A Comprehensive Study and Analysis of Artificial Intelligence-based Waiter Robot in Restaurant

by

Allama Bakhtiyar Nafis
18201085
Kawsar Ahammed
19101126
Humaira Rahman Oishi
19101391
Sumya Afroj
19301164
Rezwana Chaudhury Raka
19101128

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

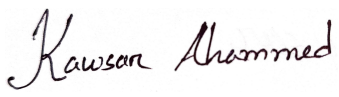
Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

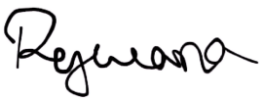
Allama Bakhtiyar Nafis
18201085



Kawsar Ahammed
19101126

Humaira Rahman Oishi
19101391

Sumya Afroj
19301164



Rezwana Chaudhury Raka
19101128

Approval

The thesis titled "A Comprehensive Study and Analysis of Artificial Intelligence-based Waiter Robot in Restaurant" submitted by

1. Allama Bakhtiyar Nafis (18201085)
2. Kawsar Ahammed (19101126)
3. Humaira Rahman Oishi (19101391)
4. Sumya Afroj (19301164)
5. Rezwana Chaudhury Raka(19101128)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 18, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Khalilur Rahman
Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md.Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

The rapid development of technology has led to the implementation of numerous solutions aimed at streamlining processes, one of which is the incorporation of artificial intelligence. The Simultaneous Localization and Mapping (SLAM) algorithm is fundamental to the restaurant robots operation and thereby determines its success or failure in carrying out its tasks. Few studies have looked at how well SLAM algorithm work when combined with path planning for indoor location, even though the present two-dimensional Lidar-based SLAM algorithm has done quite well, especially in indoor scenarios. Planning and mapping routes for restaurant robots operating in an indoor setting is the topic of the following essay. The goal of this research is to find out how indoor location systems may make use of path planning algorithms in conjunction with SLAM methods. To verify the mapping data, real-time path planning must be investigated. For global path planning, the A* algorithm is used to find the most efficient route while avoiding obstacles. Local path planning makes use of the Dynamic window approach (DWA) algorithm. After extensive testing in simulated, emulated, and competitive indoor situations, it was determined that both the SLAM and path planning algorithms performed admirably. Further, we employ a speech recognition component to facilitate communication with clients and an object identification model to track down lost items. Finally, experts may find this papers results useful when deciding which algorithms to use when building SLAM systems that meet their specific needs.

Keywords: Robot Operating System (ROS), Simultaneous Localization and Mapping (SLAM), LiDAR, Python, Navigation, Object detection, SpeechRecognition , Automation.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis has been completed without any major interruption. By the grace of Allah, we were able to put our best efforts and successfully complete it on time.
Secondly, we thank our supervisor Dr. Md. Khalilur Rahman for giving us this opportunity to research on this topic and assisting us through the process. We would like to express our special gratitude and appreciation to him for guiding us throughout the whole phase and giving us such attention and time.

Table of Contents

Declaration i

Approval ii

Abstract iii

Dedication iv

Acknowledgment iv

Table of Contents v

Nomenclature viii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Research Problem . . . 2
- 1.3 Research Objective . . . 3
- 1.4 Problem Statement . . . 4
- 1.5 Scope and Limitations . . . 4
- 1.6 Thesis Outline . . . 5

2 Literature Review 6
- 2.1 Mapping . . . 6
  - 2.1.1 GMapping . . . 6
  - 2.1.2 Hector-Slam . . . 7
  - 2.1.3 Cartographer . . . 8
- 2.2 Localization . . . 8
  - 2.2.1 Kalman Filter Algorithm . . . 8
  - 2.2.2 Particle Filter Algorithm . . . 9
- 2.3 Path Planning . . . 9
  - 2.3.1 Local Path Planning . . . 10
  - 2.3.2 Global Path Planning . . . 11
- 2.4 Object Recognition . . . 11
  - 2.4.1 SSD_MobileNet_V3 . . . 12
  - 2.4.2 Faster R-CNN . . . 12
- 2.5 SpeechRecognition . . . 13
  - 2.5.1 Natural Language . . . 13
- 2.6 The Main Menu Language . . . 14

2.6.1 The Alternate Menu Language . . . 14
2.7 Machine learning . . . 14
2.8 Neural Networks . . . 15
2.9 CNNs . . . 15
2.10 YOLO . . . 15
2.11 NN-based object detection . . . 16
2.12 Related Work . . . 16

3 Methodology, Requirement Analysis, Implementation 20
3.1 Platform, tools and asset selection for Navigation . . . 21
3.1.1 ROS Noetic . . . 21
3.1.2 Ubuntu Linux 20.04 . . . 21
3.1.3 Gazebo Simulator . . . 21
3.1.4 Testing of Simultaneous Localisation and Mapping Algorithm 22
3.1.5 Rviz . . . 23
3.1.6 Object Recognition (Faster R-CNN) . . . 23
3.1.7 Data Collection . . . 23
3.2 Annotation and Labeling . . . 23
3.2.1 Model Selection and Training . . . 23
3.2.2 Testing and Results Analysis . . . 24
3.2.3 Speech Recognition . . . 24
3.2.4 Initialization and Setup . . . 24
3.2.5 Functions Definition . . . 24
3.2.6 Speech Recognition . . . 24
3.2.7 Intent Matching . . . 24
3.2.8 Response Generation . . . 25
3.2.9 Main Loop . . . 25
3.3 Implementation . . . 25
3.3.1 Simultaneous localization and mapping (SLAM) . . . 25
3.3.2 Localization . . . 26
3.3.3 Path Planning . . . 27
3.3.4 Object Recognition . . . 28
3.3.5 SpeechRecognition . . . 33

4 Result Analysis 37
4.1 Mappping . . . 37
4.2 Path Planning . . . 40
4.3 Object Detection and Recognition . . . 42
4.3.1 Faster-R-CNN comparison and SSD MobileNet V3 . . . 42
4.4 SpeechRecognition . . . 42
4.5 Research Challenges . . . 47

5 Conclusion 48
5.1 Future Works . . . 49

References 50

List of Figures

1.1 Line Following Robot [6] 2
1.2 SLAM algorithm based Robot 2

2.1 Gmapping 7
2.2 Path Planning [10] 10
2.3 Network structure of SSD-MobileNetV3 [30] 12
2.4 Faster R-CNN [13] 13

3.1 Autonomous Restaurant Robot Diagram 20
3.2 Simultaneous Localization and Mapping. 22
3.3 Gazebo Indoor Image 25
3.4 Hector Slam 26
3.5 Localization 27
3.6 Path planning 28
3.7 The bot reached to the target node 28
3.8 Object recognition code 29
3.9 Object recognition using FasterR-CNN 30
3.10 Better accuracy using Faster-R-CNN 30
3.11 Object Recognition using ssd mobilenet v3 32
3.12 use more objects for performance test 33
3.13 Speech Recognition code 34
3.14 Main Conversation Dataset 35
3.15 Jokes Dataset 35
3.16 Birthday Greeting Dataset 36

4.1 Mapping comparison 38
4.2 Gmapping memory use 39
4.3 Hector slam memory use 40
4.4 Comparison between A* Algorithm and Dijkstra Algorithm 41
4.5 Google Speech API 43
4.6 CMUSphinx 43
4.7 Final results of three different API [53] 45
4.8 The Comparison Result [53] 46

Nomenclature

The following list describes several symbols and abbreviations that will be later used within the body of the document:

AGB Automated Guided Vehicle

AMI Application Programming Interface

AMCL Adaptive Monte Carlo Localization

APIs Application Programming Interfaces

ASR Automatic Speech Recognition

DWA Dynamic Window Approach

Faster R-CNN Faster Region-Based Convolutional Neural Network

GUI Graphical User Interface

GMapping Grid Mapping

Hector SLAM Hector Simultaneous Localization and Mapping

IMU Inertial Measurement Unit

IoT Internet of Things

LIDAR Light Detection and Ranging

RBPF Rao-Blackwellized Particle Filter Algorithm

ROS Robot Operating System

ROI Region of Interest

RPN Reverse Polish Notation

SIoT Social Internet of Things

SLAM Simultaneous Localization and Mapping

WER Word Error Rate

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Background

4Ds, Specialists in the field of robotics concur that autonomous mobile robots and manipulators are designed to undertake activities that pose a risk, involve monotonous repetition, or are tiresome for individuals. A conventional method for classifying these types of work is through the utilization of the 4 Ds: Dull, Dirty, Dangerous, and Dear. A specialized type of robotic system tailored for operations within a restaurant setting is referred to as an automated restaurant robot. These functions encompass order taking, food and beverage delivery, as well as cleaning tasks. The primary goal of integrating these robots into restaurant settings is to streamline processes, boost productivity, and enhance the overall dining experience for customers. The utilization of automated restaurant robots proves advantageous for eateries, leading to increased operational efficiency, reduced labor costs, and a more personalized and efficient service delivery to patrons [1]. Automated restaurant robots are diverse and are designed to undertake specific tasks, including order processing, food and beverage delivery, and various cleaning responsibilities within kitchen spaces. The incorporation of automated restaurant robots yields several benefits for establishments, including accelerated service delivery, decreased reliance on human labor, and heightened overall efficiency. Additionally, these robots can tackle tasks deemed challenging or hazardous for humans, such as kitchen activities like cooking and cleaning. Furthermore, they contribute to a distinctive and interactive dining experience for customers [2]. These robots utilize a blend of cutting-edge technology like natural language processing, computer vision, robotics, machine learning, and control systems to carry out their tasks [3]. Key technological requisites for automated restaurant robots involve the ability to navigate and interact with their surroundings, comprehend customer orders, and execute tasks like transporting dishes and utensils. Machine learning algorithms play a crucial role in enhancing the robots performance over time. Training these algorithms with simulated environments and test data aids in refining the robots capabilities. Integration with other systems, such as Point of Sale, Inventory Management, and Kitchen Display Systems, further enhances operational efficiency [4]. The design and implementation of automated restaurant robots necessitate in-depth comprehension of various technologies and rigorous testing to ensure their effective and safe task execution. This involves considering 4D robotics facilities for comprehensive understanding and optimization of the robotic system [5].

1.2 Research Problem

The problem statement highlights the most difficult parts of making an automatic robot for a restaurant. This shows the importance of the robot to be able to easily move through complex environments while focusing on providing quick and excellent service in order to cut down on labor costs. The first thing we thought of was a way to solve the problem of finding our way around inside without following the line.

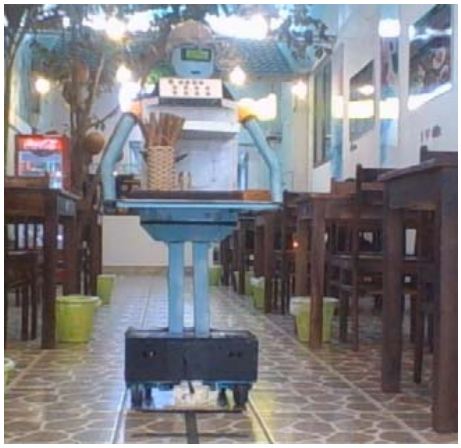


Figure 1.1: Line Following Robot [6]

Line-following robots in Fig. 1.1 follows pre-made paths. They can only follow a certain path and may struggle in changing situations or with unanticipated impediments. Devices let line followers determine how different the line is from its surroundings. Lighting and line quality can affect the robots success. Line-following robots need clean paths to maneuver. They can not navigate unfamiliar places [6]. The majority of line followers cannot map their surroundings. It hinders their understanding and memory of spatial assembly.

Figure 1.2: SLAM algorithm based Robot

A Simultaneous Localization and Mapping (SLAM) algorithm based robot in Fig. 1.2 has shown that it can move around without using lines on the ground. The SLAM technology is effective in enabling robots to move across uneven surfaces, adapt to new events, and find new paths, demonstrating their ability to see and avoid obstacles in real-time [7].
SLAM allows robots to move without pre-programmed paths, making it useful in outdoor environments. It allows for accurate location tracking, making it more precise than line-following methods. SLAM can be used for research, search and rescue, and self-driving cars, offering an alternative way to navigate [8].
Researchers are developing an AI-driven transport robot that can navigate busy cities without incident. They are focusing on overcoming issues, creating powerful perception systems, and building decision-making systems. The project aims to ensure safe and useful transportation jobs, allowing people and robots to work together on tasks like delivering goods and communicating [9].
We want the robots hardware and software to last. Keep solid maintenance plans so we can fix technological issues and failures quickly. Another crucial step is following robot and self-driving car laws including safety and responsibility issues. To utilize technology morally and legally, we must observe all laws and guidelines. Yet, this issue shows how important it is to protect workers and customers and have a robot that can adapt to different environments [10].
The team developed a speech recognition system and object identification model for a restaurant robot. The goal is to improve food service, customer experience, and human worker jobs. The robot can move independently, handle maintenance, and adapt to changes in its environment. Researchers must overcome challenges in creating an AI-driven transport robot, such as navigating busy cities and developing powerful perception systems to distinguish people, cars, and parcels [10].

1.3 Research Objective

The main goal of the research is to transform the hospitality sector by addressing and resolving issues related to using human wait staff in establishments like hotels and businesses. It identifies common issues like forgetting orders and delays in service. In addition, the conflicts are made worse by the time it takes to wait for customers to place their orders and the fact that given items have to be carefully remembered by hand [11]. Because now a days teens and young adults value flexibility and free time, there is a chance to look into using robot waiters as a revolutionary answer in the restaurant business [12] . To solve the problems that come with having human wait staff in hotel settings, putting advanced technologies, Faster R-CNN, into robots could be a good idea [13].

The study explores advantages of robot waiters, such as increased working hours and heavier loads. The research proposes solutions using advanced technologies like Faster R-CNN for object identification and intelligent features like SLAM for navigation and Speech Recognition for effective interaction.

1. The Simultaneous Localization and Mapping (SLAM) algorithm is greatly helpful in our model for creating accurate maps with information about obstacles. Utilizing these maps, the robot can accurately pinpoint its location, allowing it to find the

best way to get to its intended locations.

2. Faster-R-CNN, which stands for Faster Region-based Convolutional Neural Network, is used in our model, which makes the robot much more efficient. In particular, it helps find food and makes it easier to find lost items that belong to customers, which is useful for situations where customers leave things behind by accident.

3. Our model also has Speech Recognition technology, which is a notable feature. This feature improves the robot's capacity to engage with consumers in a more effective and optimized way, hence enhancing the entire eating experience.

1.4 Problem Statement

Robots in restaurants can revolutionize the industry by enhancing efficiency, reducing costs, and improving customer satisfaction. They offer fast, personalized service while minimizing contact with food, which is particularly valued for cleanliness and safety concerns post-outbreaks. Making restaurant robots that use less trash and energy is good for the environment [2]. These technological advancements require ongoing maintenance, updates, and repairs to ensure longevity and optimal performance. In Bangladesh, where labor shortages can affect service quality and prices, implementing robots in fast food and casual dining establishments can address these challenges effectively. When comparing restaurant robots, a comparison between line-following and SLAM-based models highlights clear differences in their operational methods [14]. The comparison between line-following and SLAM-based robots reveals distinct operational methods, with SLAM-based models offering greater adaptability and autonomy, making them more suitable for dynamic restaurant environments. Embracing restaurant robots not only enhances operational efficiency but also aligns with sustainability goals by potentially reducing waste and energy consumption. Overall, integrating robots into restaurant settings presents significant opportunities for improving both business operations and customer experiences. Although line-following robots are effective for regular work in regulated areas, SLAM-based robots provide more adaptability and autonomy, rendering them well-suited for dynamic restaurant settings where the arrangement may vary [15].

1.5 Scope and Limitations

Although ROS SLAM is an effective instrument for mapping and navigating environments, it is important to acknowledge its limits when using it in a restaurant robot or similar scenarios. Quality of initial map influences SLAM accuracy. Poor sensor data or unknown features may hinder robot navigation. Device kind and quality affect SLAM performance. Electronics, obstacles, and reflections influence sensors. Flexible restaurant layouts, waitstaff, and lighting. Dynamic environments might create SLAM map problems. Big, sophisticated settings can make SLAM computationally demanding. Resources may be limited for restaurant robots. Processing overload might delay real-time navigation [16]. A restaurant robot can do a lot more and connect with people better if it has object recognition and speech recognition

built-in [17], [18]. But there are some limitations and challenges also. Restaurant lights impair navigation. Uneven illumination, glare, and shadows impede classification and detection. Robots need real-time object recognition for sensitivity [19]. Robots may have trouble seeing hidden objects. Restaurants change food often. Object identification technology is updated constantly to reflect these changes. Quality and variety of training data affect object recognition model performance. Restaurants have background chatter, clinking dishes, etc. Background noise impairs voice detection. Dialects and accents complicate speech recognition. Training linguistically diverse models is difficult. Crowded restaurants can host simultaneous discussions. Many voices may be difficult for voice recognition. Restaurant-specific language can confuse speech recognition. Voice recognition records and processes sound, raising privacy concerns. Speech recognition failures can slow robots and upset users. To work around these problems, we pick SLAM, object detection and speech recognition algorithms [20], [21] that are strong and flexible, keeping models up to date [22]. Context awareness and designing ways for humans and robots to communicate can also help restaurant robots do their jobs better and make people happier.

1.6 Thesis Outline

The remaining portion of the paper has been structured in the following manner. Chapter 2 consists of a literature review that examines relevant and existing methodologies connected to the planned research. It provides a comprehensive description of the works that are already well-known in the field. Chapter 3 focuses on the explanation of the methodology. We presented our proposal and detailed all the procedures employed in our research. Chapter 3.3 clarifies the implementations that are employed in this research. The text provides information regarding the process of constructing the simulator using the concept of SLAM. Furthermore, it covers the execution of Object Recognition and also SpeechRecognition. The section containing the production of data and the analysis of results can be found in Chapter 4. This section provides comprehensive information regarding the structure of the methodology employed to get the result. Additionally, it encompasses a comparative analysis of the obtained data. It provides an overview of the result analysis and discusses the problems that were successfully solved. Finally, chapter 5 provides a comprehensive summary of the research conducted in the study, including a discussion on future work about potential extensions or implementations of the findings.

Chapter 2

Literature Review

We have discussed the current research papers and achievements in the literature review and associated studies. This literature review explores restaurant robotics advancements, focusing on sensor technology, localization, mapping, and navigation algorithms. It examines robot-human interaction, user satisfaction, and potential obstacles. The review also examines the impact of AI, machine learning, and sensor technologies on autonomous restaurant robots. It evaluates their design, performance, pros and cons, challenges, and remedies. It also addresses legal and regulatory issues, emphasizing safety and compliance. Future research considers existing knowledge and technological restrictions to ensure the feasibility and implementation of new ideas.

2.1 Mapping

In the world of Robotic Operating System (ROS), mapping is the process of making a picture of the world around a robot. The robot can understand its surroundings and find its way around different settings with the help of this representation, which is often called a map. Mapping is an important part of autonomous robotic systems because it helps robots make smart decisions about where to go and how to connect with their surroundings. The mapping system in ROS is a key part of making it possible for robots to understand and move around in their own environments. The method includes processing data from sensors to make a picture of the surroundings. Then, this image is used to make smart choices about how the robot moves and interacts with other things [23].

2.1.1 GMapping

Grid mapping is accomplished by the SLAM with a laser technique known as grid mapping (GMapping). Presumably, this is the most often used SLAM algorithm. OpenSLAM.org provides an implementation of the standard algorithm on the PR2, a widely used platform for manipulating mobile devices. The core idea of the technique, which was first proposed in, is to guess the function that modifies the state by using Rao-Blackwellized particle filters (RBPFs). Another term for this technique is the RBPF SLAM algorithm. The Rao-Blackwellized particle filters that are employed are the source of its name. Two key modifications that significantly improved

the method for practical use were the optimization of the proposal distributions and the addition of adaptive resampling. Then, because grid maps are used, it is termed GMapping, which is G for grid [24].

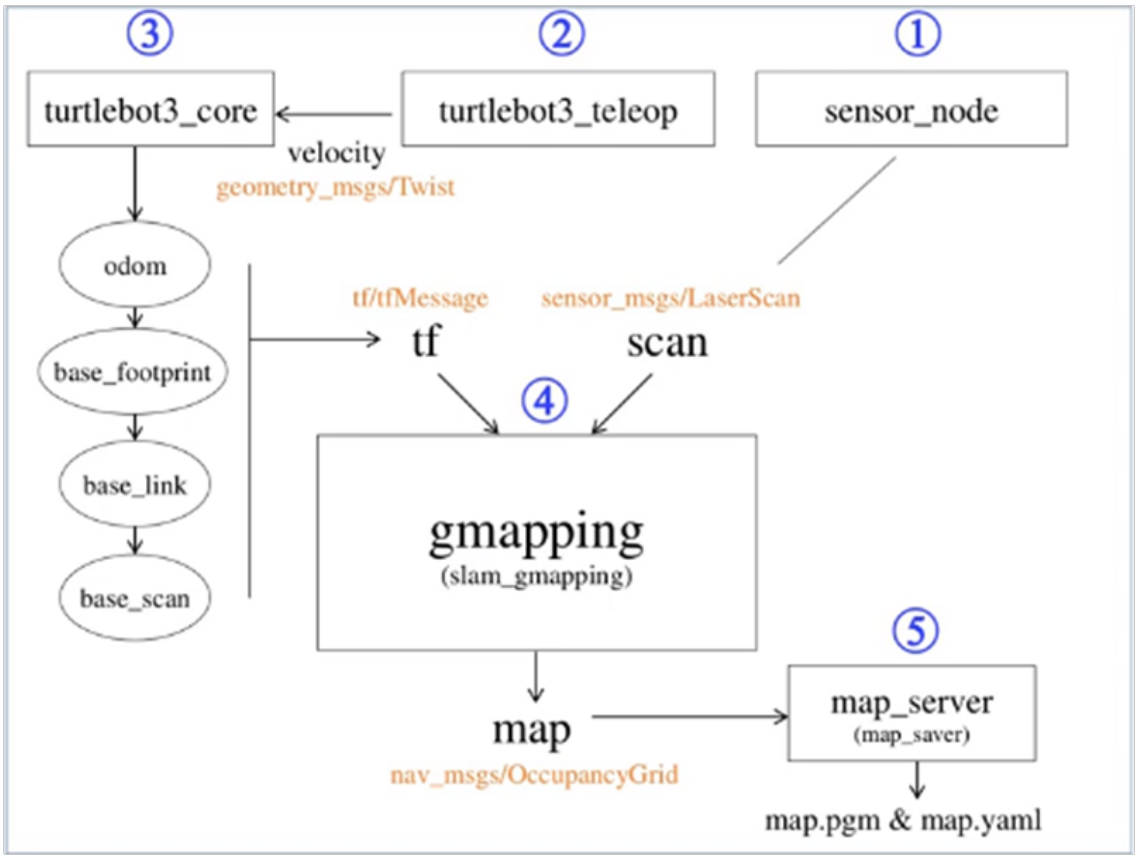


Figure 2.1: Gmapping

A Light Detection and Ranging (LiDAR) is part of the robots sensor node, and when it is turned on, it sends scan data through a laser scan topic. The telemetry core node in Fig. 2.1 starts working at the same time and gets data on linear and rotary speeds from a teleoperation node. After that, the telemetry core node sends out automatic orders and transforms (TF) that are linked in the telemetry model. In ROS, the tf package handles transformations and how different coordinate frames relate to each other. After that, the GMapping node draws a map using the robots TF and LiDAR scan data. The GMapping SLAM algorithm uses sensor data and odometry information to make a map of the surroundings and guess where the robot is at the same time. After making the map, the map server nodes are run to get a map file. For the navigation step to work, this map file is very important. It gives the robot a sense of it is surroundings that it can use to move around on its own. For accurate mapping and movement in the robots working area, the TF framework makes sure that all of its parts use the same coordinate transformations.

2.1.2 Hector-Slam

Hector SLAM, also known as Hector Simultaneous Localization and Mapping, is a highly utilized algorithm in the field of robotics for the purpose of mapping and

determining the location of objects. An important advantage of this technology is its capacity to do Simultaneous Localization and Mapping (SLAM) in real-time, enabling robots to dynamically traverse and comprehend their environment. Hector slam demonstrates exceptional performance in contexts characterized by constrained computational resources, rendering it very suitable for deployment in applications such as autonomous restaurant robots. The utilization of a laser scanner for mapping ensures precise and intricate depictions of the surroundings. Nevertheless, there are obstacles that arise when dealing with extensive settings and the possibility of gradual deviation. To resolve these problems, it is necessary to meticulously adjust and calibrate the system, and ongoing progress is being done to improve the resilience and expand ability of hector slam for wider implementation in the field of robotics. Hector-SLAM works very well in real time, which means it can be used in situations where the robot needs to work quickly and efficiently. The algorithm can be set up quickly, which means that the robot can start moving right away after being released. Hector-SLAM might use a lot of memory, which could be a problem for a restaurant robot that doesn't have a lot of tools on board. Effectively managing big loops might be hard in places where spaces are connected [16].

2.1.3 Cartographer

It is possible to make thorough 3D maps with Cartographer, which makes it useful in situations that need it. The algorithm is made to handle loop closing well, which improves the accuracy of mapping when the robot goes back to an area. Cartographer allows multi-sensor fusion, which lets the robot combine data from different sensors to get a better picture. Setting up a Cartographer to work in a certain environment can be hard, and you might need to modify some settings, which could make quick launch harder. Cartographers can use a lot of resources, which could be a problem for a restaurant computer that can not do a lot of computing [16].

2.2 Localization

In robotics, localization is the process of figuring out where and how a robot is positioned in it is surroundings. This is a very important part of independent robotics because it lets the robot know where it is in relation to a map of its surroundings. Different algorithms are used to find the location of something. The Kalman filter algorithm and the particle filter algorithm are two popular ones [25].

2.2.1 Kalman Filter Algorithm

The method developed by Jeong et al. (2014) to track different objects is based on the Kalman filter. The cost function is configured using a few parameters, and the number of Kalman filters it runs is equal to the number of moving objects in the frame. Matlab experiments demonstrate the multi-item tracking capability of the proposed technique. A method for locating objects and corresponding with them via ROS master messages is built into a NuBot soccer robot. TX2 and the industrial computer of the robot exchange these messages. It is a challenging profession in the

realm of computer vision, though. When some of the things are in the way of one another, it becomes more difficult to match them correctly. Numerous techniques, including the mean shift based object tracking by L. Ido, the condensation filter by K. Hyun-Bok, the dynamic Bayesian network by Wu, the feature correspondence for occlusion handling by Tao Yang, and others, have been proposed for successful multiple object tracking. Still unresolved, though, is the issue of tracking several objects in rapidly changing scenes. A robust foundation for tracking objects in ambiguous conditions is offered by the Kalman filter. It can track many objects effectively and consistently while accounting for occlusion problems in dynamic environments. The issue of tracking several items in dynamic scenes is resolved by doing this [26].

2.2.2 Particle Filter Algorithm

The Particle Filter Algorithm is a probabilistic localization method that is widely used in robotics, such as for self-driving robots in restaurants. It works by using a set of particles, each with a chance of being the robots real location, to show what it thinks about its location. The algorithm changes the particle weights based on sensor readings and movement data as the robot moves and looks around. Particles with higher weights are copied during resampling, while particles with lower weights are thrown away. This process brings the particle cloud closer to where the robot is, which gives a good idea of where it is in settings that change over time. Particle Filter can handle uncertain and non-linear sensor data, which makes it a good choice for restaurant robots that need to move through busy and changing areas on their own. Its ability to respond to real-time updates and strong performance in the face of doubt make it useful for improving the accuracy of localization in these particular situations [25].

2.3 Path Planning

Restaurant robot path planning involves determining the optimal trajectory for the robot to navigate within the restaurant. Local route planning is a method used to enable robots to navigate around tables and customers by addressing immediate obstacles and making necessary adjustments. The utilization of sensors and real-time data enables the avoidance of collisions through prompt decision-making. Nevertheless, global path planning takes into account the arrangement of the restaurant [27]. It is necessary to do environmental mapping and establish a general route. This meticulous approach enables the robot to fulfill its objective while taking into account spatial constraints. Navigation method we can see that the process of a robots navigation where map, localization, global path planning and local planning works in Fig. 2.2 together to complete the navigation process. Local and global path planning must be interconnected to ensure seamless navigation. The global plan provides a predetermined path, whereas the local plan adapts to the prevailing circumstances. The two collaborate to assist the restaurant robot in circumventing barriers and adhering to the overarching plan. The robot may respond to the restaurants real-time adjustments and broader spatial arrangement using this integrated approach.

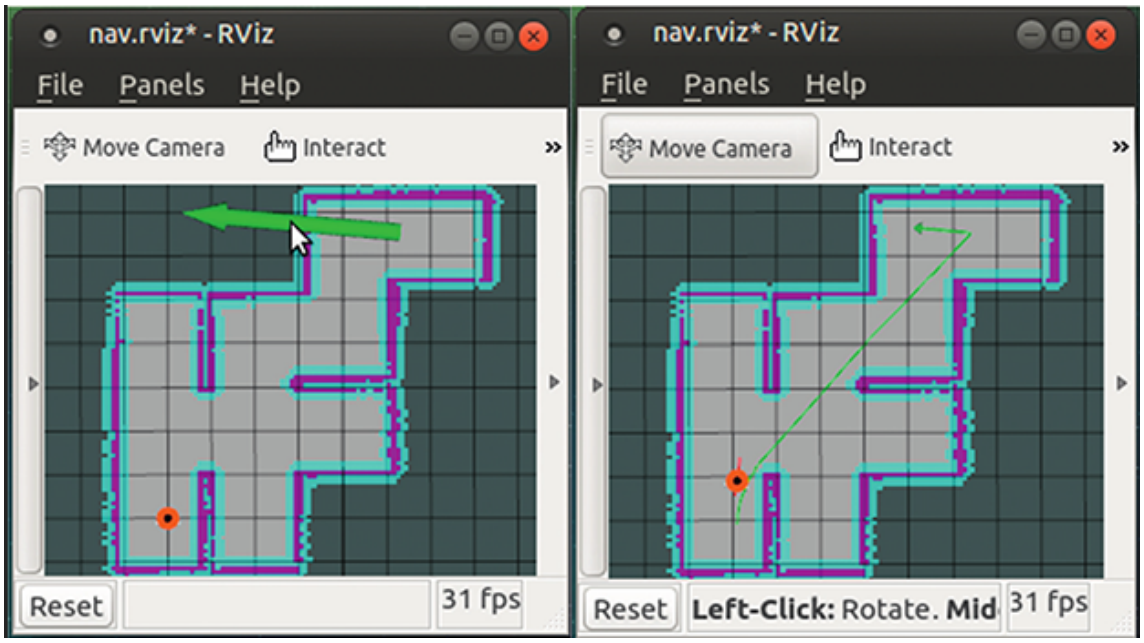


Figure 2.2: Path Planning [10]

2.3.1 Local Path Planning

The goal is to move the robot around in real time while keeping an eye on its near surroundings and avoiding obstacles. The Dynamic Window Approach (DWA) is a popular local path planning technique. Based on the robots kinematics and dynamics, it dynamically considers a window of potential velocities and chooses the one that optimizes a particular objective function.

Differential-Drive Dynamic Window Approach, or DWA, is a motion planning approach that is frequently used in robotics, particularly for path planning and navigation in the restaurant robotics industry. Sensor data from laser range finders, depth cameras, or other perception sensors are sent to the local planner so that it can understand its near surroundings. The robots allowed speeds and angle speeds are shown by the dynamic window. It is constantly changed based on the current situation, which lets the robot think about what it can do right now. The local planner looks at possible paths for each set of speeds in the dynamic window by modeling the robots movement over a short period of time. Trajectories are given points based on how close they are to obstacles, how well they line up with the goal, and how smooth they are. The best route is chosen as the one with the highest score. This path tells the robot how to move right now. The DWA algorithm is often built into ROS as part of the Navigation Stacks local planner function. The local planner gets information from sensors and the global path, then makes a path for the robot to follow and sends it out into the world. While trying to line up the robot with the world path, DWA tries to avoid collisions first. It lets the robot move through changing surroundings, avoid obstacles, and get where it needs to go [28]. Robots benefit from the Dynamic Window Approach (DWA) motion planning

algorithm in dynamic situations like restaurants. It generates a dynamic window of allowable velocities for real-time obstacle avoidance and safe crowded navigation. DWA avoids sudden curves with velocity limits, improving security. Trajectories are

evaluated using cost functions for obstacle avoidance, essential for safe navigation in dynamic settings. DWA optimizes delivery by setting local goals for table-to-kitchen routing. Its adaptive velocity control adjusts speed dynamically, boosting efficiency and safety in busy locations. DWA reduces wait times for robots handling orders and delivery, improving customer service [28].

2.3.2 Global Path Planning

The objective is to locate a path that is either optimal or nearly optimal from the current position of the robot to a goal position while taking into account the complete environment. Approaches that are based on grids involve representing the environment as a grid in which each cell is either currently occupied or vacant. Dijkstra's algorithm and A* (A-star) are two examples of algorithms that are frequently utilized. Approaches that are based on graphs involve representing the environment as a graph, with nodes representing distinct places and edges representing potential transitions [29]. Applications include algorithms such as dijkstra's algorithm and A* algorithm.

Cost Function for A* Algorithm

The cost function for the A* algorithm is given by:

$$f(n) = g(n) + h(n)$$

Through node n, $f(n)$ shows how much it is thought to cost to get from the beginning to the end. The cost of the road from the beginning to node n is $g(n)$. As a rough guide, $h(n)$ tells you how much it will cost to get from point n to the goal [29].

2.4 Object Recognition

Object recognition is a computer vision technology that allows machines to accurately detect and classify things present in an image or video stream. The process involves collecting distinctive characteristics and patterns from visual information and comparing them to pre-existing models or databases. Within the framework of an autonomous restaurant, the utilization of object recognition demonstrates numerous benefits for a range of jobs. The restaurant robot uses cameras and image-processing algorithms to detect and recognize items such as tables, seats, and plates, which helps it navigate and interact more effectively. The utilization of this technology proves highly advantageous for a restaurant robot as it allows the system to proficiently maneuver through congested areas, evade obstructions, and manipulate objects with accuracy. Consequently, this technology greatly enhances the robot's operational efficiency and safety within the dynamic setting of a restaurant [13].

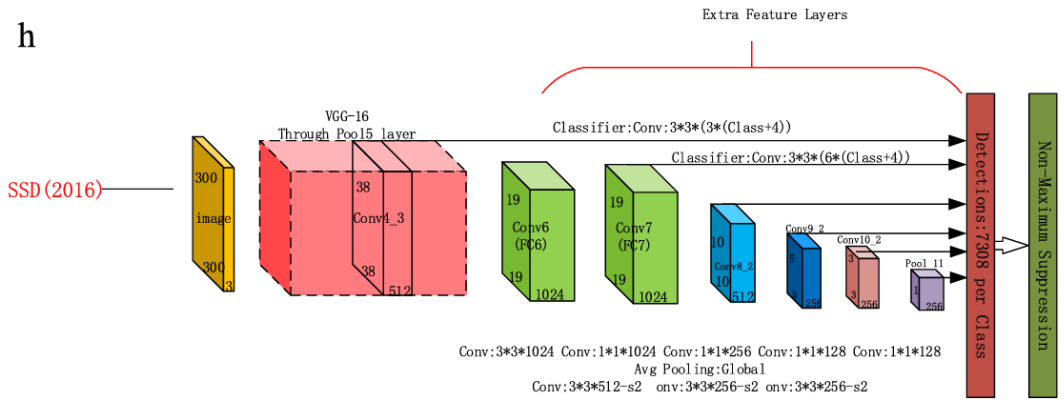


Figure 2.3: Network structure of SSD-MobileNetV3 [30]

2.4.1 SSD_MobileNet_V3

MobileNetV3 has an optimized structure that delivers exceptional accuracy and speed for tasks such as object recognition and image classification, surpassing current performance benchmarks. The SSD_MobileNet_V3 model integrates the MobileNetV3 architecture in Fig. 2.3 and the SSD algorithm to provide both accurate object identification and efficient feature extraction. The SSD_MobileNet_V3 model utilizes MobileNetV3 as the underlying network for extracting features, with the aim of achieving precise real-time object recognition on devices with limited resources. SSD_MobileNet_V3 is a deep learning model designed primarily to achieve rapid and precise object identification on mobile and edge devices. It utilizes the multi-scale object localization capabilities of SSD and the efficient feature extraction capabilities of MobileNetV3, making it ideal for applications with restricted computer resources [30].

2.4.2 Faster R-CNN

Object recognition is a crucial task in computer vision, and Faster R-CNN, also known as Faster Region-Based Convolutional Neural Network, is an innovative deep learning model designed specifically for this purpose. Faster R-CNN in Fig. 2.4 is an integrated region proposal network, efficiently generating region suggestions for potential object locations inside an image. The Region Proposal Network (RPN) uses anchor boxes of different sizes and shapes to propose areas. It does this by analyzing the convolutional feature maps produced by the backbone network. Utilizing a RPN eliminates the need for external region proposal methods, resulting in a seamless and trainable item identification process. Faster R-CNN utilizes a Convolutional Neural Network (CNN) as its basis to extract hierarchical information from the input image [13]. Any form of backbone network can be utilized, although networks with topologies capable of capturing intricate visual components, such as ResNet or VGG16, are frequently selected. The Region of Interest (ROI) pooling technique is employed to align the region proposals generated by the RPN to a predefined size, facilitating consistent feature extraction. ROI pooling is a technique used to simplify the processing and classification of variable-sized regions by converting them into a fixed-size representation. In Smarter R-CNN, a bounding

box regressor is employed to refine the bounding box coordinates, while a classifier network is utilized to provide class labels to each suggested region. These features enable accurate categorization and identification of objects within the designated areas. The concept of utilizing a Region Proposal Network to generate region proposals within the same network was initially introduced by the pioneering object detection model Faster R-CNN . The integration enhances precision and effectiveness in real-time object recognition applications by optimizing the object detection process and enabling end-to-end training [21].

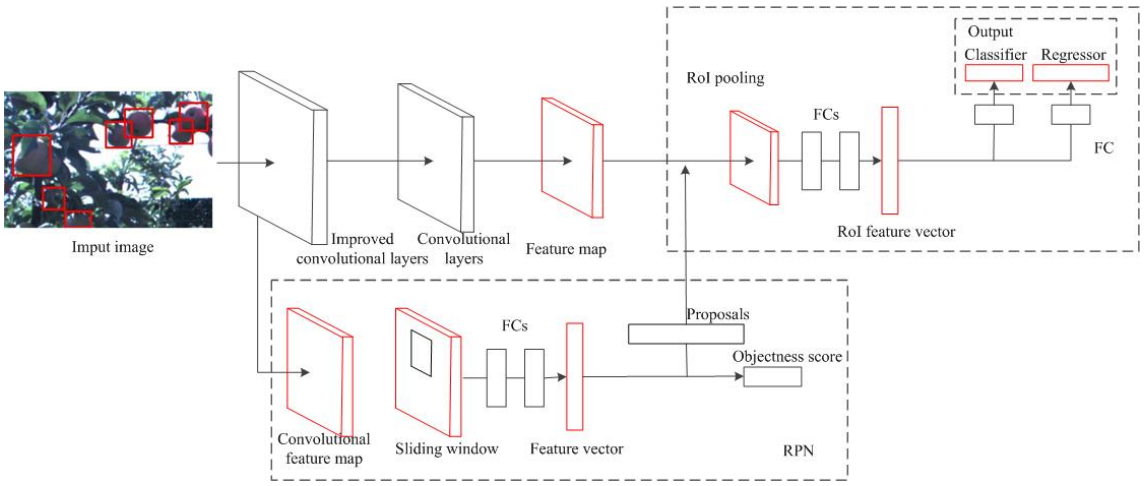


Figure 2.4: Faster R-CNN [13]

2.5 SpeechRecognition

Speech recognition is a technological process that translates spoken language into written text, as well as converting written material into spoken words. This enables machines to understand and react to spoken orders given by humans. The system functions by employing acoustic and linguistic models to analyze and interpret audio data, thereby identifying spoken words and translating them into written text. It also has the capability to convert written text into spoken words using text-to-speech technology. Speech recognition is beneficial for human-robot interaction in an autonomous restaurant setting. It improves the user experience by enabling customers to verbally place orders, obtain information, or provide directions. This technique is especially beneficial in a loud restaurant setting where conventional entry methods may be less efficient. Through the incorporation of speech recognition technology, autonomous restaurant robots are able to effectively comprehend and address client inquiries, hence enhancing a smooth and instinctive contact between customers and the robotic system [12].

2.5.1 Natural Language

Restaurant focused recognition is a sort of AI that enables robots to locate and establish connections with eateries. Thanks to this technology, robots can now in-

teract with restaurants and navigate their surroundings by reading menus, comprehending orders, and even having conversations with patrons. Robotics uses natural language processing and computer vision to identify restaurants. To identify the items in a restaurant, such as menus, tables, and seats, computer vision is employed. Using natural language processing enables you to ascertain the customers needs and provide them with the appropriate response. There are several applications for robot-based restaurant recognition. People can order food, pay for their meal, and navigate a restaurant with the assistance of robots. Furthermore, individuals can receive customized recommendations from robots depending on their interests. Although it is still in it is infancy, robot-based restaurant identification has the potential to significantly alter the food industry. Due to their autonomous recognition and interaction capabilities, robots can improve customer satisfaction and operational efficiency in restaurants [22].

2.6 The Main Menu Language

The main menu language of a robot in a restaurant should be simple and easy to understand. It should be able to communicate the different options available to customers in a clear and concise manner [12]. For example, the robot could say "Welcome to our restaurant. Please select from the following options: Appetizers, Entrees, Desserts, Drinks, and Specials."

2.6.1 The Alternate Menu Language

The alternate menu language of a robot in a restaurant should be more conversational and engaging. It should be able to engage customers in a friendly manner and provide more detailed information about the different options available. For example, the robot could say "Happy Birthday" or able to tell short jokes [22].

2.7 Machine learning

According to Bai et al. (2020), machine learning is being utilized more and more to combine robots and machine vision to improve grab speed and accuracy. CNNs are a worldwide center of research, but the acquisition of labeled data restricts their expansion. The research on tactile feedback in the domains of robot grasping and target recognition is reviewed in this paper. It has been discovered that combining vision and touch feedback can improve the accuracy and success of robot grasping. Many applications such as regular machine learning, deep learning, reinforcement learning, unsupervised learning, self-supervised learning, and merging vision and touch have been carried out using it. Conventional machine learning techniques are fast, simple to understand, and need little data. However, these programs become less and less effective as the volume of data increases. The success of the Alex network in 2012 led to a resurgence of the deep neural network. It is applied in numerous machine vision domains. Unsupervised and self-supervised learning

systems have been developed since labeled data is necessary for deep learning. Reinforcement learning considers issues that occur repeatedly and considers long-term returns, whereas supervised learning considers problems that occur just once and considers short-term returns. A novel approach to studying robot grasping involves attaching pressure sensors to the dexterous hand to provide tactile feedback and combine it with vision [21].

2.8 Neural Networks

This literature review explores diverse applications of artificial intelligence and neural networks in robotics. Jiang et al. delve into the evolution of You Only Look Once (YOLO) algorithms for efficient object detection, while Kim et al. employ neural networks for dynamic object recognition in robotic systems [31]. Luo et al. utilize convolutional neural networks for soccer robot tracking, Jeong et al. introduce a Kalman filter-based method for multi-object tracking, and Bai et al. discuss the integration of machine learning, vision, and tactile feedback in robot grasping. The review also touches upon the emerging field of robot-based restaurant recognition, emphasizing language customization and event detection in the restaurant industry [32].

2.9 CNNs

Luo et al. (2017) present a novel approach to track and locate soccer robots in the RoboCup MSL using convolutional Neural Networks (CNNs) in their 2017b study. The method consists of two steps: first, the robot is located using an RGB picture, and then it is located using a depth point cloud. The suggested approach demonstrated good performance in terms of Mean Average Precision (MAP) and high accuracy for robot recognition in the MSL competition. This will support strategy planning and future barrier avoidance. CNNs have proven to be quite effective in locating objects, segmenting images, and identifying certain areas and objects [32]. Artificial neural networks in deep learning on the MSL are trained with images from an omnidirectional vision sensor to find objects. A Deep Learning approach for locating NAO robots was introduced in the RoboCup Standard Platform League, and it functions incredibly well. With the state-of-the-art object recognition technology YOLO v2, the entire image is processed by a single neural network. Next, the bounding boxes and probabilities for every position are predicted by this network. Nowadays, object recognition algorithms like SSD, YOLO v2, Faster-RCNN, and others may operate at various resolutions, making it simple to establish the ideal balance between accuracy and speed (67 FPS with mAP 76.8) [32].

2.10 YOLO

Jiang et al. (2022) highlighted the YOLO approach and its more sophisticated variations are introduced. According to the findings, there are distinctions as well as

parallels between the many iterations of YOLO and CNNs. YOLO V2, V3, V4, and V5 are the varying iterations of the meme. Limited edition versions exist as well, such as YOLOLITE. YOLO can detect time-based videos quickly because it only needs to submit an image to the network and wait for the outcome. YOLO directly employs the global picture for identification [33]. In order to reduce the possibility of the background being mistaken for the item, this might encrypt the data globally. Although the challenge of target identification is transformed into a regression problem, identification accuracy still requires improvement. YOLO uses a considerable number of lower sampling levels in order to advance more quickly. Detection, Darknet-19, High Resolution Classifier, Fine Features, Multi-scale Training, and Classification Training are a few of these [34].

2.11 NN-based object detection

According to K. Kim et al. (2017), the dynamic identifying objects system uses neural networks to recognize landmark features, which enables it to function similarly to robots that can pick objects at random, take items out of bins, and visually reposition objects. Determining the object's location, posture, distance, and type is how the work is evaluated. In this study, moving objects are recognized using a neural network-based object classifier and feature extractor. To minimize the impacts of illumination, a dynamic recognized object system includes dynamic object recognition, linked NN-based object detection, and landmark feature extraction. The use of neural network-based feature detectors has made it possible to identify landmark features that distinguish between an object area and a complex background. Neural networks are very good at generalizing to slightly changing or absent data, hence NN-based feature descriptors have been used to detect objects even when their properties have changed [33]. The performance evaluation approach includes finding and labeling several types of things, including bin-picking objects, visual servoing objects, and other varieties of mixed objects. To evaluate the effectiveness of object recognition, robot operators have been given tasks such as selecting which objects to move and where to place them [35].

2.12 Related Work

A paper with the title "ORB-SLAM: A Versatile and Accurate Monocular SLAM System" presents ORB-SLAM, a real-time monocular simultaneous localization and mapping (SLAM) system. ORB-SLAM is specifically built to operate effectively in various indoor and outdoor environments, regardless of their size [36].

In the article "Non-Contact Service Robot Development in Fast-Food Restaurants" emphasized the use of ORB-SLAM, a real-time monocular simultaneous localization and mapping (SLAM) system. ORB-SLAM is specifically designed to operate effectively in various indoor and outdoor environments, regardless of their size [37].

The study titled "A Collaborative Visual SLAM Framework for Service Robots" discusses a specialized framework for service robots that enables cooperative visual

simultaneous localization and mapping (SLAM). By utilizing an edge server that oversees a map database and does global optimization, each robot has the ability to join an existing map, update the map, or create new maps. These tasks are made easier by using a standardized interface, resulting in very little computational and memory costs. We have developed an advanced communication system that enables robots to share information in real-time. By utilizing a cutting-edge method of organization and retrieval on the server, every robot is able to access landmarks that are expected to be within its visual range, thereby improving its local map [38].

The proposed scheme titled "Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing," focuses on mobile robots specifically designed for indoor use. These robots are commonly employed in industrial environments, such as large logistics warehouses, where their main tasks include collecting and sorting products. For these robots, tasks that need a lot of computing power make up a significant part of their total energy usage, which in turn affects how long the battery lasts [39].

The study titled "A Lidar SLAM based on Improved Particle Filter and Scan Matching for Unmanned Delivery Robot" discusses the rapid advancement of robotics. Intelligent robots have increasingly become essential in human daily life, performing specific tasks and greatly improving convenience. SLAM technology is crucial in guiding mobile robots to effectively carry out delivery operations in confined spaces like dining establishments, hotels, and logistics warehouses. The algorithm utilizes a scan matching technique that incorporates both point and line features. It takes advantage of the geometric information from the two-dimensional point cloud by employing a unique error function. This methodology demonstrates higher matching accuracy compared to standard methods that rely purely on point features. The researchers perform two-dimensional SLAM tests in both the Gazebo simulation environment and a real-world natural context. The results suggest that the suggested approach exhibits superior mapping precision in comparison to the commonly employed two-dimensional SLAM technique, Gmapping, at this point in time. The efficacy of their technique is further supported by path planning experiments conducted on a two-dimensional grid map, utilizing our algorithm and an enhanced A* algorithm [40].

In their paper titled "Application of Simultaneous Location and Map Construction Algorithms Based on Lidar in the Intelligent Robot Food Runner," Jia Cheng, Zhendong Liu, Jiaxi He, Yuting Deng, and Hao Zhang examined the rapid progress of artificial intelligence. They highlighted the growing prominence of AI applications in dining establishments. The expensive cost of intelligent robot food runners on the market is impeding their wider adoption. This study presents a cost-efficient and highly efficient intelligent robot food runner specifically designed for restaurant applications. This research aims to investigate the utilization of laser radar SLAM for the purpose of constructing restaurant maps, achieving localization, and facilitating navigation. Furthermore, it explores the advancement of tactics for avoiding obstacles and designing paths. By utilizing the ROS platform, the entire operation of the intelligent robot food runner is simulated and verified, effectively meeting the precise demands of restaurants [41].

The research titled "Visual Perception Framework for an Intelligent Mobile Robot" demonstrated that visual perception is a fundamental capacity essential for intelligent mobile robots to interact efficiently and safely with humans in real-world settings. In recent years, there has been significant and innovative development in deep learning, resulting in notable breakthroughs in vision technology. However, the incorporation of various visual perception techniques into robotic systems is currently in its initial phases and lacks confirmation through real-world testing. This article presents a visual perception framework specifically developed for an intelligent mobile robot. Our framework is based on the robot operating system middleware and smoothly integrates a diverse range of advanced algorithms. These algorithms are capable of accurately identifying individuals, objects, human poses, and offering detailed descriptions of observed scenarios. The proposed framework's effectiveness and applicability are evaluated in diverse and demanding scenarios encountered in international robotics competitions, employing two mobile service robots [42].

The paper titled "To build a smart unmanned restaurant with multi-mobile robots" presents a comparison between a single robot and a system comprising many mobile robots, highlighting the various advantages of the latter. By breaking certain jobs into smaller parts, many robots can work on different subtasks at the same time, which improves total performance. Furthermore, the cooperation among the members enhances the stability of the robot, enabling the development of more sophisticated designs for robots specialized in particular jobs and increased adaptability. This study proposes the creation of an automated smart restaurant by utilizing the combined capabilities of many robots. The MATLAB software is used as the preferred platform for this system, while the Vision C# a design tool is employed to model the ultimate result. The system employs dynamic allocation of robots, adjusting their numbers according to the number of guests. It prioritizes assigning the nearest robot to a task through decision-making processes. In addition, the system utilizes Particle Swarm Optimization for path planning, which results in faster convergence speed and enhances global search capabilities. This method enhances communication, coordination, and collaboration among the robots, allowing them to independently handle tasks such as guest hosting, meal delivery, table cleaning, and other operations in the restaurant [43].

The A* method in artificial intelligence is a very effective pathfinding algorithm that effectively determines the shortest path in a graph by taking into account both the current cost and an estimation of the remaining cost. The paper titled "A path planning method based on the geometric A-star algorithm" proposes an approach to address the challenges of navigating an Automated Guided Vehicle (AGV) through complex paths with numerous nodes, long distances, and sharp turning angles. These challenges commonly arise in sawtooth and cross paths generated by the traditional A-star algorithm. This method is highly effective for path planning [44].

In the paper titled "An approach to restaurant service robot SLAM" we aim to find the shortest weighted path using the A Star algorithm. We improve the algorithm by adjusting the weight of the restaurant channel in real-time based on the degree of congestion. To validate our results, we use a gridded map of a restaurant. The findings demonstrate the effectiveness of the suggested method in comparison to the

conventional gridded map of a restaurant. An A* algorithm with a fixed path weight and enhanced performance. The utilization of the A* algorithm can effectively mitigate channel congestion and enhance the operational efficiency of mobile service robots. The restaurant service robot systems find it to be of significant practical utility. In this study, the A* algorithm is defined as the most efficient method for path planning, surpassing other algorithms in terms of finding the shortest path [45].

A study titled "Optimal mapping trajectory for autonomous robot waiter" presents the findings of a series of tests conducted with Dijkstra's algorithm. The results demonstrate that the robot is capable of efficiently navigating from the starting point to the desired destination, even when there are modifications to the working environment and the movement is executed smoothly using the three omni-directional trajectories. In summary, it is evident that the Dijkstra algorithm operates with precision and finesse in the domain of robot path planning [46].

Chapter 3

Methodology, Requirement Analysis, Implementation

The research methodology adopts a multimodal approach to create an autonomous restaurant robot in Fig. 3.1 with core functionalities in object recognition, speech recognition, and navigation. Navigation, a primary focus, is implemented and tested through simulation using Simultaneous Localization and Mapping (SLAM) methods, aiming to identify optimal techniques for smooth restaurant navigation [16]. Object recognition involves sequential processes such as image capture, annotation, labeling, dataset generation, and rigorous testing of various models to determine the most accurate and effective one for integration into the robot. For voice recognition, Application Programming Interfaces (APIs) are employed to enhance human-robot interaction, enabling the robot to comprehend and respond to spoken orders in a restaurant setting, contributing to a coherent and context-aware speech experience [47].

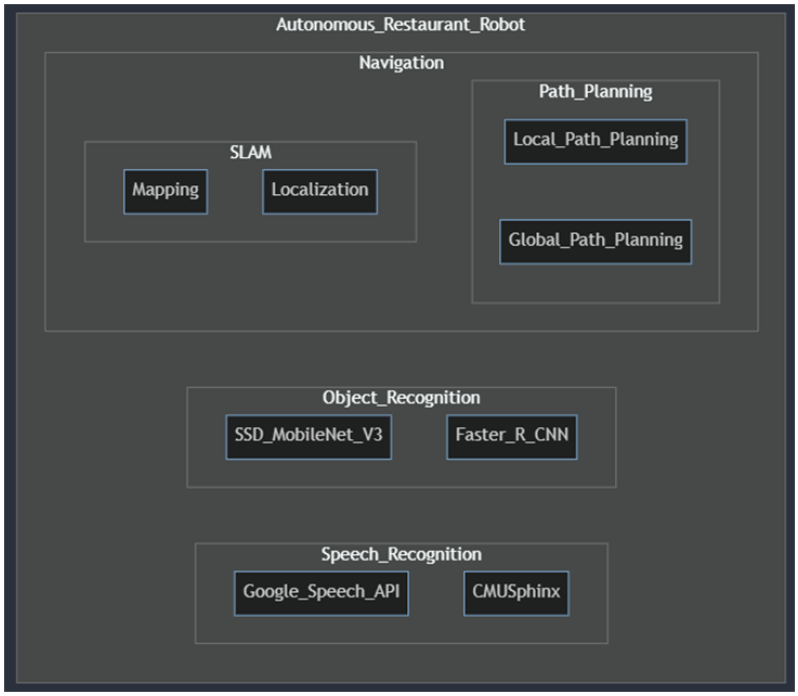


Figure 3.1: Autonomous Restaurant Robot Diagram

3.1 Platform, tools and asset selection for Navigation

The selection of a suitable platform, tools, and assets is vital for the navigation of a robot. The choice of the platform, such as Linux 20.04, Gazebo, and Rviz, is based on the specific requirements of the robots environment, considering factors such as terrain, size, and weight capacity. Robot Operating System (ROS), Simultaneous Localization and Mapping (SLAM) algorithms, object identification, and speech recognition algorithms are essential tools that enable the robot to comprehend and map its environment. Lidar sensors are commonly favored for precise distance readings in navigation. By combining dependable sensor and software components, the robot may achieve effective navigation, obstacle detection, and smooth interaction with its surroundings, resulting in a resilient and competent system.

3.1.1 ROS Noetic

Robot Operating System (ROS) Noetic is a middleware framework designed to develop software for robots. It provides a flexible and distributed architecture for developing modular and scalable robotic systems. In this study, we use ROS Noetic as the basis for the TurtleBot3 robots Simultaneous Localization and Mapping (SLAM) simulation. In order for various software components, or "nodes," to share information without any problems, a communication framework is the core of ROS Noetics functioning. These nodes can be created in multiple programming languages, which makes it easier for different modules to communicate with one another. ROS Noetic is a stable and adaptable framework that we have chosen for our researchs TurtleBot3 SLAM simulation. Our research goals are aided by ROS Noetic, which makes use of simulation capabilities, modular architecture, and community support to make it easier to develop and test complex robotic systems [48].

3.1.2 Ubuntu Linux 20.04

The popular and adaptable open-source Linux distribution Ubuntu 20.04 is renowned for its dependability, security features, and robust package management system. Ubuntu 20.04 is the operating system used in our study for installing and setting up ROS Noetic packages. Like other Linux distributions, Ubuntu 20.04 runs on a kernel resembling Unix and adheres to the open-source and free software philosophy. The selection of Ubuntu 20.04 as the operating system for our research provides a stable, secure, and well-supported environment for deploying ROS Noetic. Its community support, large package repository, ROS interoperability, and user-friendly features all work together to provide a research technique that is both simplified and effective [49].

3.1.3 Gazebo Simulator

A popular open-source robot simulation program in the robotics community is called Gazebo. It offers a platform that is both realistic and expandable for simulating robots and their interactions in various situations. In our study, Gazebo is used as the simulation environment, specifically for the TurtleBot3 SLAM simulation, to

test and validate ROS Noetic packages. Gazebo Simulator is a feature-rich robotic system simulation tool that comes with the following main attributes: Sensor simulation, ROS integration, physics engine, and 3D visualization. The following are some advantages of using Gazebo Simulator: reproducibility, compatibility with ROS, realistic simulation, sensor and actuator simulation, and community support [38].

3.1.4 Testing of Simultaneous Localisation and Mapping Algorithm

Simultaneous Localization and Mapping (SLAM) is a key part of robotic navigation. It helps robots not only understand and map their surroundings, but also figure out where they are in that area. In this study, we applied three different Simultaneous Localization and Mapping (SLAM) algorithms in Fig. 3.2 to enable the TurtleBot3 robot to navigate and map on its own. The three SLAM algorithms (gmapping, Hector SLAM, and Cartographer) that were selected were carefully applied to different robotic navigation tasks [49]. The implementation of Gmapping, which is another name for Grid-based FastSLAM, aimed to generate a probabilistic occupancy grid map of the surrounding area. The program concurrently creates a map using data from laser scans and estimates the robots attitude using particle filters [50]. Hector SLAM uses a grid-based mapping technique in conjunction with a scan matcher to predict the robots trajectory and produce a high-resolution map. The method works very well in settings with lots of open areas and few loop closures. Google created Cartographer, a flexible SLAM system that makes use of both 2D and 3D mapping methods. To produce intricate maps, it combines data from multiple sensors—including Light Detection and Ranging (LIDAR) and Inertial Measurement Unit (IMU) using a global optimization technique. Our research employed a methodical approach that included parameter adjustment, data collection, and thorough testing in order to deploy gmapping, Hector SLAM, and Cartographer [51]. The goal in choosing these algorithms was to provide us a comprehensive understanding of their capabilities so that we could base our selections on the particular needs of autonomous navigation in various contexts [45] .

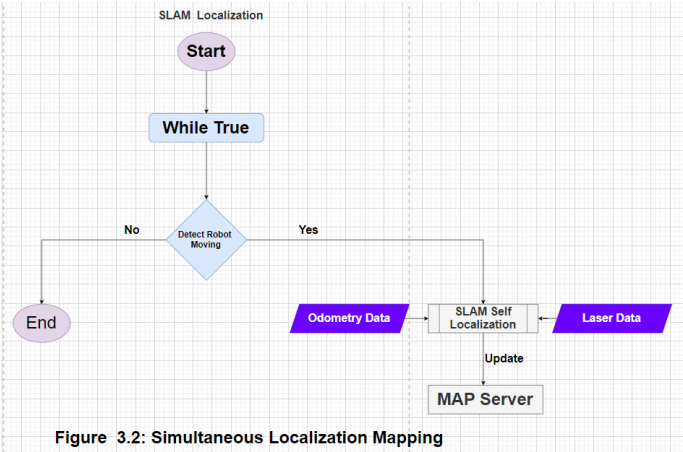


Figure 3.2: Simultaneous Localization and Mapping.

3.1.5 Rviz

The data processing stage of our study uses ROS Visualization (Rviz) to create dynamic 3D graphs and visual representations from the outputs of the SLAM algorithm. A useful tool for analyzing and assessing how well implemented algorithms perform is Rviz. To summarize, the process of creating a Rviz graph entails setting up ROS subjects, starting Rviz with the proper parameters, setting up visual representations, and utilizing Rviz interactive features to conduct thorough data analysis. The SLAM algorithms performance in the simulated environment is assessed and understood better thanks to the insights and visualizations produced by Rviz.

3.1.6 Object Recognition (Faster R-CNN)

A robots object recognition quickly identifies and categorizes nearby things to help find misplaced items. The robot can efficiently scan and analyze the environment to find lost items using modern computer vision techniques. The system can also save accessory information like location, description, and usage for intelligent retrieval. This makes the robot a better assistant, especially when users need help finding things or accessories.

3.1.7 Data Collection

Collecting a wide range of picture data that accurately depicts the restaurant setting in which the autonomous robot will work. After that it ensure that the dataset includes a range of backgrounds, objects, and lighting scenarios that the robot is probably going to come across.

3.2 Annotation and Labeling

Labeling objects of interest in the photos with manual annotation tools. To train the Faster R-CNN model, annotate each objects bounding box with the class label. Additionally, we have partitioned the annotated dataset into test, validation, and training sets. The validation set helps with hyperparameter tweaking, the test set assesses the models ability to generalize to new data, and the training set is used to train the model.

3.2.1 Model Selection and Training

We opted for the Faster R-CNN architecture for object recognition, enabling precise item localization and classification by combining a classifier with a region proposal network (RPN) [13]. Initially, the Faster R-CNN model was pre-trained using weights from a large-scale dataset like COCO, leveraging transfer learning. Fine-tuning was performed on the annotated restaurant dataset, with ongoing performance assessment on the validation set. Model optimization involved adjusting hyper parameters, such as batch sizes and learning rates, based on validation metrics.

3.2.2 Testing and Results Analysis

The finalized model underwent testing on a dedicated set of cases to ensure applicability to new, untested data. Evaluation encompassed correctness and robustness under diverse scenarios, including practical trials on a self-governing food service robot equipped with the Faster R-CNN object identification mechanism. Precision in object identification and localization within the bustling restaurant environment was assessed. Results were documented in the analysis, considering model correctness, computational efficiency, and real-world performance. Challenges encountered were outlined, along with potential enhancements for future tasks.

3.2.3 Speech Recognition

This approach guides the speech recognition technology of the restaurant robot through an API. After a thorough examination of available speech recognition APIs, considering factors such as cost, integration ease, language support, and accuracy [47], we selected an API aligned with the robot's specifications. The emphasis is on meticulous API selection, integration with ROS, data gathering, potential model training, system integration, testing in virtual and real scenarios, correctness assessment, continuous development, and documentation for the thesis report.

3.2.4 Initialization and Setup

- Importing `pyttsx3` and `speech_recognition` libraries.
- Initializing the text-to-speech engine using the 'sapi5' backend, and setting the voice.

3.2.5 Functions Definition

- `speak(audio)`: for text-to-speech.
- `takeCommand()`: for speech recognition.

3.2.6 Speech Recognition

The `takeCommand()` function captures audio input through the microphone, prints "Listening..." to indicate that it's ready, and listens for user input. The recognized speech is printed, and the function returns the recognized query.

3.2.7 Intent Matching

The `get_intent(query)` function matches the recognized query with predefined intents in the data list. It uses a simple scoring mechanism based on word matching to determine the best-matched intent.

3.2.8 Response Generation

The response (intent) function generates appropriate responses based on the matched intent. For "jokes" and "wish" intents, random jokes and birthday wishes are fetched from dedicated functions (`fetch_joke()` and `fetch_birthday_wish()`). For other intents, random responses from the predefined list are selected. Data Definitions: The data list contains dictionaries for each intent, specifying associated queries and responses.

3.2.9 Main Loop

The main loop continuously listens for user input, determines the intent, and responds accordingly. Overall, the model forms a conversational interface where the restaurant robot can engage in simple interactions, providing information, jokes, birthday wishes, and responding to various user queries. It serves as an introductory implementation for a voice-driven restaurant assistant.

3.3 Implementation

This section provides a comprehensive simulation of the navigation system, as well as the object recognition model and the speech recognition model.

3.3.1 Simultaneous localization and mapping (SLAM)

We start by setting up the operating system, which is Linux. The version we use is 20.04. After that, we need to set up the ROS (Robot Operating System). After that, we set up the necessary packages to run the simulation, such as ROS Tool Box, Adaptive Monte Carlo Localization (AMCL), and others.

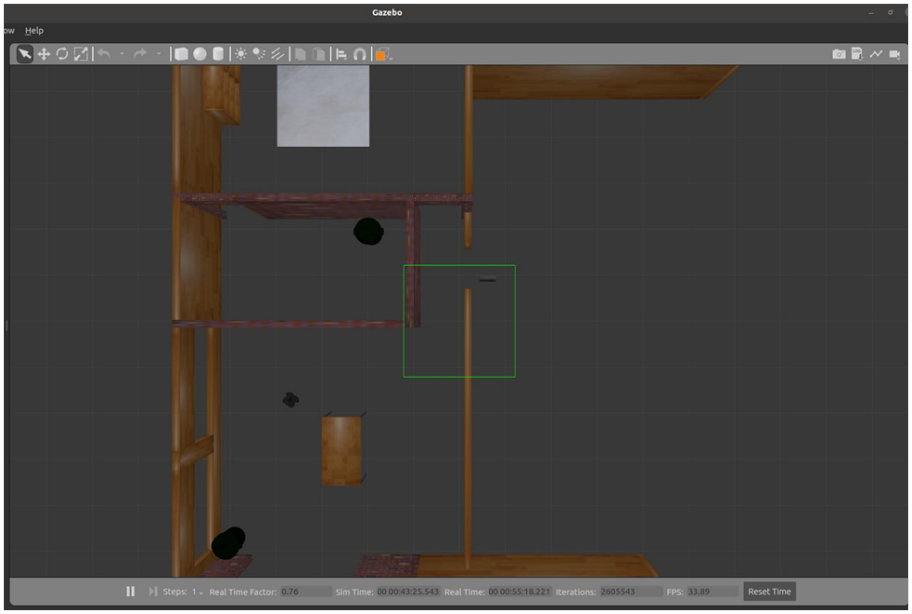


Figure 3.3: Gazebo Indoor Image

The Gazebo simulator in Fig. 3.3 is used to make the indoor area. Then we could run the maps with the Gazebo Simulator. Then the Rviz generates the map.

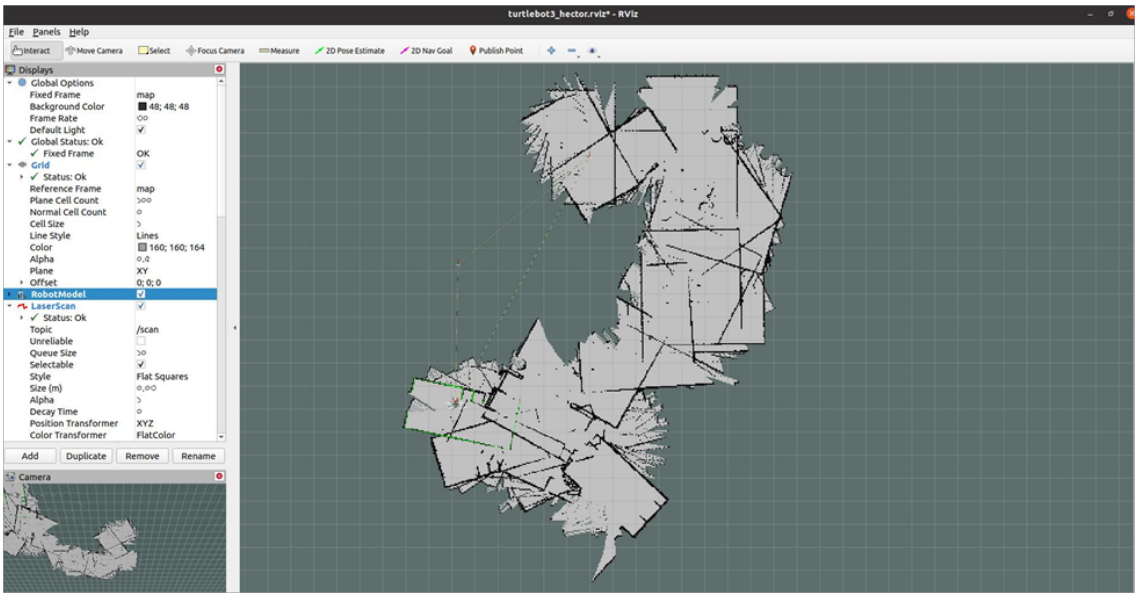


Figure 3.4: Hector Slam

We can see that the white part is in Fig. 3.4 the occupied place, those black points are unoccupied places meaning there are objects, and the rest of the blue part is an unknown area for the robot. The data that the Lidar camera collected is shown in bright green, and the landscape map that has been made is a light gray color that we can see in the Rviz. As more of the map is filled in, the Lidar scan area slowly changes from a light gray color to a white color.

3.3.2 Localization

In the localization part a particle filter can help a restaurant robot figure out where it is and how to get around in the restaurant. It state includes where it is and how it is facing. The particle filter helps guess these things by using readings from sensors like cameras, lidars, and other environmental sensors.

Particle Filter Algorithm

A particle filter is a type of recursive Bayesian filter that is often used to estimate the state of systems that are dynamic and whose state changes over time. It works especially well when the distributions are not linear or Gaussian. The particle filter in Fig 3.5 uses a group of particles, each of which is a guess about the current state of the system, to show what people think about its state. These particles move around over time and are changed based on sensor readings to get a better idea of what is going on with the system

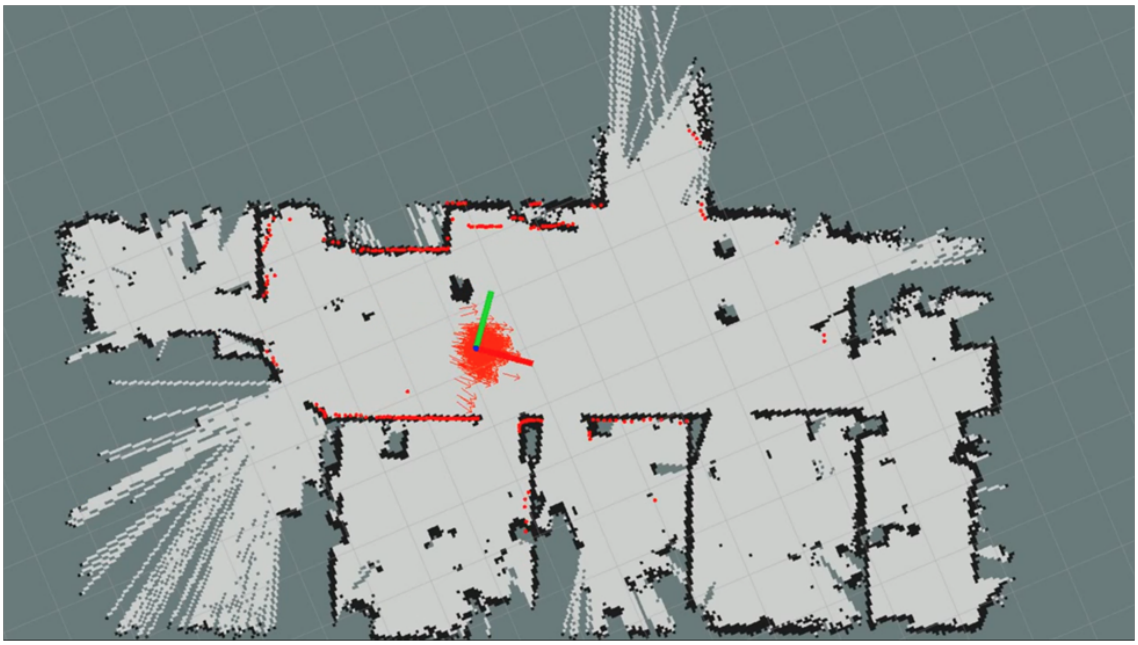


Figure 3.5: Localization

The filter starts with a base set of particles, each of which represents a possible state of the system. The particles are moved through the dynamic model of the system to figure out what their future states will be. Particles are given weights based on how well their predicted states fit the actual measurements when new sensor measurements come in. The filter copies the particles with higher weights because they are more likely to be correct. It gets rid of the particles with lower weights. This process of resampling makes sure that the particles better show the state of the system. The process is done over and over, and each time the guess of the systems state gets better.

A particle filter can be used to figure out where the robot is right now, for instance if it has a picture of the restaurant and sensors that can find landmarks or recognize specific spots. As the robot moves and gets data from its sensors, the filter will change what it thinks about its location.

3.3.3 Path Planning

The environment map was acquired and then imported into the ROS framework for route planning. To simulate routes and use the ROS framework, we employ the Rviz program. This in Fig. 3.6 on the left, shows the finished map. Also, the cyan area shows how far away the robot should be from the obstruction, while the black dot shows that there is an obstacle.

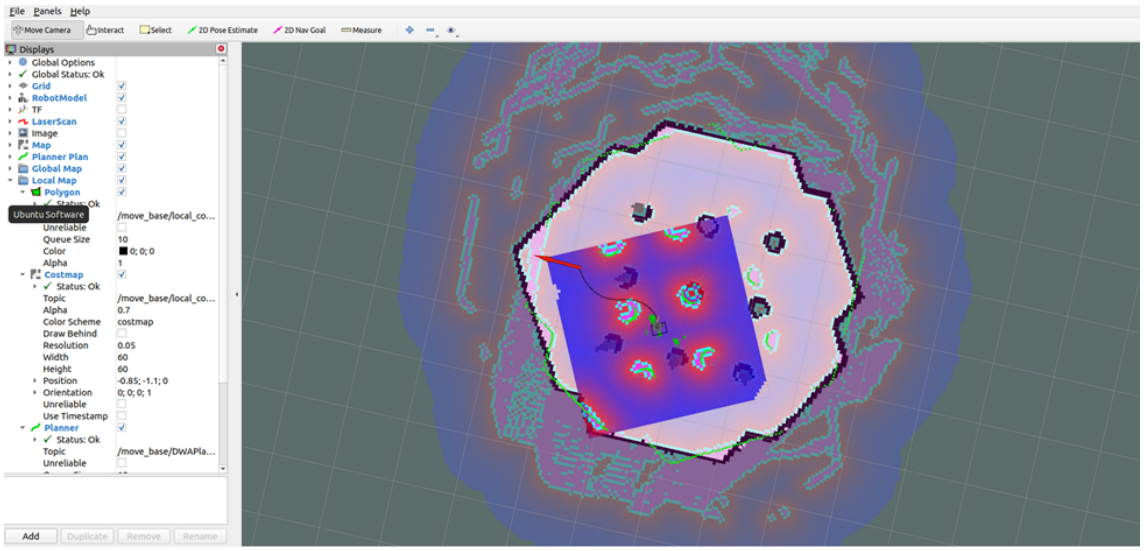


Figure 3.6: Path planning

We can see the red arrow indicates the robots intended path and destination location. The red line in Fig. 3.7 using DWA(Dynamic Window Approach) and A* algorithm get the results of the path planning and navigation process, which are located on the right side of the figure.

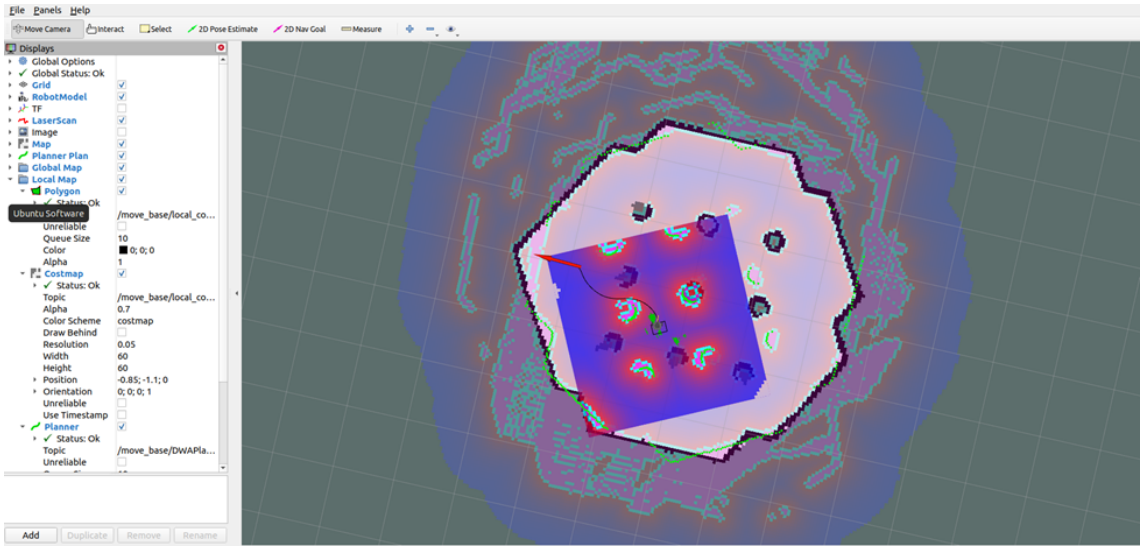


Figure 3.7: The bot reached to the target node

The robot will begin its journey in Fig 3.7 just below the obstacle. The virtual route not only bypasses the obstruction with ease, but it also follows a very straight line. Finally, the bot reached the target node.

3.3.4 Object Recognition

The Faster R-CNN object recognition model helps a robot quickly find lost things by correctly identifying and classifying objects in its environment. It quickly finds areas of interest with its region suggestion network, which ensures accurate localization.

The ability of the model to give users specific information about accessories, like where they are and what they do, makes it even more useful. Faster R-CNN is more accurate than SSD MobileNetV3 because its two-stage architecture allows for more accurate localization and fewer false positives. This makes it a better choice for jobs that need very accurate object recognition and localization.

Faster R-CNN

Faster R-CNN is a deep learning model created by Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. (2015) to help find objects. It is based on the Region-based Convolutional Network (R-CNN) design and aims to make object detection faster and more accurate. In our object identification model represented in Fig 3.8 we utilize the coco names dataset combined with our own custom data to enhance the accuracy of the object recognition model.It is evident that the system can accurately identify all the parts with improved precision.

```
import torch
import torchvision
from torchvision import transforms as T
from PIL import Image
import cv2
from google.colab.patches import cv2_imshow

# Load pre-trained FasterRCNN model
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Load and transform the image
img_path = "/content/table.jpeg"
img = Image.open(img_path)
transform = T.ToTensor()
img_tensor = transform(img)

# Make prediction with the model
with torch.no_grad():
    pred = model([img_tensor])

# Get bounding boxes, labels, and scores
bboxes, labels, scores = pred[0]["boxes"], pred[0]["labels"], pred[0]["scores"]
```

Figure 3.8: Object recognition code

The keyboard and cell phone represent the highest levels of accuracy in Fig 3.9 with percentages of 0.96 and 0.93 respectively. In addition, the mouse, cup, and table represent accuracy of 0.54 , 0.59 , and 0.56 percent correspondingly.

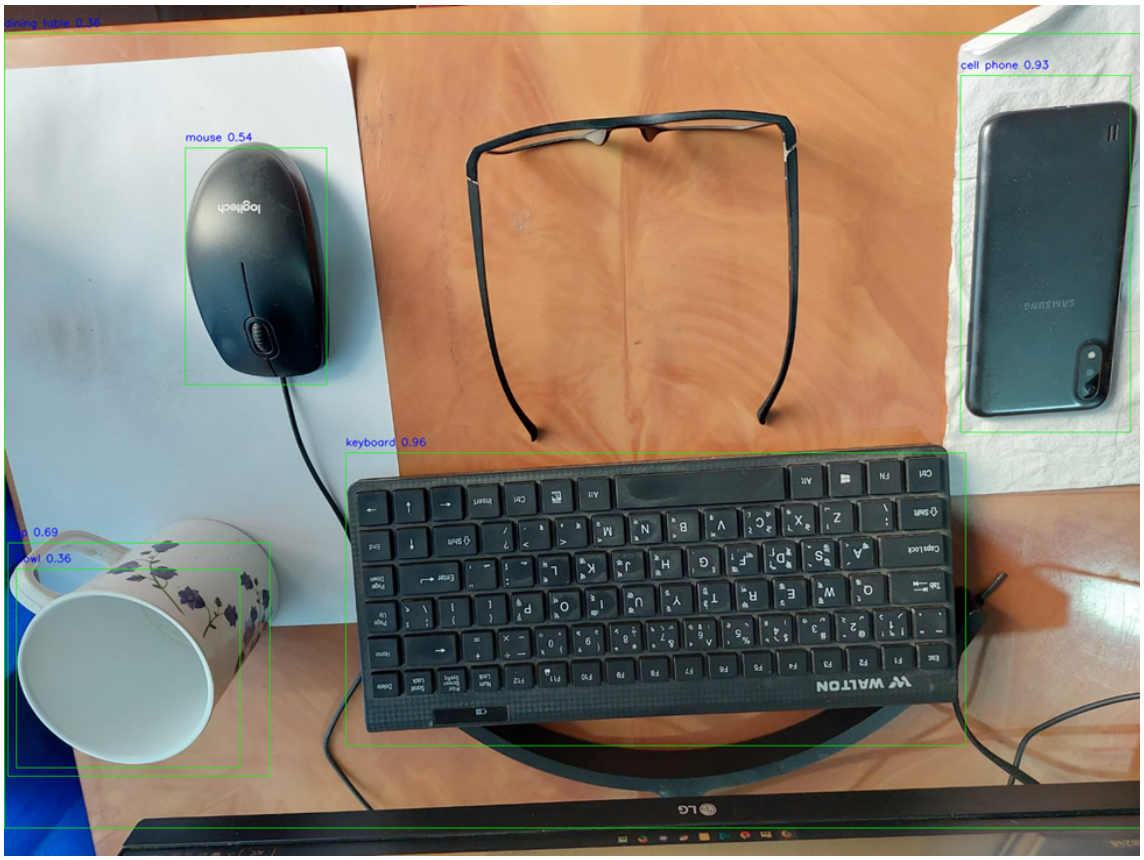


Figure 3.9: Object recognition using FasterR-CNN

The accuracy of the phone in Fig 3.10 is 0.91, the accuracy of the cup is 0.92, and the detection percentage of the person is 1.00 percent. These are the results that the object detection model shown in Figure 3.10 is able to determine.

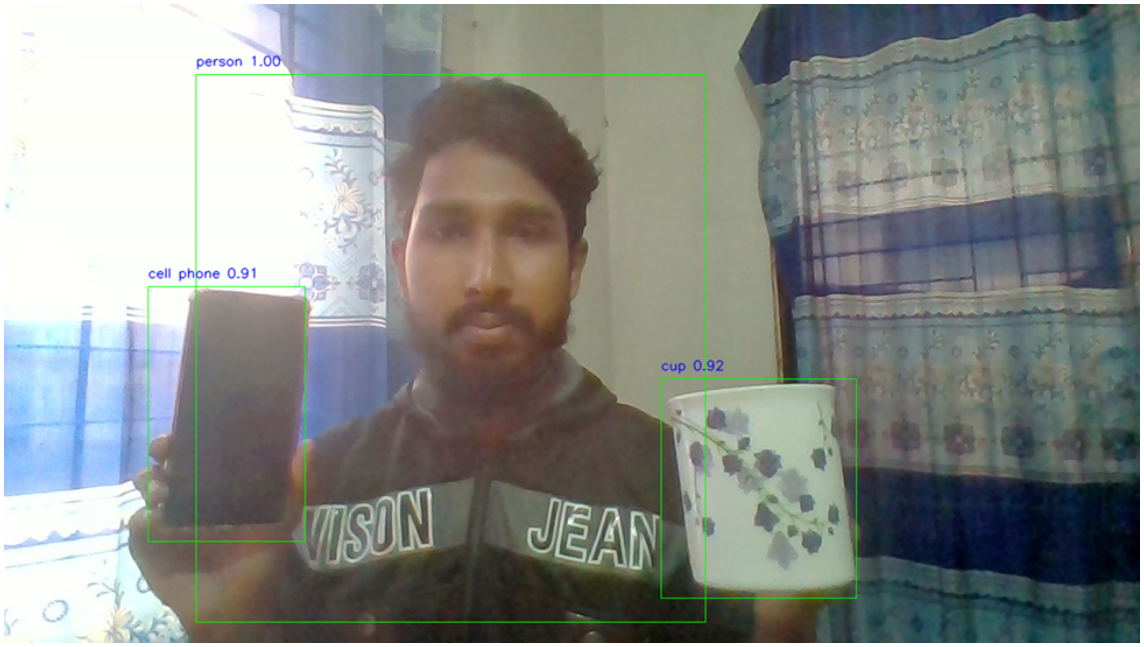


Figure 3.10: Better accuracy using Faster-R-CNN

Overall we get better accuracy than the ssd mobilenet v3 model and the model is also much faster than the others. Accuracy is raised as a result of improvements in image quality. We selected the model based on its fast and better accuracy.

SSD MobileNet V3

In the case of an automated restaurant robot, SSD MobileNet V3 is used to find and identify objects. It works by turning a picture you give it into a grid and guessing multiple bounding boxes and the class scores that go with them for each grid cell at the same time. The design of MobileNet V3 helps keep the model small and fast, which makes it good for real-time use on devices with limited resources, like robots.

The robot uses cameras to take pictures of things around it, like tables, chairs, and other things.SSD MobileNet V3 looks at these pictures and figures out what things are in them, like people, tables, or barriers.The model predicts where each item it finds will be by drawing a box around it.

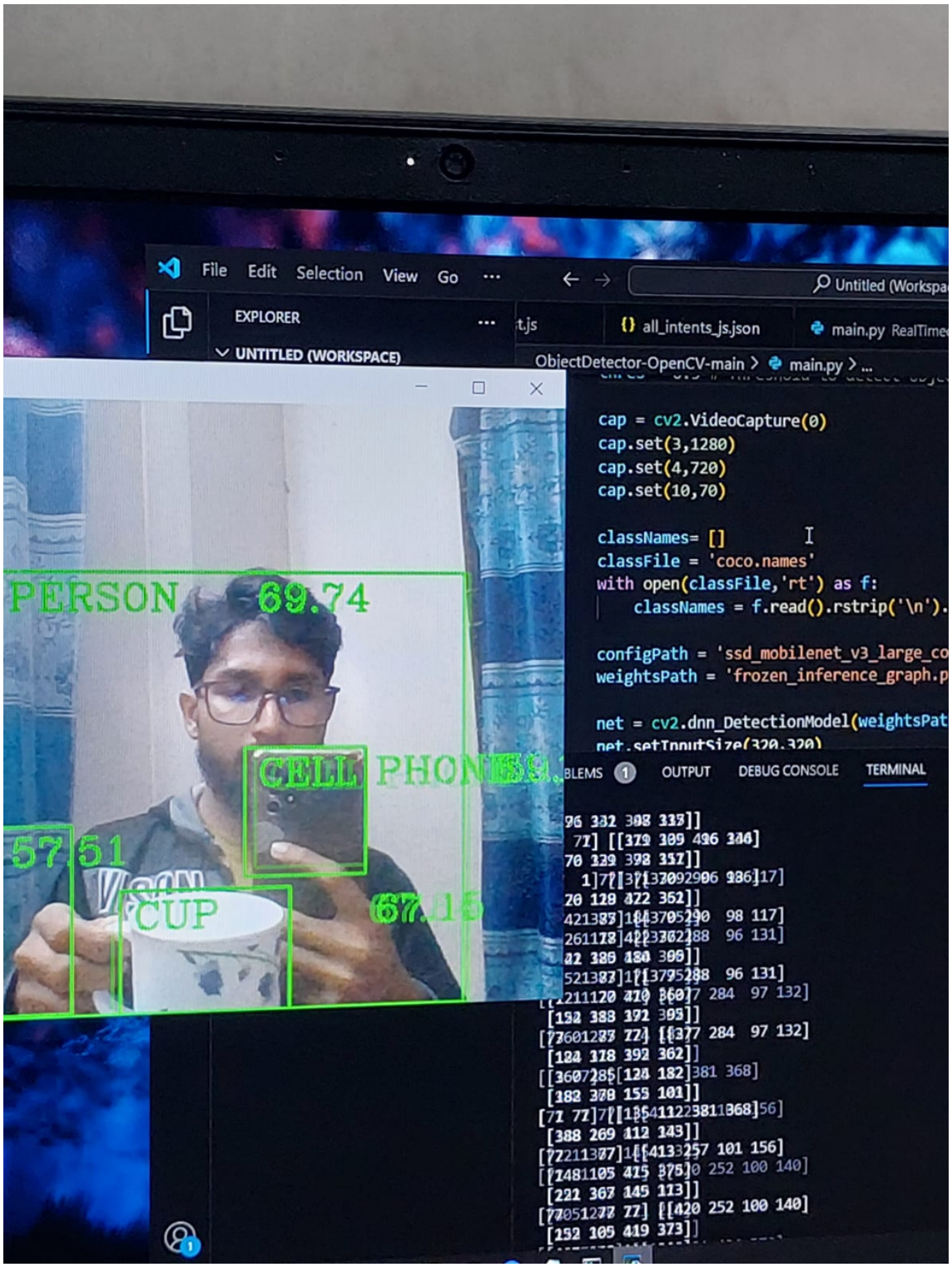


Figure 3.11: Object Recognition using ssd mobilenet v3

The model gives each bounding box a class score that shows in Fig 3.11 how likely it is that the object belongs to a certain class, like “table,” “cup,” or “human.” Moreover, the model can recognise the person with a higher accuracy.

Figure 3.12: use more objects for performance test

We use more object to test the performance of our ssd mobile net v3 model and get the accuracy result in Fig 3.12 . We can see that it can detect cup, keyboard and mouse with highest accuracy result. Moreover, the ssd mobile net v3 model a few lower than the Faster-R-CNN model. The Faster-R-CNN model also faster than the ssd mobile net v3 model.

3.3.5 SpeechRecognition

The given code utilizes the Google Web Speech API in Fig 3.13 to perform speech recognition. This Speech Recognition model will help our robot to interact with the customer more efficiently.

pyttsx3 and speechrecognition libraries are imported. The text-to-speech engine is initialized using the 'sapi5' backend, and the voice is set.

Figure 3.13: Speech Recognition code

Here two functions are defined. One is speak(audio) for text-to-speech and the other is takeCommand() for speech recognition.

The takeCommand() function captures audio input through the microphone, prints "Listening..." to indicate that it is ready, and listens for user input. The recognized speech is printed, and the function returns the recognized query. The getintent(query) function matches the recognized query with predefined intents in the data list. It uses a simple scoring mechanism based on word matching to determine the best-matched intent. The response (intent) function generates appropriate responses based on the matched intent.

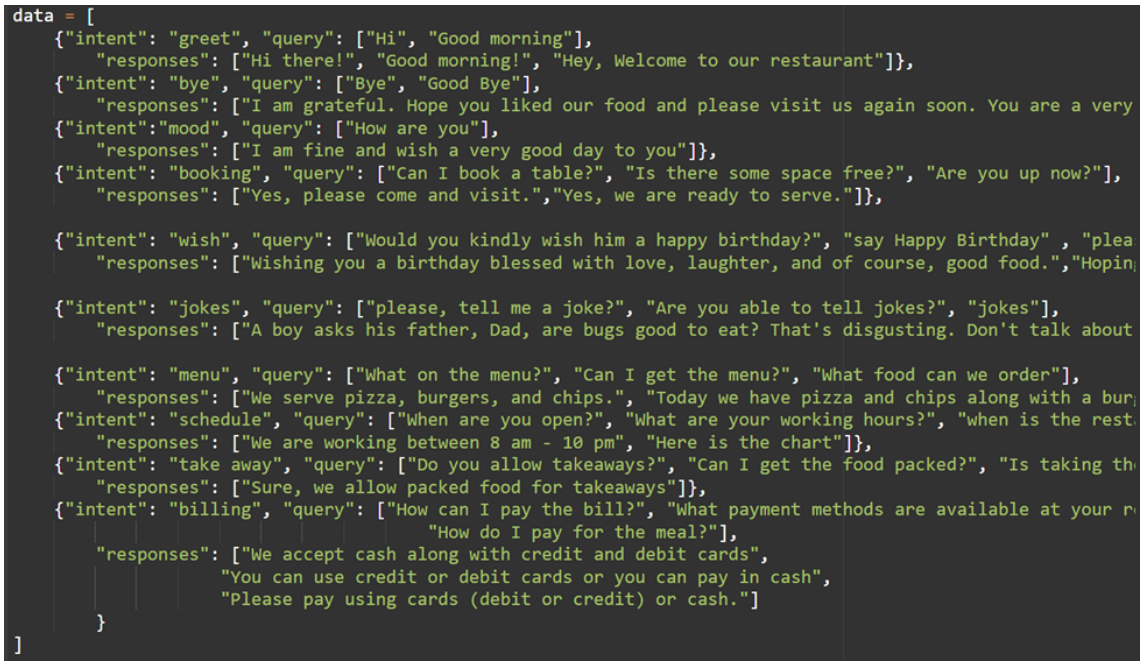


Figure 3.14: Main Conversation Dataset

The waiter bot has the ability to generate all the necessary information such as the greetings, payment method in Fig 3.14, the opening and closing time of the restaurant, all the available food items etc.

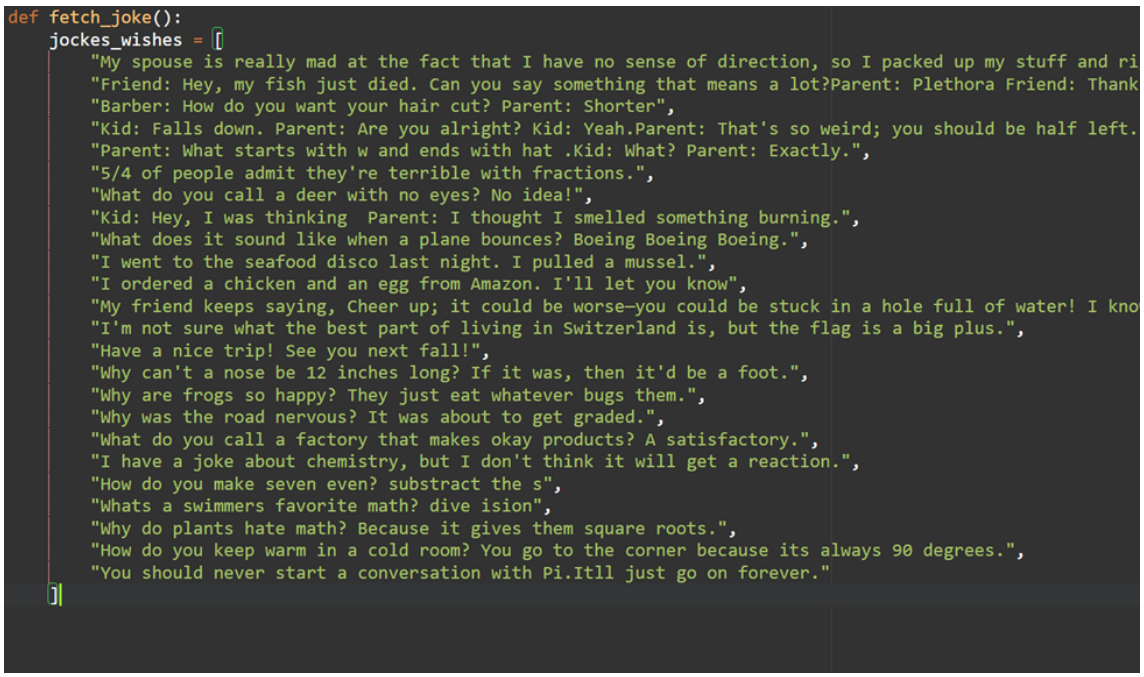


Figure 3.15: Jokes Dataset

For “jokes” and “wish” intents, random jokes in Fig 3.15 and birthday wishes in Fig 3.16 are fetched from dedicated functions (fetchjoke() and fetchbirthdaywish()). For other intents, random responses from the predefined list are selected. Data Definitions: The data list contains dictionaries for each intent, specifying associated

queries and responses. The main loop continuously listens for user input, determines the intent, and responds accordingly.

```
def fetch_birthday_wish():
    birthday_wishes = [
        "Happy Birthday! May your day be filled with lots of love and happiness.",
        "Wishing you a very Happy Birthday! Enjoy your special day.",
        "Happy Birthday! Here's to another year of success and happiness.",
        "Wishing you a day filled with love and cheer. Happy Birthday!",
        "May your birthday be as amazing as you are. Happy Birthday!",
        "Happy Birthday! May all your dreams come true in the coming year.",
        "Wishing you a day that is as special as you are. Happy Birthday!",
        "Happy Birthday! May your day be as bright as your smile.",
        "Wishing you a very Happy Birthday! May your day be as wonderful as you are.",
        "Here's to celebrating you! Happy Birthday!",
        "Happy Birthday! Wishing you a day filled with happiness and a year filled with joy.",
        "Sending you smiles for every moment of your special day. Have a wonderful time and a
        "Hope your special day brings you all that your heart desires! Happy Birthday!",
        "On your birthday, I wish you happiness and joy for this special day. Happy Birthday!"
        "May your birthday be filled with sunshine and smiles. Happy Birthday!",
        "Wishing you a day filled with happiness and a year filled with joy. Happy Birthday!",
        "Sending you best wishes for success, health, and good fortune today and in the year t
        "Enjoy your day and may God bless you with many more years! Happy Birthday!",
        "Happy Birthday, wishing you a day filled with happiness and a year filled with joy.",
        "Sending you warm wishes and lots of love on your birthday. Happy Birthday!"
    ]
```

Figure 3.16: Birthday Greeting Dataset

The Speech Recognition model is being utilized in our model in an effort to develop the interaction between the customer and the waiter robot. We have put into action the primary discussion in which the waiter is responsible for generating all of the information regarding the restaurant, including the opening and closing times, the meal items, the payment system, greetings, and further information.

Overall, the model forms a conversational interface where the restaurant robot can engage in simple interactions, providing information, jokes, birthday wishes, and responding to various user queries. This feature will help to improve the customer waiter interaction more enjoyable

Chapter 4

Result Analysis

All of the models, including navigation, object identification, and speech recognition, are compared in this section, and the results are displayed according to their performance.

4.1 Mappping

In this section on mapping, we analyze and compare various algorithms that produce maps. We also evaluate their performance to determine the most suitable option for our research. The choice between these SLAM algorithms depends on the specific requirements and constraints of the robotic application in a restaurant setting.

We can observe that the map that was formed is referred to as (a) gmapping here in Fig 4.1 The white part is the occupied place, those dotted black points are unoccupied places meaning there are objects, and the rest of the green part is an unknown area for the robot. Gmapping is a simple and efficient SLAM algorithm for robotic mapping, suitable for robots with limited processing power. With a particle filter-based approach, it handles uncertainty in pose estimation, making it robust. Gmapping supports map merging techniques, making it suitable for collaborative mapping in multi-robot systems. Gmapping is mostly made for 2D mapping, and it might not work straight with 3D mapping without some changes. It might react badly to changes in its surroundings, and moving parts could affect how well it works [10].

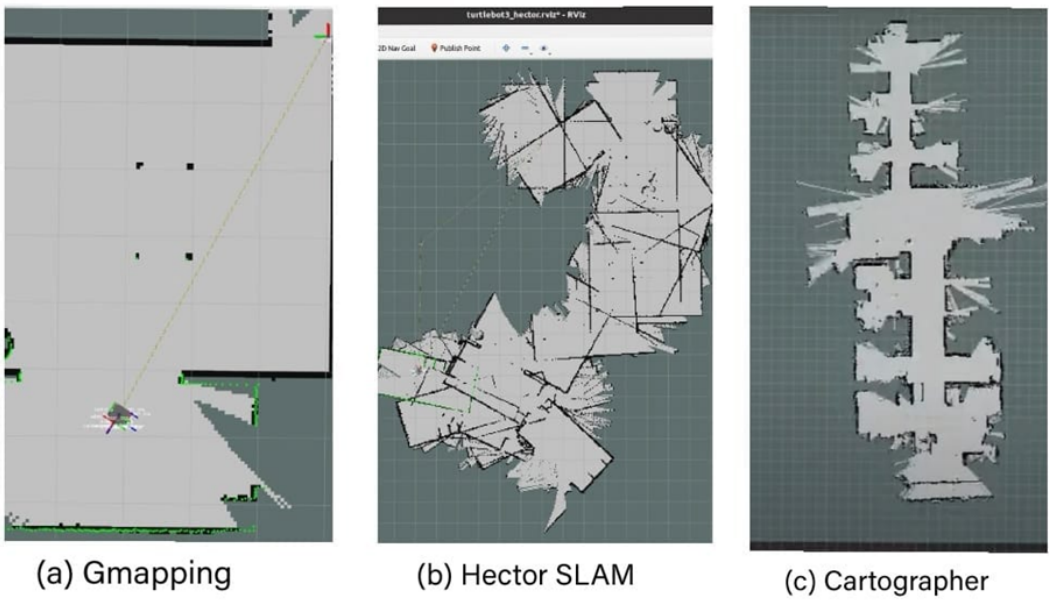
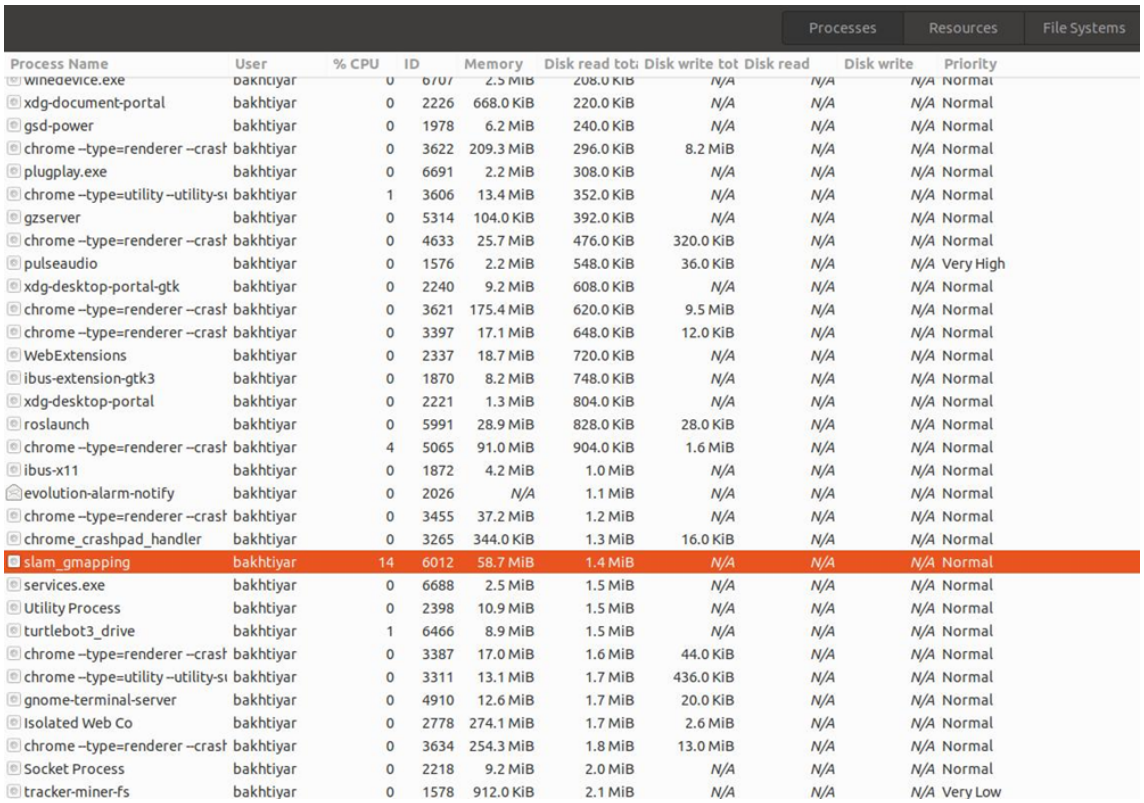


Figure 4.1: Mapping comparison

The third image in Fig 4.1 is referred to as (c) Cartographer Slam. This particular map offers a higher level of clarity compared to other maps. But It needs the most complex set up and also the computational complexities are very high. It is possible to make thorough 3D maps with Cartographer, which makes it useful in situations that need it. The algorithm is made to handle loop closing well, which improves the accuracy of mapping when the robot goes back to an area. Cartographer allows multi-sensor fusion, which lets the robot combine data from different sensors to get a better picture. Setting up a Cartographer to work in a certain environment can be hard, and it might need to modify some settings, which could make quick launch harder. Cartographers can use a lot of resources, which could be a problem for a restaurant computer that can not do a lot of computing [49].

In addition, we can see that Fig 4.1 shows that the produced map is known as (b) Hector Slam and the black part that actually the objects and the green parts are unknown areas. Here we can see the map generated more clearly than the previous map. Hector-SLAM works very well in real time, which means it can be used in situations where the robot needs to work quickly and efficiently. The algorithm can be set up quickly, which means that the robot can start moving right away after being released.

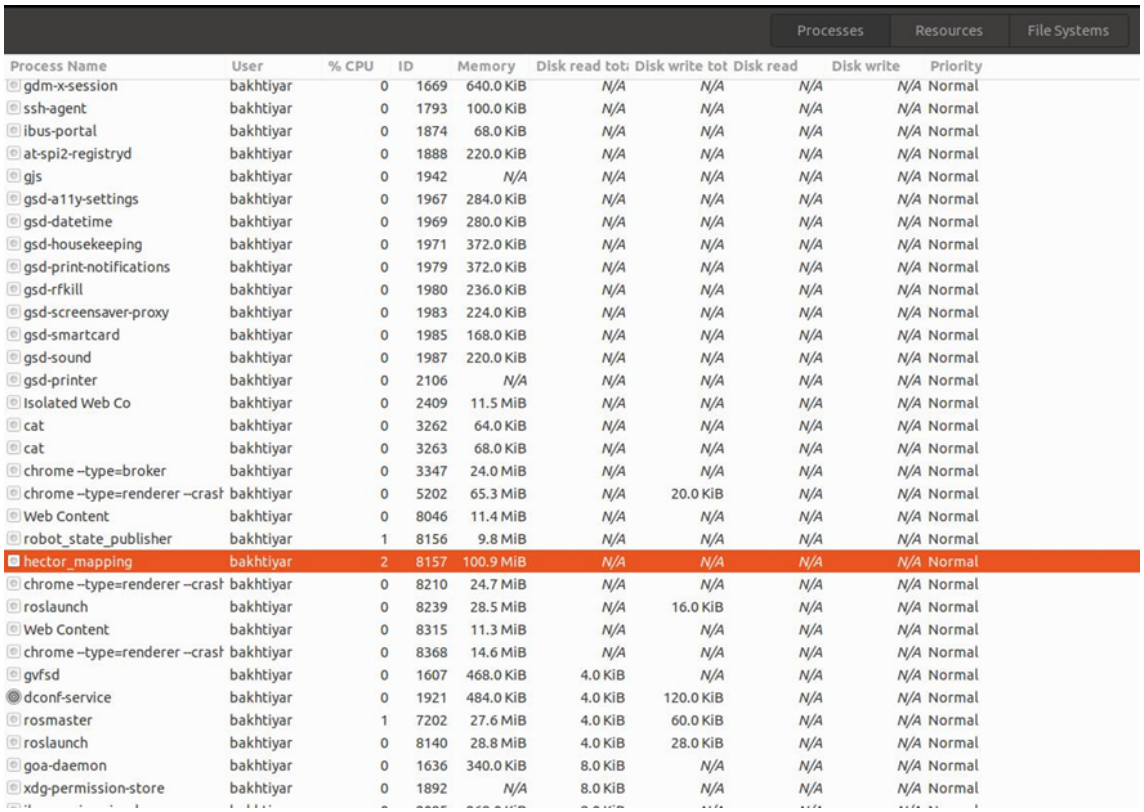


Processes | Resources | File Systems

Process Name	User	% CPU	ID	Memory	Disk read tot	Disk write tot	Disk read	Disk write	Priority
winedevice.exe	bakhtiyar	0	6707	2.5 MiB	208.0 KiB	N/A	N/A	N/A	Normal
xdg-document-portal	bakhtiyar	0	2226	668.0 KiB	220.0 KiB	N/A	N/A	N/A	Normal
gsd-power	bakhtiyar	0	1978	6.2 MiB	240.0 KiB	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3622	209.3 MiB	296.0 KiB	8.2 MiB	N/A	N/A	Normal
plugplay.exe	bakhtiyar	0	6691	2.2 MiB	308.0 KiB	N/A	N/A	N/A	Normal
chrome --type=utility --utility-su	bakhtiyar	1	3606	13.4 MiB	352.0 KiB	N/A	N/A	N/A	Normal
gzserver	bakhtiyar	0	5314	104.0 KiB	392.0 KiB	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	4633	25.7 MiB	476.0 KiB	320.0 KiB	N/A	N/A	Normal
pulseaudio	bakhtiyar	0	1576	2.2 MiB	548.0 KiB	36.0 KiB	N/A	N/A	Very High
xdg-desktop-portal-gtk	bakhtiyar	0	2240	9.2 MiB	608.0 KiB	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3621	175.4 MiB	620.0 KiB	9.5 MiB	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3397	17.1 MiB	648.0 KiB	12.0 KiB	N/A	N/A	Normal
WebExtensions	bakhtiyar	0	2337	18.7 MiB	720.0 KiB	N/A	N/A	N/A	Normal
ibus-extension-gtk3	bakhtiyar	0	1870	8.2 MiB	748.0 KiB	N/A	N/A	N/A	Normal
xdg-desktop-portal	bakhtiyar	0	2221	1.3 MiB	804.0 KiB	N/A	N/A	N/A	Normal
roslaunch	bakhtiyar	0	5991	28.9 MiB	828.0 KiB	28.0 KiB	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	4	5065	91.0 MiB	904.0 KiB	1.6 MiB	N/A	N/A	Normal
ibus-x11	bakhtiyar	0	1872	4.2 MiB	1.0 MiB	N/A	N/A	N/A	Normal
evolution-alarm-notify	bakhtiyar	0	2026	N/A	1.1 MiB	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3455	37.2 MiB	1.2 MiB	N/A	N/A	N/A	Normal
chrome_crashpad_handler	bakhtiyar	0	3265	344.0 KiB	1.3 MiB	16.0 KiB	N/A	N/A	Normal
slam_gmapping	bakhtiyar	14	6012	58.7 MiB	1.4 MiB	N/A	N/A	N/A	Normal
services.exe	bakhtiyar	0	6688	2.5 MiB	1.5 MiB	N/A	N/A	N/A	Normal
Utility Process	bakhtiyar	0	2398	10.9 MiB	1.5 MiB	N/A	N/A	N/A	Normal
turtlebot3_drive	bakhtiyar	1	6466	8.9 MiB	1.5 MiB	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3387	17.0 MiB	1.6 MiB	44.0 KiB	N/A	N/A	Normal
chrome --type=utility --utility-su	bakhtiyar	0	3311	13.1 MiB	1.7 MiB	436.0 KiB	N/A	N/A	Normal
gnome-terminal-server	bakhtiyar	0	4910	12.6 MiB	1.7 MiB	20.0 KiB	N/A	N/A	Normal
Isolated Web Co	bakhtiyar	0	2778	274.1 MiB	1.7 MiB	2.6 MiB	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	3634	254.3 MiB	1.8 MiB	13.0 MiB	N/A	N/A	Normal
Socket Process	bakhtiyar	0	2218	9.2 MiB	2.0 MiB	N/A	N/A	N/A	Normal
tracker-miner-fs	bakhtiyar	0	1578	912.0 KiB	2.1 MiB	N/A	N/A	N/A	Very Low

Figure 4.2: Gmapping memory use

Gmapping use less memory than hector slam. We can see from the image in Fig 4.2 that gmapping uses 58.7 mb memory. Moreover hector slam uses a lot memory in Fig 4.3 we can see that 100.9 mb memory used by hector slam.



Processes | Resources | File Systems

Process Name	User	% CPU	ID	Memory	Disk read tot	Disk write tot	Disk read	Disk write	Priority
gdm-x-session	bakhtiyar	0	1669	640.0 KiB	N/A	N/A	N/A	N/A	Normal
ssh-agent	bakhtiyar	0	1793	100.0 KiB	N/A	N/A	N/A	N/A	Normal
ibus-portal	bakhtiyar	0	1874	68.0 KiB	N/A	N/A	N/A	N/A	Normal
at-spi2-registryd	bakhtiyar	0	1888	220.0 KiB	N/A	N/A	N/A	N/A	Normal
gjs	bakhtiyar	0	1942	N/A	N/A	N/A	N/A	N/A	Normal
gsd-a11y-settings	bakhtiyar	0	1967	284.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-datetime	bakhtiyar	0	1969	280.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-housekeeping	bakhtiyar	0	1971	372.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-print-notifications	bakhtiyar	0	1979	372.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-rfkill	bakhtiyar	0	1980	236.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-screensaver-proxy	bakhtiyar	0	1983	224.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-smartcard	bakhtiyar	0	1985	168.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-sound	bakhtiyar	0	1987	220.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-printer	bakhtiyar	0	2106	N/A	N/A	N/A	N/A	N/A	Normal
Isolated Web Co	bakhtiyar	0	2409	11.5 MiB	N/A	N/A	N/A	N/A	Normal
cat	bakhtiyar	0	3262	64.0 KiB	N/A	N/A	N/A	N/A	Normal
cat	bakhtiyar	0	3263	68.0 KiB	N/A	N/A	N/A	N/A	Normal
chrome --type=broker	bakhtiyar	0	3347	24.0 MiB	N/A	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	5202	65.3 MiB	N/A	20.0 KiB	N/A	N/A	Normal
Web Content	bakhtiyar	0	8046	11.4 MiB	N/A	N/A	N/A	N/A	Normal
robot_state_publisher	bakhtiyar	1	8156	9.8 MiB	N/A	N/A	N/A	N/A	Normal
hector_mapping	bakhtiyar	2	8157	100.9 MiB	N/A	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	8210	24.7 MiB	N/A	N/A	N/A	N/A	Normal
roslaunch	bakhtiyar	0	8239	28.5 MiB	N/A	16.0 KiB	N/A	N/A	Normal
Web Content	bakhtiyar	0	8315	11.3 MiB	N/A	N/A	N/A	N/A	Normal
chrome --type=renderer --crash	bakhtiyar	0	8368	14.6 MiB	N/A	N/A	N/A	N/A	Normal
gvfsd	bakhtiyar	0	1607	468.0 KiB	4.0 KiB	N/A	N/A	N/A	Normal
dconf-service	bakhtiyar	0	1921	484.0 KiB	4.0 KiB	120.0 KiB	N/A	N/A	Normal
rosmaster	bakhtiyar	1	7202	27.6 MiB	4.0 KiB	60.0 KiB	N/A	N/A	Normal
roslaunch	bakhtiyar	0	8140	28.8 MiB	4.0 KiB	28.0 KiB	N/A	N/A	Normal
goa-daemon	bakhtiyar	0	1636	340.0 KiB	8.0 KiB	N/A	N/A	N/A	Normal
xdg-permission-store	bakhtiyar	0	1892	N/A	8.0 KiB	N/A	N/A	N/A	Normal

Figure 4.3: Hector slam memory use

Although, gmapping use less memory than hector slam but gmapping can generate the clear map and also slower than hector slam. Hector-SLAM might use a lot of memory, which could be a problem for but providing better accessories, downsample sensor data and adjusting parameter setting can solve this problem [49].

For our restaurant robot model hector slam can generate a better map than gmapping and also it could perform very well in a dynamic environment with moving objects. Hector slam is not that complex and also does not require more resources like cartographer. Moreover, it quickly generates maps more than any other algorithm.

4.2 Path Planning

The path planning process for an autonomous restaurant robot includes finding the best way to get from where it is now to a certain location in the restaurant. It looks at things like obstacles, changes in the environment, and the skills of robots to find a path that works well and does not hit anything. This process makes it easier for the robot to move around, serve orders, and do other tasks on its own, which makes it safer and more useful in a busy restaurant.

A* algorithm and Dijkstra algorithm comparison shows in Fig 4.4 This section focuses on comparing the A* algorithm and the Dijkstra algorithm in the context of mapping. Additionally, we analyze their respective performance to determine better one for our research.

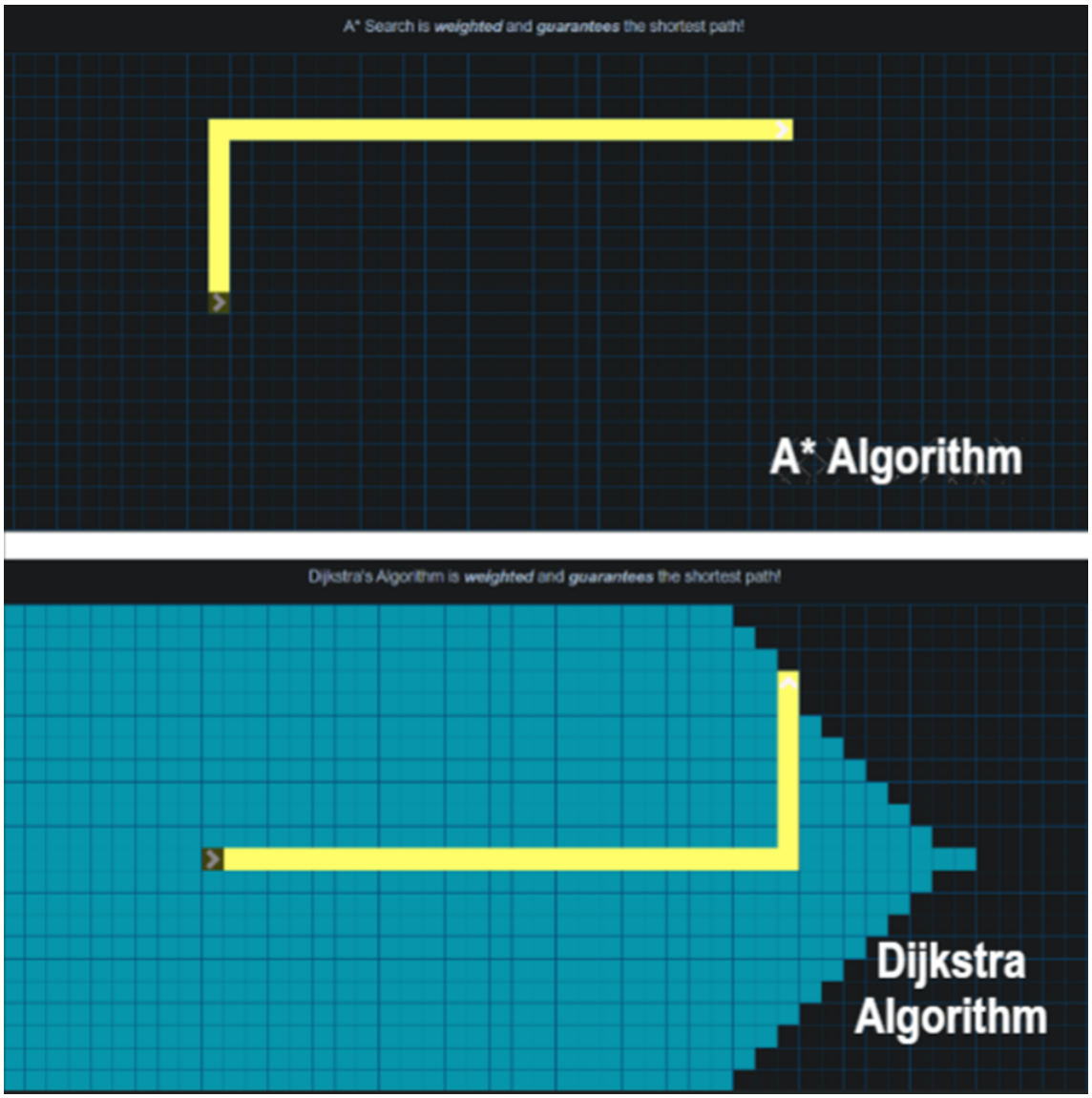


Figure 4.4: Comparison between A* Algorithm and Dijkstra Algorithm

Dijkstra Algorithm is acceptable, but it does not reflect reality. It promises that the solution is optimal without using heuristics. A* and dijkstra algorithm in Fig 4.2 are the most popular for path planning algorithms. In A* algorithm heuristic knowledge is used to help the search go quickly. The dijkstra algorithm does not use any heuristic data. It looks for paths based only on edge weights. A* algorithm uses heuristic information to put more promising paths at the top of the list, which narrows down the search area. The dijkstra algorithm may not be as good at exploring, especially for bigger graphs that do not have clear rules. A*-based method by focusing on ways that are most likely to lead to the goal, it tends to be more efficient. The dijkstra algorithm Usually needs less memory than the A* method. A*-based method might need more memory, especially when there are a lot of nodes. O((V+E)log V) is the time complexity of dijkstra algorithm where v is vertex and E means edge and the time complexity is A* algorithm is $O(b^d)$ where b is the branching factor. So, comparatively A* algorithm is faster than dijkstra algorithm [52]. To consider all above the fact, we choose A* algorithm to our model.

4.3 Object Detection and Recognition

This section focuses on comparing the object recognition model in the context of fast and better accuracy. Additionally, we analyze their respective performance to determine better one for our research.

4.3.1 Faster-R-CNN comparison and SSD MobileNet V3

In the beginning, we built ssd mobile net v3 and used the most famous coco names dataset. It is able to identify the things. This is then compared to the Faster-R-CNN model using the same set of data. The Faster-R-CNN model gives us better precision.

Object	SSD_MobileNet_V3	Faster-R-CNN
Person	69.74	1.00
Cell Phone	69.33	0.93
Cup	67.16	0.92
Keyboard	72.33	0.96
Mouse	64.21	0.54

Table 4.1: Comparison between SSD MobileNet V3 and Faster-R-CNN

Based on the table above in Fig 4.3 we can see that person has the best accuracy in the FasterRCNN model at 1.00 percent, while mouse has the lowest accuracy at 0.54 percent. The keyboard is the most accurate in the SSD MobileNet V3 model, at 72.33 percent, while the mouse is the least accurate, at 64.21 percent. Another thing is that the cup gets 67.16 percent accuracy in the SSD Mobile Net V3 model and 0.9 percent accuracy in the FasterRCNN model. If we look at cell phones again, the SSD mobile net v3 model gets 69.33 percent accuracy and the FasterRCNN model gets 0.93 percent accuracy. In the Faster-R-CNN model, all of the parts are more accurate. Plus, it is faster than SSD Mobile Net V3. So, we picked this Faster-R-CNN model for our research.

4.4 SpeechRecognition

In our model for speech recognition, the Google Web Speech API is used for the given code. Depending on the underlying speech recognition engine, speech recognition performance and accuracy can change. The CMU Sphinx, commonly referred to as PocketSphinx, is another well-liked voice recognition engine.

```
SpeechRecognition > orderinteractvoice.py > ...
1   import pyttsx3
2   import speech_recognition as sr
3   import requests
4   import random
5
6   engine = pyttsx3.init('sapi5')
7   voices = engine.getProperty('voices')
8   engine.setProperty('voice', voices[0].id)
9
10  def speak(audio):
11      engine.say(audio)
12      engine.runAndWait()
13
14  def takeCommand():
15      r = sr.Recognizer()
16      with sr.Microphone() as source:
17          print("Listening...")
18          r.pause_threshold = 1
19          audio = r.listen(source)
20      try:
21          print("Recognizing...")
22          query = r.recognize_google(audio, language='en-in')
23          print("You:", query)
24          return query
25      except Exception as e:
26          print("Say that again, please...")
27          return "None"
28
```

Figure 4.5: Google Speech API

The Google Web Speech API in Fig 4.4 is known for its high accuracy, especially in processing natural language. The cloud-based voice recognition solution is highly robust. Additionally, it is compatible with several languages. Using the speech recognition library in python is straightforward. As it relies on Google based cloud services, an internet connection is required. The Google Speech API Programming Interface (API) allows only for the enforcement of the available rate limitations. As audio is sent to googles servers for processing, there is a potential for privacy concerns.

```
from pocketsphinx import LiveSpeech

def takeCommand():
    speech = LiveSpeech()
    print("Listening...")

    for phrase in speech:
        audio = str(phrase)
        print("Recognizing...")
        print("You:", audio)
        return audio

speak("Hello, I am your restaurant assistant. How can I help you today?")

while True:
    query = takeCommand()
    intent = get_intent(query)
    respond(intent)
```

Figure 4.6: CMUSphinx

CMU Sphinx (PocketSphinx) is an embedded speech recognition system in Fig 4.5 that works even where the user is not online. It has a faster response time because it does not use a cloud-based tool. Once the system is set up, speech recognition does not need any outside help. It is not as accurate as cloud-based options most of the time, especially when it comes to conversational or complex language. Compared to cloud-based solutions, it does not handle as many languages. Needs more setup and may need to be customized to work best in some situations. Getting the Word Error Rate (WER) is the best way to compare the quality of different Automatic Speech Recognition (ASR) systems [53].

WER = (I + D + S) / N

Here I words for included, D words for cut out, and S words for put in where they belong. Google Speech API has a higher accuracy than the CMU Sphinx (PocketSphinx) model so we choose the model Google Speech API for our research.

File	The Final Results of Sphinx-4								
	S	N	I	S	D	CW	EW	WA	WER
TSX223	1	8	1	1	0	6	2	0.88	0.25
TSX293	1	11	0	2	2	7	4	0.64	0.36
TSI1894	1	9	0	2	0	7	2	0.78	0.22
TSI1400	1	14	1	3	0	10	4	0.79	0.29
TSX188	2	6	0	4	0	2	4	0.33	0.67
TSI1628	2	12	0	4	3	5	7	0.42	0.58
TSX314	2	12	0	2	2	8	4	0.67	0.33
DIG001	3	15	0	1	0	14	1	0.93	0.07
TSX216	1	9	0	1	0	8	1	0.89	0.11
TSX209	1	7	0	1	1	5	2	0.71	0.29
TSI1584	2	13	0	6	2	5	8	0.38	0.62
TSX371	1	11	0	6	0	5	6	0.45	0.55
TSI1373	1	14	0	7	3	4	10	0.29	0.71
TSX233	1	7	1	2	0	4	3	0.71	0.43
OSE003	1	8	1	3	0	4	4	0.63	0.5
AENGM8	1	9	0	1	0	8	1	0.89	0.11
AENGF8	1	9	0	5	1	3	6	0.33	0.67
AENGF7	1	6	0	1	0	5	1	0.83	0.17
AENGM2	1	7	0	1	0	6	1	0.86	0.14
Mean									0.37

Table 3. The Final Results of Sphinx-4

File	The Final Results of Microsoft Speech API								
	S	N	I	S	D	CW	EW	WA	WER
TSX223	1	8	0	1	0	7	1	0.88	0.13
TSX293	1	11	0	0	0	11	0	1.0	0.0
TSI1894	1	9	0	2	0	7	2	0.78	0.22
TSI1400	1	14	0	0	0	14	0	1.0	0.0
TSX188	2	6	0	6	0	0	0	0.0	0.1
TSI1628	2	12	0	8	2	2	10	0.17	0.83
TSX314	2	12	0	0	0	12	0	1.0	0.0
DIG001	3	15	0	0	0	15	0	1.0	0.0
TSX216	1	9	0	0	0	15	0	1.0	0.0
TSX209	1	7	0	1	1	5	2	0.71	0.29
TSI1584	2	13	0	5	2	6	7	0.46	0.54
TSX371	1	11	0	1	0	10	1	0.91	0.09
TSI1373	1	14	0	5	1	8	6	0.57	0.43
TSX233	1	7	0	2	0	5	2	0.71	0.29
OSE003	1	8	0	3	0	5	3	0.63	0.38
AENGM8	1	9	0	0	0	9	0	1.0	0.0
AENGF8	1	9	0	0	0	9	0	1.0	0.0
AENGF7	1	6	0	0	0	6	0	1.0	0.0
AENGM2	1	7	0	1	0	6	1	0.86	0.14
Mean									0.18

Table 4. The Final Results of Microsoft API

File	The Final Results of Google Speech API								
	S	N	I	S	D	CW	EW	WA	WER
TSX223	1	8	0	0	0	9	0	1.0	0.0
TSX293	1	11	0	1	1	9	2	0.82	0.18
TSI1894	1	9	0	0	0	9	0	1.0	0.0
TSI1400	1	14	0	1	0	13	1	0.93	0.07
TSX188	2	6	0	0	0	6	0	1.0	0.0
TSI1628	2	12	0	2	0	10	2	0.83	0.17
TSX314	2	12	0	0	0	12	0	1.0	0.0
DIG001	3	15	0	0	0	15	0	1.0	0.0
TSX216	1	9	0	0	0	9	0	1.0	0.0
TSX209	1	7	0	0	0	7	0	1.0	0.0
TSI1584	2	13	0	5	2	6	7	0.46	0.54
TSX371	1	11	0	0	0	11	0	1.0	0.0
TSI1373	1	14	0	0	0	14	0	1.0	0.0
TSX233	1	7	1	0	0	6	1	0.71	0.14
OSE003	1	8	0	2	1	5	3	0.63	0.38
AENGM8	1	9	0	0	0	9	0	1.0	0.0
AENGF8	1	9	0	2	0	7	2	0.78	0.22
AENGF7	1	6	0	0	0	6	0	1.0	0.0

Figure 4.7: Final results of three different API [53]

In their paper titled " Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx)", Këpuska, V., and Bohouta, G. examine several libraries such as Text to Speech API, Graph API and Math API for different tasks. Moreover, this tool was connected with the classes of Sphinx4, Microsoft API and Google API to work together to recognize the audio files in Fig 4.7 Then they compared the recognition results with the original recording texts [53].

File	Sphinx4		Google API		Microsoft API	
	WA	WER	WA	WER	WA	WER
TSX223	0.88	0.25	1.0	0.0	0.88	0.13
TSX293	0.64	0.36	0.82	0.18	1.0	0.0
TSI1894	0.78	0.22	1.0	0.0	0.78	0.22
TSI1400	0.79	0.29	0.93	0.07	1.0	0.0
TSX188	0.33	0.67	1.0	0.0	0.0	0.1
TSI1628	0.42	0.58	0.83	0.17	0.17	0.83
TSX314	0.67	0.33	1.0	0.0	1.0	0.0
DIG001	0.93	0.07	1.0	0.0	1.0	0.0
TSX216	0.89	0.11	1.0	0.0	1.0	0.0
TSX209	0.71	0.29	1.0	0.0	0.71	0.29
TSI1584	0.38	0.62	0.46	0.54	0.46	0.54
TSX371	0.45	0.55	1.0	0.0	0.91	0.09
TSI1373	0.29	0.71	1.0	0.0	0.57	0.43
TSX233	0.71	0.43	0.71	0.14	0.71	0.29
OSE003	0.63	0.5	0.63	0.38	0.63	0.38
AENGM8	0.89	0.11	1.0	0.0	1.0	0.0
AENGF8	0.33	0.67	0.78	0.22	1.0	0.0
AENGF7	0.83	0.17	1.0	0.0	1.0	0.0
AENGM2	0.86	0.14	1.0	0.0	0.86	0.14
Mean	WER: 0.37		WER: 0.09		WER: 0.18	

Table 6. Comparison Between Three Systems

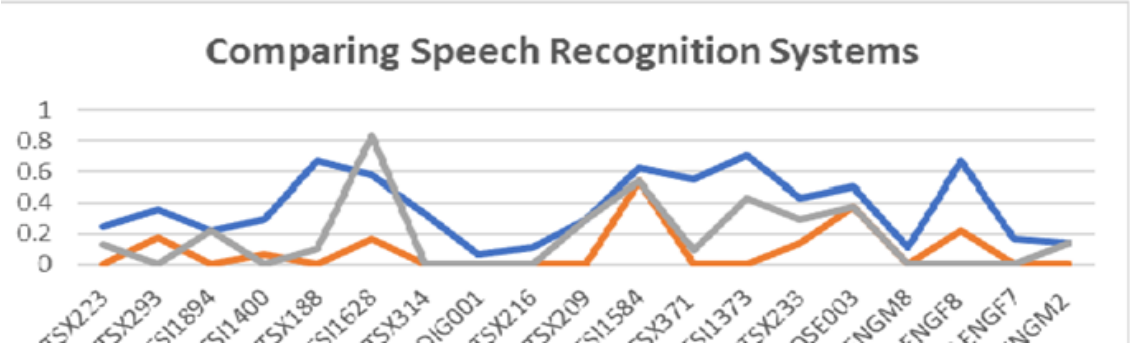


Figure 4.8: The Comparison Result [53]

According to the WER, they also try the different models in ASR systems like the acoustic model, the language model, and the size of the dictionary. This paper says that the tool they made to test Sphinx-4, Microsoft API, and Google API using audio recordings with the original sentences from different sources showed that in Fig 4.8 Sphinx-4 got 37 percent WER, Microsoft API got 18 percent WER, and

Google API got 9 percent WER [53]. As a result, we can say that googles audio modeling and language modeling are better. In our model we focus more on accuracy as google Web .

4.5 Research Challenges

Several study problems need to be solved before a SLAM-based autonomous restaurant robot with navigation, object detection, and speech recognition can be made. We have to face difficulties when it comes to the Linux Operating System. We need to suffer also if one application is supported by any version of linux then the other may not work. In terms of mapping, using the SLAM algorithm is also a challenge for us. First, putting these complicated features together requires synchronizing real-time data from different sensors, like mics and cameras, to make sure that everything works correctly and without any problems. Strong and reliable simultaneous localization and mapping (SLAM) is hard to achieve in busy, changing restaurants with lots of people because of things like changing lighting, different layouts, and moving objects. Object detection adds another level of difficulty, calling for complex algorithms that can find and avoid items, such as tables and customers that move around. Making sure that speech recognition works well in noisy, busy restaurants is even harder. Therefore, the models need to be created that can correctly understand different voices and orders. So, a multidisciplinary method that combines progress in computer vision, sensor fusion, SLAM algorithms, and natural language processing to make a SLAM-based autonomous restaurant robot that is reliable and effective for solving these research problems.

Chapter 5

Conclusion

The autonomous restaurant robots promises to revolutionize the food industry by increasing restaurant efficiency and lowering labor expenses. In addition to the financial gains, these robots have the capacity to completely transform the customer experience by providing a more effective and customized level of service. They relieve human staff of duties like order processing, food and beverage delivery, and cleanup, freeing them up to concentrate on important areas of quality control and client interaction.

But it is important to recognize that the field of automated restaurant robots is still quite young and full of technological difficulties. Overcoming obstacles that necessitate thoughtful thought and creative solutions including congested navigation and dynamic surroundings, guaranteeing safety and security, and resolving possible effects on work.

The deployment of an automated restaurant robot involves a number of critical procedures. Each stage is necessary for the smooth integration of these robots into restaurant operations, from specifying requirements and designing the robot to training, testing, deployment, maintenance, and updates. To further improve overall efficiency, integration with current systems is essential. Examples of these systems include Point of Sale, Inventory Management, and Kitchen Display Systems.

Specifically, in the pursuit of advancing the capabilities of our automated restaurant robot, our work is anchored in the utilization of state-of-the-art technologies, underscoring our commitment to cutting-edge innovation. Specifically, we leverage the powerful synergy of ROS and SLAM for the critical functionalities of mapping, navigation, and object avoidance. ROS serves as the backbone by providing a robust middleware framework that facilitates seamless communication between different components of our robot. This not only enhances the efficiency of data exchange but also streamlines the integration of various features within the system. Complementing ROS and SLAM plays a pivotal role in enabling the robot to dynamically map and navigate its environment in real time. This is particularly essential in the complex and ever-changing landscapes of restaurant settings. Furthermore, to fortify our robots perceptual capabilities, we have integrated SSD MobileNetV3 into its features. This cutting-edge object detection framework empowers our robot with the ability to accurately and swiftly identify objects within its surroundings. The inclusion of SSD MobileNetV3 enhances the overall intelligence of our robot, contributing to its proficiency in tasks requiring object identification, a crucial aspect in

the context of restaurant operations. By seamlessly integrating these advanced technologies, our automated restaurant robot not only addresses the challenges posed by dynamic environments but also elevates its performance in providing efficient and reliable service. This strategic fusion of ROS, SLAM, and SSD MobileNetV3 represents a significant leap forward in the realm of robotic systems, positioning our work at the forefront of technological innovation within the culinary landscape.

Even with mentioning the obstacles, autonomous restaurant robots have a bright future ahead of them. These robots are expected to improve productivity and save labor costs, which will boost customer happiness and boost business in the restaurant sector. According to our research, a more dynamic and effective culinary environment will be possible in the future thanks to the clever solutions that are being used to address present problems and the seamless integration of technology.

5.1 Future Works

In this research, the slam-based robot has been successfully developed with high precision in the chosen scenario where the simulation was deployed. Nevertheless, there is always potential for further expansion of this subject in future investigations. There are multiple functionalities and characteristics that can improve the skills and interactions of the robots in various scenarios, such as restaurant settings. Incorporate the functionality enabling the robot to independently navigate back to a charging station when its battery level is depleted, guaranteeing continuous operation [54]. Developing functionalities to effectively manage emergency scenarios, including the capability to halt operations securely or request aid in the event of technical complications by implementing facial recognition technology to accurately identify and recall customers, enabling the provision of tailored services or assistance. If relevant to the area, offer language choices and assistance for a variety of languages spoken by clients. It will create a feedback and rating system for customers to enable the restaurant to consistently enhance the performance of robots and service [53].

References

[1] A. S. Abdelhakim, M. Abou-Shouk, N. A. F. W. Ab Rahman, and A. Farooq, "The fast-food employees' usage intention of robots: A cross-cultural study," *Tourism Management Perspectives*, vol. 45, p. 101 049, 2023.

[2] M. A. Qasim, F. Abrar, S. Ahmad, and M. Usman, "Ai-based smart robot for restaurant serving applications," in *AI and IoT for Sustainable Development in Emerging Countries: Challenges and Opportunities*, Springer International Publishing, 2022, pp. 107–123.

[3] D. Mazzei, F. Chiarello, and G. Fantoni, "Analyzing social robotics research with natural language processing techniques," *Cognitive Computation*, vol. 13, pp. 308–321, 2021.

[4] J. Jeon, H. R. Jung, F. Yumbla, T. A. Luong, and H. Moon, "Primitive action based combined task and motion planning for the service robot," *Frontiers in Robotics and AI*, vol. 9, p. 713 470, 2022.

[5] A. Gonzalez-Ruiz, A. Ghaffarkhah, Y. Mostofi, *et al.*, "A comprehensive overview and characterization of wireless channels for networked robotic and control systems," *Journal of Robotics*, vol. 2011, 2011.

[6] V. N. Thanh, D. P. Vinh, and N. T. Nghi, "Restaurant serving robot with double line sensors following approach," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, IEEE, 2019, pp. 235–239.

[7] H. Temeltas and D. Kayak, "Slam for robot navigation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, no. 12, pp. 16–19, 2008.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[9] M. C. Bingol and O. Aydogmus, "Performing predefined tasks using the human–robot interaction on speech recognition for an industrial robot," *Engineering Applications of Artificial Intelligence*, vol. 95, p. 103 903, 2020.

[10] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2d lidar-based slam and path planning for indoor rescue using mobile robots," *Journal of Advanced Transportation*, pp. 1–14, 2020.

[11] E. Ma, Y. Bao, L. Huang, D. Wang, and M. Kim, "When a robot makes your dinner: A comparative analysis of product level and customer experience between the us and chinese robotic restaurants," *Cornell Hospitality Quarterly*, vol. 64, no. 2, pp. 184–211, 2023.

[12] İ. AYDIN, "Investigation of the effect of robot waiter usage desire on word of mouth communication and robot waiter usage attitude in restaurants," *Turkish Business Journal*, vol. 2, no. 4, pp. 93–105, 2021.

[13] S. Wan and S. Goudos, "Faster r-cnn for multi-class fruit detection using a robotic vision system," *Computer Networks*, vol. 168, p. 107 036, 2020.

[14] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ros-based visual slam methods in a homogeneous indoor environment," in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, IEEE, Oct. 2017, pp. 1–6.

[15] J. Zhang, Y. Ou, G. Jiang, and Y. Zhou, "An approach to restaurant service robot slam," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2016, pp. 2122–2127.

[16] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, and S. Furber, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proceedings of the IEEE*, vol. 106, no. 11, pp. 2020–2039, 2018.

[17] C. Cao, B. Wang, W. Zhang, X. Zeng, X. Yan, Z. Feng, and Z. Wu, "An improved faster r-cnn for small object detection," *Ieee Access*, vol. 7, pp. 106 838–106 846, 2019.

[18] S. Kibria, "Speech recognition for robotic control," 2005, Master's Thesis in Computing Science, Umea University, 1-77.

[19] K. Wang and M. Z. Liu, "Object recognition at night scene based on dcgan and faster r-cnn," *IEEE Access*, vol. 8, pp. 193 168–193 182, 2020.

[20] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, Z. Liu, H. I. Christensen, and F. Dellaert, "Multi robot object-based slam," in *2016 International Symposium on Experimental Robotics*, Springer International Publishing, 2017, pp. 729–741.

[21] Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, "Object detection recognition and robot grasping based on machine learning: A survey," *IEEE access*, vol. 8, pp. 181 855–181 879, 2020.

[22] O. Mubin, C. Bartneck, L. Feijs, H. Hooft van Huysduynen, J. Hu, and J. Muelver, "Improving speech recognition with the robot interaction language," *Disruptive science and Technology*, vol. 1, no. 2, pp. 79–88, 2012.

[23] K. Hauser, "Learning the problem-optimum map: Analysis and application to global optimization in robotics," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 141–152, 2016.

[24] C. Tian, H. Liu, Z. Liu, H. Li, and Y. Wang, "Research on multi-sensor fusion slam algorithm based on improved gmapping," *IEEE Access*, vol. 11, pp. 13 690–13 703, 2023.

[25] G. Du and P. Zhang, "A markerless human–robot interface using particle filter and kalman filter for dual robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2257–2264, 2014.

[26] J.-M. Jeong, T.-S. Yoon, and J.-B. Park, "Kalman filter based multiple objects detection-tracking algorithm robust to occlusion," in *2014 Proceedings of the SICE Annual Conference (SICE)*, IEEE, 2014, pp. 941–946.

[27] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," pp. 3–27, 2015.

[28] J. Dai, D. Li, J. Zhao, and Y. Li, "Autonomous navigation of robots based on the improved informed-rrt algorithm and dwa," *Journal of Robotics*, 2022.

[29] H. Liu and Y. Zhang, "Asl-dwa: An improved a-star algorithm for indoor cleaning robots," *IEEE Access*, vol. 10, pp. 99 498–99 515, 2022.

[30] Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, "Object detection recognition and robot grasping based on machine learning: A survey," *IEEE access*, vol. 8, pp. 181 855–181 879, 2020.

[31] K. Kim, J. Cho, J. Pyo, S. Kang, and J. Kim, "Dynamic object recognition using precise location detection and ann for robot manipulator," in *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, IEEE, 2017, pp. 237–241.

[32] S. Luo, H. Lu, J. Xiao, Q. Yu, and Z. Zheng, "Robot detection and localization based on deep learning," in *2017 Chinese Automation Congress (CAC)*, IEEE, 2017, pp. 7091–7095.

[33] M. Koteswararao and P. Karthikeyan, "Accurate and real-time object detection system using yolo v3-320 in comparison with mobilenet ssd network," in *AIP Conference Proceedings*, AIP Publishing, vol. 2822, 2023.

[34] L. Jiang, B. Yuan, Y. Wang, Y. Ma, J. Du, F. Wang, and J. Guo, "Ma-yolo: A method for detecting surface defects of aluminum profiles with attention guidance," *IEEE Access*, 2023.

[35] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural networks*, vol. 13, no. 2, pp. 143–148, 2000.

[36] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[37] C. S. Chen, C. J. Lin, and C. C. Lai, "Non-contact service robot development in fast-food restaurants," *IEEE Access*, vol. 10, pp. 31 466–31 479, 2022.

[38] M. Ouyang, X. Shi, Y. Wang, Y. Tian, Y. Shen, D. Wang, and Z. Cao, "A collaborative visual slam framework for service robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 8679–8685.

[39] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "Offloading slam for indoor mobile robots with edge-fog-cloud computing," in *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, IEEE, 2019, pp. 1–6.

[40] J. Zhang, X. Zhang, X. Shen, J. Wu, and Y. Li, "A lidar slam based on improved particle filter and scan matching for unmanned delivery robot," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2506, 2023, p. 012 009.

[41] J. Cheng, Z. Liu, J. He, Y. Deng, and H. Zhang, "Application of simultaneous location and map construction algorithms based on lidar in the intelligent robot food runner," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1972, 2021, p. 012 010.

[42] C. Y. Lee, H. Lee, I. Hwang, and B. T. Zhang, "Visual perception framework for an intelligent mobile robot," in *2020 17th International Conference on Ubiquitous Robots (UR)*, IEEE, 2020, pp. 612–616.

[43] G. S. Huang and Y. J. Lu, "To build a smart unmanned restaurant with multi-mobile robots," in *2017 International Automatic Control Conference (CACS)*, IEEE, 2017, pp. 1–6.

[44] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric a-star algorithm: An improved a-star algorithm for agv path planning in a port environment," *IEEE access*, vol. 9, pp. 59 196–59 210, 2021.

[45] J. Zhang, Y. Ou, G. Jiang, and Y. Zhou, "An approach to restaurant service robot slam," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2016, pp. 2122–2127.

[46] F. Umam, S. Wahyuni, H. Budiarto, and R. M. Putra, "Optimal mapping trajectory for autonomous robot waiter," *Technology Reports of Kansai University*, vol. 62, no. 11, pp. 6429–6435, 2020.

[47] B. Alibegović, N. Prljača, M. Kimmel, and M. Schultalbers, "Speech recognition system for a service robot-a performance evaluation," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2020, pp. 1171–1176.

[48] J. Zhao, S. Liu, and J. Li, "Research and implementation of autonomous navigation for mobile robots based on slam algorithm under ros," *Sensors*, vol. 22, no. 11, p. 4172, 2022.

[49] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ros-based visual slam methods in a homogeneous indoor environment," in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, IEEE, Oct. 2017, pp. 1–6.

[50] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[51] J. Cheng, Z. Liu, J. He, Y. Deng, and H. Zhang, "Application of simultaneous location and map construction algorithms based on lidar in the intelligent robot food runner," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1972, 2021, p. 012 010.

[52] Z. Zhang and Z. Zhao, "A multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm," *International Journal of Smart Home*, vol. 8, no. 3, pp. 75–86, 2014.

[53] V. Këpuska and G. Bohouta, "Comparing speech recognition systems (microsoft api, google api and cmu sphinx)," *Int. J. Eng. Res. Appl*, vol. 7, no. 03, pp. 20–24, 2017.

[54] A. St. Clair and M. Mataric, "How robot verbal feedback can improve team performance in human-robot task collaborations," in *Proceedings of the tenth annual acm/ieee international conference on human-robot interaction*, 2015, pp. 213–220.