypoint and to provide navigational instructions to the local planner about the next waypoint. Concurrently, the local planner's role is to assist the robot in progressing towards the designated waypoint goal while simultaneously facilitating obstacle avoidance when necessary.

The global planner

The global planner is tasked with orchestrating the robot's movement from one waypoint to the next. It maintains the list of waypoints, keeping track of the list of waypoints to and

from each table.

The functionality of the global planner extends to determining the appropriate next waypoint to the local planner. This decision-making process is facilitated by the implementation of an 'arrived' flag, which is initially set to a default state of false. As the local planner nears the target waypoint, it triggers the emission of a 'finished' message. This message is then published on a pre-specified topic. Upon receipt of this message, the global planner responds by changing the state of the 'arrived' flag to true. This alteration signals the completion of the current waypoint task and triggers the global planner to initiate the transmission of the subsequent waypoint to the local planner.

```
subscribe to the arrived callback
on_arrived():
    arrived_flag = true
for waypoint in waypoint_list:
    publish waypoint
    while not arrived_flag:s
        sleep 2ms
```

The local planner

After receipt of a designated waypoint from the global planner, the responsibility shifts to the local planner to navigate the robot to this target location. The local planner operates by first acquiring the estimated current position of the robot. It then calculates the angular difference between the current position and the goal way point. This calculation is crucial as it informs the robot of the required orientation to proceed directly towards the goal. To ensure efficient movement, the robot is programmed to make decisions based on this angular difference: it will turn left, turn right, or move forward if the angle discrepancy is sufficiently small.

A key aspect of this process involves normalizing the angle difference to fall within the range of -pi to pi. This normalization ensures that the robot always turns in the shortest possible direction towards the desired orientation. Without this normalization, the robot's movement could become highly inefficient. For instance, if the goal is oriented at 0 degrees and the robot is facing 350 degrees, without normalization, the robot would turn counter-clockwise through the entire 350-degree arc to align with the goal. With normalization, the robot would recognize that a minor 10-degree clockwise turn is more efficient. Furthermore, without normalization, overshooting the goal could lead to unnecessarily rotational corrections, as the robot might complete an entire 360-degree turn to realign with a slightly missed target.

```
target_angle = atan2(target_y - y, target_x - x)
angle_diff = target_angle - robot_angle
if absolute(angle_diff) < ANGLE_TOLERANCE:
    go forward
if angle_diff > 0:
    turn right
else:
    turn left
```

In the implemented code, an obstacle avoidance feature is also integrated, details of which are presented in the subsequent sections, and the robot will stop when it is sufficiently close to the target waypoint, so it won't go around non-stop finding the perfect spot.

Obstacle avoidance

Given the inherent uncertainty in our non-deterministic localization approach, there exists a possibility that the robot's perceived position may not align with its actual location, or it may navigate with less precision. Such discrepancies could lead to scenarios of unexpected collisions with walls or other physical obstacles. To mitigate this risk, we have developed a method enabling the robot to autonomously steer away from obstacles when proximity thresholds are threatened.

This method involves continuous monitoring of the space directly ahead of the robot to detect imminent collisions. Should the robot detect proximity to an obstacle within this frontal zone, it then proceeds to scan both its left and right flanks. The robot is programmed to pivot towards the direction offering greater open space, thereby navigating away from the obstacle and continuing its path.

```
Distance = min(laser scan of right, left, front)
if Distance < OBSTACLE_DISTANCE_THRESHOLD:
    avg_right = avg of right side scan readings
    avg_left = avg of left side scan readings
    if avg_right > avg_left:
        turn right and move forward
    else
        turn left and move forward
```

In addition to the obstacle avoidance manoeuvres, we have programmed the robot to move forward simultaneously. This adjustment addresses a specific issue observed in the initial strategy. Without forward movement, the robot, after avoiding an obstacle by turning, tends to re-orient itself towards the original goal. This action can potentially result in the robot being caught in a loop of continuous turning both ways, without making any actual progress towards the target. This progress ensures that when the robot subsequently adjusts its orientation back towards the initial goal, it has already moved forward a bit. Therefore, this integrated approach of turning and moving forward effectively prevents the robot from falling into an infinite loop of directional adjustments.

Limitations and Improvements for navigation

Several limitations in the current configuration of our robotic system need to be considered. A primary limitation is that the robot's movement is currently limited to direct routes from the till to a specific table and back. The absence of navigation between tables functionality restricts the robot's operational efficiency. For example, in scenarios where multiple customers from different tables place orders, the ability for the robot to sequentially visit multiple tables before returning to the till would significantly enhance service efficiency. Future research could focus on developing a path-finding algorithm that interconnects waypoints with linear paths, enabling the robot to navigate freely to any table from any location.

Another notable limitation of our system is that it is not operative to unseen maps, constrained by its reliance on predefined waypoints and a fixed map. This restricts the robot to direct, straight-line paths in known environments only, limiting its adaptability to new or altered settings. An advancement

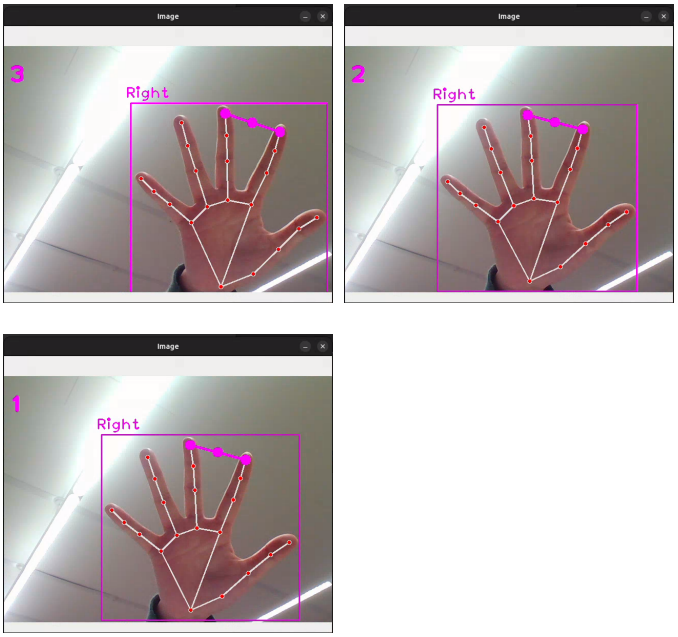

Fig. 2: Countdown for three seconds before confirming the command

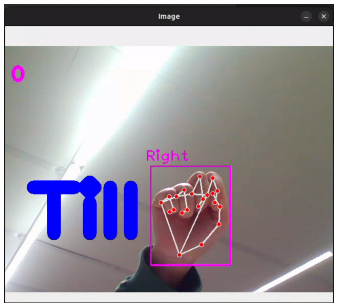

Fig. 3: Recognising TILL

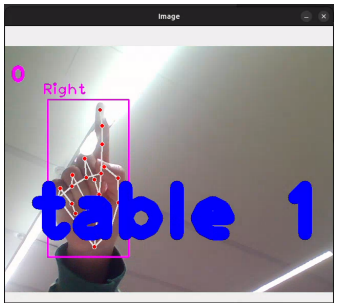

Fig. 4: Recognising TABLE 1

Fig. 5: Recognising TABLE 2

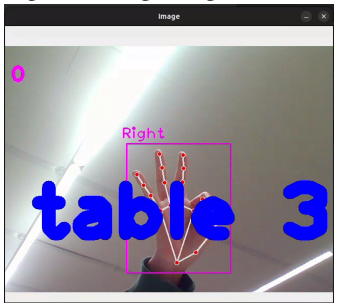

Fig. 6: Recognising TABLE 3

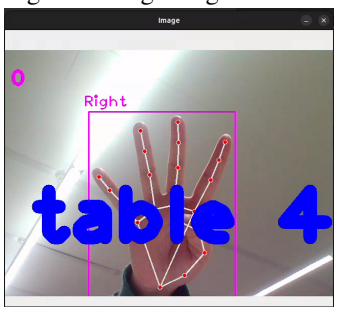

Fig. 7: Recognising TABLE 4

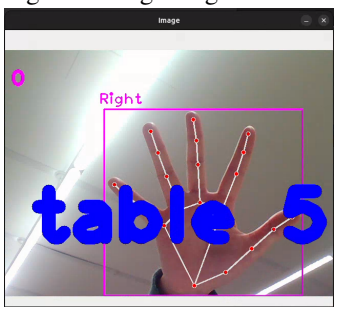

Fig. 8: Recognising TABLE 5

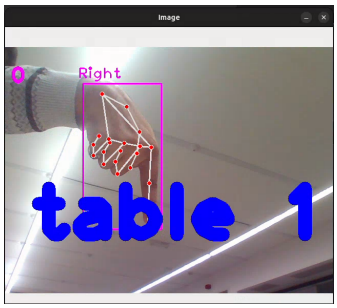

Fig. 9: Recognising flipped hand as TABLE 1

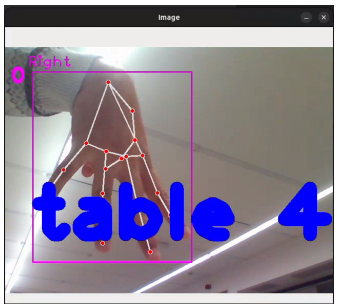

Fig. 10: Recognising flipped hand as TABLE 4

to overcome this limitation could be the introduction of an algorithm capable of automatically generating waypoints in unfamiliar environments. This would significantly enhance the robot's flexibility and operational effectiveness across diverse spatial landscapes.

Furthermore, the current system does not account for dynamic environmental changes, such as people moving across the robot's path or shifts in the placement of furniture. In real-world settings, these factors can significantly impact the robot's navigational effectiveness. Addressing this limitation would involve incorporating features that allow the robot to pause for moving obstacles and resume its course once the path is clear.

V. Experimental Results

Below are the results we got after running our code doing various experiments.

A. Gesture

For gesture recognition, the user must position their hand in front of the camera and maintain the specific gesture for three seconds. This duration ensures that the system accurately recognizes the gesture, as depicted in figure 2.

The system is designed to recognize five distinct gestures, each corresponding to a specific hand position. When the hand is clenched in a fist, it is identified as 'Till'. A hand forming the number 1 is recognized as 'Table 1'. Displaying the number 2 with the hand signifies 'Table 2'. The gesture for the number 3, where the hand shows three fingers, is identified as 'Table 3'. When the hand presents the number 4, it is recognized as 'Table 4'. Finally, a hand gesture showing the number 5 is recognized as 'Table 5'. The demonstration of these are shown in figure 3-8. Each of these gestures corresponds to a unique command within the system.

Moreover, the system is also capable of detecting inverted hand gestures. This feature enhances the versatility of the system, allowing for a broader range of gesture recognition. For instance:

As shown in figure 9, when the hand is positioned upside down and forms a specific gesture, the system can accurately identify and interpret this variation.

This functionality is particularly useful in scenarios where

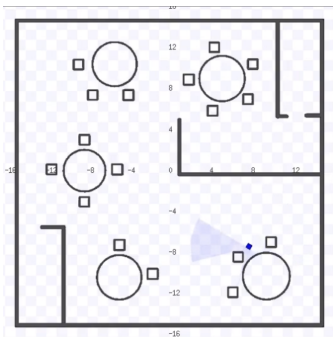

Fig. 11: 1. too close to the wall

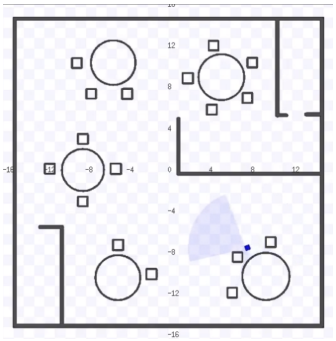

Fig. 12: 2. starts turning right

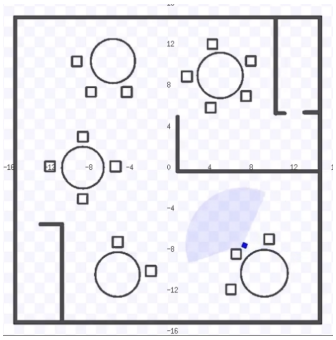

Fig. 13: 3. successfully escaped the situation

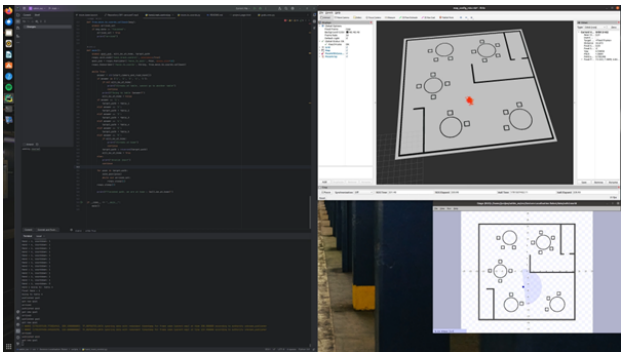

Fig. 14: The particle cloud is very scattered

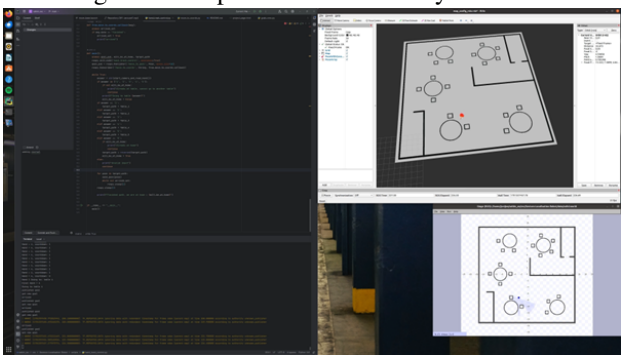

Fig. 15: The particle cloud narrows down

the user's hand orientation may vary due to ergonomic or situational constraints.

B. Navigation

In the navigation component, the ROS simulator 'stage_ros' is utilized for testing. This simulator requires a 'world file', which defines the characteristics of our simulation, such as the simulation map and robot specifications. Since accurate localization is crucial, The 'map_server' is also initialised with our designated map. To visualize the simulation, including the particle cloud from the particle filter, the estimated pose, and the map, 'rviz' is employed.

In the testing stage, various nodes will be running: 'node.py' for the particle filter, 'move_to_coords.py' for the local planner, 'hand_track_control.py' which combines the functions of the global planner and gesture recogniser and use 'set_initial_pose.py' to set the initial pose automatically. These functions can be tested by running the launch file, 'everything.launch', to streamline the testing process. The complete code and instructions are accessible on our GitHub repository.

By testing, we have observed the robot is capable of moving to and from every table. A video of it being demonstrated can be seen on the project webpage. The robot receives a gesture signal at the till, then proceeds to Table 4 via predefined waypoints. Upon reaching its destination, it pauses for another gesture command before successfully returning to the till.

Throughout the journey, when the robot enters an empty territory, the estimates become less accurate as evident in figure 14. However, the deviation in estimates is not significant and as the robot gets close to physical objects, the particle cloud condenses, as shown in figure 15, indicating a return to precise localization.

Furthermore, the robot's obstacle avoidance capabilities are validated, as seen in figures 11-13. In these instances, the robot, detecting tolerance to a wall, executed a sharp right turn, effectively avoiding a collision and confirming its ability to navigate around obstacles.

VI. Conclusion

The development of a gesture-controlled delivery robot as outlined in this report represents an innovative try of service automation, specifically in the cafe environment. Our project has successfully integrated two pivotal features: gesture control and navigation, to create a highly intuitive and efficient system for beverage delivery in cafes. The gesture control system, leveraging advanced libraries such as Mediapipe and OpenCV, enables staff and customers to interact with the robot through simple hand gestures. This not only adds an element of interactivity but also streamlines the ordering process.

The navigation component, utilizing a particle filter for self-localization and a waypoint navigation system, ensures the robot's precise and efficient navigation within the cafe. These technologies, together, contribute to the robot's ability to deliver orders accurately and promptly, enhancing the overall customer experience.

Throughout the development process, the project faced several challenges, particularly in gesture recognition accuracy and the robot's ability to navigate and avoid obstacles. Through iterative testing and enhancements, including the integration of the CV zone package, these issues were substantially addressed, leading to a more robust and reliable system.

The experimental results demonstrate the system's capability to accurately recognise gestures, navigate to designated locations, and avoid obstacles. This success is a testament to the potential of such technology in revolutionizing service delivery in various environments.

Some future work that can be done includes further improving the gesture detection to detect more than one hand, allowing for more commands to be issued. There can also be improvements on the navigation side, like delivering to multiple tables in each run, anticipating people walking in front of it, and implementing more advanced ways to generate waypoints, and path find between them.

In conclusion, this project fills the research gap of a gesture-controlled delivery robot. Showcasing the feasibility and effectiveness of combining gesture control with navigation techniques in service robots. As we look to the future, the integration of such technologies could play a crucial role in further enhancing customer service experiences, not only in cafes but in a wide range of service industries.

REFERENCES

[1] H. K. Kaura, V. Honrao, S. Patil, and P. Shetty, "Gesture controlled robot using image processing," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 5, 2013. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ff0f20e3dbbdf257ec3ca36be4ed251036b49e11

[2] K. N. Lavanya, D. R. Shree, B. R. Nischitha, T. Asha, and C. Gururaj, "Gesture controlled robot," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017, pp. 465–469.

[3] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–6.

[4] G. F. Welch, *Kalman Filter*. Cham: Springer International Publishing, 2020, pp. 1–3. [Online]. Available: https://doi.org/10.1007/978-3-030-03243-2_716-1

[5] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[6] W. C. Portal. Coffee delivery robot piloted at south korea's largest airport. [Online]. Available: https://www.worldcoffeeportal.com/Latest/News/2022/July/Coffee-delivery-robot-piloted-at-Korea-s-largest-a

[7] R. Cairns. These cute robots could deliver your next coffee. [Online]. Available: https://edition.cnn.com/travel/article/rice-robotics-hong-kong-hospitality-hnk-spc-intl/index.html

[8] P. D. Moral, "Measure-valued processes and interacting particle systems. Application to nonlinear filtering problems," *The Annals of Applied Probability*, vol. 8, no. 2, pp. 438 – 495, 1998. [Online]. Available: https://doi.org/10.1214/aoap/1028903535

[9] E. Marder-Eppstein. navigation - ROS Wiki. [Online]. Available: https://wiki.ros.org/navigation