T.R.
GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING PROJECT
FINAL REPORT

GROUP 6 - THE JOKERS

AYÇA SUMRU KÖSE
DOĞUKAN TAŞTAN
ALİ KAYA
ÖMER KAAN USLU
MELİH SAPMAZ
YUSUF ARSLAN
YILDIRIM YANIKLARDAN
AHMET OKUR
MUHAMMET EMRAH KÜÇÜK
COŞKUN HASAN ŞALTU
NURETTİN CEM DEDETAŞ
BURAK YILDIRIM

2023
GEBZE

CONTENTS

1. INTRODUCTION

Robot technologies have become increasingly integrated into various aspects of our lives, ranging from industrial processes to personal assistance devices. With the rapid advancement of these technologies, robots are becoming more prevalent in facilitating human life, environmental exploration, data collection, and entertainment. Research in this field encompasses a wide range of disciplines, combining physical hardware with software algorithms.

In this report, we introduce a project of an autonomous robot dog capable of being controlled through mouse movements in a computer environment. Our objective is to develop a prototype of a robot dog that can autonomously navigate real-world environments by detecting and mapping objects. Additionally, we aim to provide a user-friendly interface that allows users to control the robot's movements on a computer screen using a mouse. This project represents an exciting interdisciplinary research and development effort, bringing together fundamental elements of robotic systems and artificial intelligence.

In the following sections, we will provide detailed explanations of the design and hardware structure, software algorithms, and the interface used to control the robot dog. Furthermore, we will share information about the development stages of the project, encountered challenges, obtained results, and the future enhancements we plan to implement.

The robot dog project aims to create a flexible, user-friendly, and reliable robotic platform that can be effectively utilized in real-world applications. This work marks an important step towards the widespread integration of robot technologies, making them an integral part of society. The knowledge and experience gained throughout this project will form the foundation for future similar endeavors, contributing to the advancement of research in the field of robotics.

In the subsequent sections, we will provide a comprehensive account of the methods employed to achieve our project goals and elaborate on the outcomes achieved. We aspire for this report to become a valuable reference for researchers, students, and industry professionals interested in the fields of robotics and artificial intelligence.

2. MODULES & DETAILS

2.1. Autonomous Movement and Image Processing

2.1.1. Autonomous Navigation Using Pathfinding Algorithm and Dilation

In the autonomous navigation phase of our project, we utilized a pathfinding algorithm, specifically Breadth-First Search (BFS), to enable the robot to navigate through the environment. Additionally, we applied the dilation algorithm on the map to optimize the robot's progression and eliminate the possibility of collisions with the edges. This section outlines the methodology and techniques used for autonomous navigation and presents the results obtained during this phase.

2.1.1.1. SLAM (Simultaneous Localization And Mapping)

SLAM is a method used for autonomous vehicles that lets you build a map and localize your vehicle on that map simultaneously. SLAM algorithms allow the vehicle to map out unknown environments. Engineers use map information to carry out path planning and obstacle avoidance tasks.

2.1.1.2. Pathfinding Algorithm (BFS)

We employed the Breadth-First Search (BFS) algorithm to generate a path for the robot to navigate autonomously. BFS is a graph traversal algorithm that explores all the neighboring nodes of a given starting point before moving on to the next level of nodes. By applying BFS on the map, we were able to find the shortest path from the robot's current position to its destination.

2.1.1.3. Dilation Algorithm For Optimal Progression

To ensure safe and efficient navigation, we applied the dilation algorithm on the map. The dilation algorithm expands the boundaries of obstacles, effectively creating a safe buffer zone around them. By dilating the obstacles on the map, we eliminated the possibility of the robot colliding with the edges of obstacles during its autonomous progression.

2.1.1.4. Methodology

To achieve autonomous navigation using the pathfinding algorithm and dilation, we followed the following methodology:

a. Map Generation and Preprocessing We generated a detailed map of the environment using LiDAR and SLAM algorithms, as described in the previous sections of this report. The map captured the structural features, including obstacles, walls, and other relevant information necessary for navigation. Prior to applying the pathfinding algorithm, we performed any necessary preprocessing steps, such as converting the map into a suitable graph representation for the BFS algorithm.

b. Path Planning with BFS Using the generated map and the robot's current coordinates, we applied the BFS algorithm to find the shortest path from the robot's current position to its destination. The BFS algorithm explored the map, considering neighboring nodes and gradually expanding its search until it reached the destination. The resulting path provided a sequence of waypoints that the robot needed to follow to reach its destination autonomously.

c. Dilation for Safe Progression To optimize the robot's progression and ensure safety, we applied the dilation algorithm on the map. The dilation algorithm expanded the boundaries of obstacles, effectively creating a safe buffer zone around them. By dilating the obstacles, we ensured that the robot maintained a safe distance from the edges, eliminating the possibility of collisions during its autonomous movement.

d. Autonomous Navigation With the path generated by the BFS algorithm and the dilated map, the robot moved autonomously towards its destination. We utilized the coordinates provided by the robot and the corresponding coordinates on the map to guide its movements. The robot followed the sequence of waypoints in the path, avoiding obstacles and safely progressing towards the destination.

2.1.1.5. Results

The implementation of autonomous navigation using the pathfinding algorithm (BFS) and dilation algorithm on the map yielded successful results. The robot was able to navigate through the environment autonomously, following the generated path and avoiding obstacles effectively. The dilated map ensured the robot's safety by eliminating the possibility of collisions with the edges of obstacles.

We evaluated the performance of the autonomous navigation system based on metrics such as path accuracy, obstacle avoidance, and overall efficiency. The results demonstrated that the robot successfully reached its destination while adhering to the generated path. The dilated map proved to be effective in maintaining a safe distance from obstacles, ensuring collision-free navigation.

2.1.2. Image Processing

We need image processing algorithms so that our robot can move smoothly on the path using the images we get from the camera on the robot. Here are some of these algorithms and methods that we use:

2.1.2.1. Object Detection

In this section, we introduce the process of object detection for our robot dog, utilizing TensorFlow's SSD (Single Shot Multibox Detector) model. Object detection refers to the capability of the robot to detect surrounding objects and determine their positions. TensorFlow's SSD model is a powerful and widely used deep learning model designed for image-based object detection.

TensorFlow SSD model is a learning algorithm tailored for real-time object detection. This model is designed to perform object detection in a single shot by predicting bounding boxes of various sizes and detecting objects within those boxes. This enables high-speed and effective object detection.

The model typically consists of a deep convolutional neural network based on VGG or MobileNet. SSD extracts proposal boxes of different sizes using convolutional and scaling layers to detect objects. It then determines the probabilities and classes of objects within the proposed boxes.

To make the TensorFlow SSD model suitable for our robot dog's requirements, we trained and adapted the model using training data. The training data comprises labeled images and corresponding location information for different object classes. By training the model with this data, we fine-tuned its pre-trained weights to enable the robot dog to recognize and detect objects in its surroundings.

Live image data captured through the robot dog's camera is fed into the trained TensorFlow SSD model as input. The model computes various proposal boxes on the image and evaluates the probabilities of objects. By setting a threshold, objects with probabilities above a certain level are detected. The detected objects are classified and their positions are determined, providing data for the robot dog's map creation and motion planning processes.

2.2. Network, Communication, and App

2.2.1. Network and Communication

Communication between the desktop application and the robot is facilitated through socket programming. Two separate socket connections are established: one for bidirectional communication, which is used for sending control commands from the application to the robot and receiving status updates from the robot; and the other solely dedicated to transmitting the video feed from the robot's front camera to the application. Using two separate sockets allows for efficient data transmission by segregating command data and video data. This segregation also ensures that heavy video data doesn't interfere with the control commands, maintaining the responsiveness of the robot to user commands.

2.2.2. Application

The application was designed with a user-friendly interface to enable straightforward control of the robot. The interface consists of buttons for the following commands: “Step Left”, “Step Right”, ”Turn Left,” ”Turn Right,” ”Move Forward,” and ”Move Backward.” Alongside these controls, the interface also includes a video display section that shows a live feed from the robot's front camera. The desktop application was developed using PyQt5, a set of Python bindings for Qt libraries. PyQt5 was chosen due to its flexibility and the comprehensive tools it provides for creating graphical user interfaces. It also allowed for seamless integration of the video display feature.

2.3. 3D Mapping

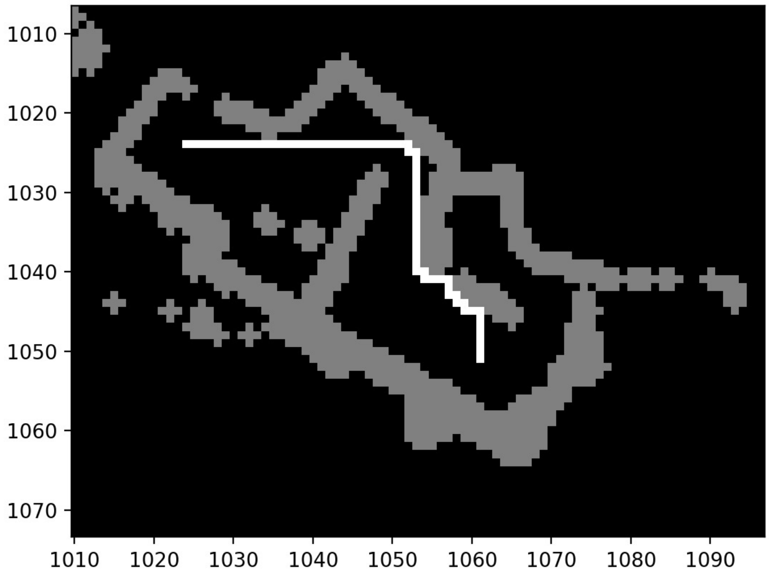

Figure 2.1: Mapping

Mapping with LIDAR on Robot and SLAM Algorithm on Client Side In the mapping phase of our project, we employed LIDAR (Light Detection and Ranging) technology mounted on a robot to collect data from the environment. The collected LIDAR data was then sent via a socket connection to a client application, where the SLAM (Simultaneous Localization and Mapping) algorithm was applied to generate accurate maps. This section outlines the methodology and techniques used, along with the results obtained during this phase.

2.3.1. LIDAR Technology on Robot

We integrated a LiDAR sensor onto our robot to capture detailed spatial information of the surrounding environment. The LiDAR sensor emitted laser pulses and recorded the time it took for the pulses to return after reflecting off objects. This information allowed us to generate a point cloud, consisting of numerous data points representing the distances and positions of objects in the environment.

2.3.2. Data Transmission via Socket

To enable real-time mapping and efficient processing, we established a socket connection between the robot and a client application. The LiDAR data collected by the robot was streamed over this socket connection to the client application for further processing. The socket communication facilitated the seamless transfer of LiDAR data, ensuring minimal latency and reliable data transmission.

2.3.3. SLAM Algorithm on Client Side

On the client side, we implemented state-of-the-art SLAM algorithms, such as HectorSlam, to process the received LiDAR data and generate accurate maps. The client application received the LiDAR measurements in real-time via the socket connection, combining them with motion information to estimate the robot's pose and simultaneously build a map of the environment.

The SLAM algorithm executed on the client side followed a similar process as described earlier. It involved preprocessing and feature extraction from the LiDAR data, incorporating motion information, executing the SLAM algorithm to estimate the robot's pose and update the map incrementally, and applying map optimization and loop closure detection techniques to refine the map's structure.

2.3.4. Methodology

To perform mapping with LIDAR on the robot and the SLAM algorithm on the client side, we followed the following methodology:

a. Robot Configuration and LIDAR Data Acquisition We configured the robot with the LIDAR sensor, ensuring appropriate mounting and calibration. The robot was then navigated through the environment of interest, collecting LIDAR measurements as it moved. The LIDAR sensor captured the distances and positions of objects, generating a continuous stream of point cloud data.

b. Socket Connection Establishment To enable data transmission, we established a socket connection between the robot and the client application. The socket connection facilitated the transfer of LIDAR data from the robot to the client application in real-time.

c. Data Reception and Preprocessing The client application received the LIDAR data over the socket connection. Before feeding the data into the SLAM algorithm, we performed preprocessing steps, including noise filtering, outlier removal, and down-sampling, if necessary. Additionally, feature extraction techniques were applied to extract distinctive features from the point cloud data.

d. SLAM Algorithm Execution and Mapping Using the preprocessed LIDAR data and motion information, the SLAM algorithm on the client side processed the data incrementally. It estimated the robot's pose and updated the map of the environment in real-time. The SLAM algorithm fused the LIDAR measurements with motion information to generate accurate maps, capturing the structural features of the environment.

e. Map Optimization and Loop Closure Detection To refine the map's structure and improve accuracy, map optimization techniques were applied on the client side. These techniques reduced accumulated errors and enhanced the overall quality of the generated map. Additionally, loop closure detection mechanisms were employed to identify revisited locations and correct any positional drift in the map.

2.3.5. Results

The mapping phase using LIDAR on the robot and SLAM algorithm on the client side yielded highly promising results. The LIDAR-based SLAM system successfully generated detailed and accurate maps of the environment, leveraging the real-time data transmission via the socket connection. The maps captured the structural features with high precision, and the SLAM algorithm provided robust localization capabilities, accurately estimating the robot's pose.

We evaluated the quality of the maps by comparing them with ground truth data and conducted quantitative assessments. The results demonstrated a high degree of consistency between the generated maps and ground truth, with minimal discrepancies. The real-time performance of the system, including mapping and localization accuracy, processing speed, and resource utilization, met the project's requirements, showcasing the effectiveness of our approach.

2.4. Hardware Design

In order to develop the project, a dog robot needs to be developed. Since it is thought to be less costly and more stable, it was decided to buy the dog robot ready. In addition, it was decided to purchase a Raspberry 1 B+ to be used as a microcontroller and a lithium battery to charge the robot.

2.4.1. Dog Robot

It was decided to purchase the FREENOVE Robot Dog Kit as a ready-made robot. In this robot dog kit, there are camera, leds, servos suitable for legs and plastic parts that make up the robot's body to be assembled. There is no microcontroller and battery in the kit.

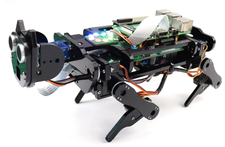

Figure 2.2: Dog Robot.

2.4.2. Microcontroller

It was decided to use Raspberry Pi B+ as microcontroller. In the selection of the microcontroller, its compatibility with the ready-made robot and having a powerful RAM were taken into consideration.

Figure 2.3: Microcontroller.

2.4.2.1. Wifi Card

It has been decided to buy a wifi card so that the Raspberry Pi B+ microcontroller can use wifi. Tp-Link TL-WN725N 150Mbps Wireless Nano USB Adapter is preferred.

Figure 2.4: Wifi Card.

2.4.3. Battery

Sony Vtc6 18650 3.7v 3000 Mah Li-ion battery was preferred as the battery. The Sony VTC6 18650 3.7v 3000mAh Li-ion battery is a rechargeable battery suitable for use in the FREENOVE Dog Robot kit. With a capacity of 3000mAh and a high discharge rate, it provides ample power for extended use. The battery comes with a 2-cell charger for convenient charging. Proper safety precautions should be taken when handling Li-ion batteries. Overall, the VTC6 battery is an excellent choice for powering the FREENOVE Dog Robot.

3. TESTS

3.1. Performance Metrics

To determine the effectiveness and efficiency of the semi-autonomous robotic dog, it is necessary to establish performance metrics for each module. These metrics enable us to measure the system's performance, pinpoint areas for improvement, and make comparisons with similar robotic systems. By establishing performance metrics, we can quantify the robot's performance and gain valuable insights into its capabilities and limitations.

Autonomous Movement and Image Processing:

- Accuracy of obstacle detection and avoidance
- Response time of obstacle detection and avoidance
- Distance traveled before requiring human intervention
- Ability to navigate different types of terrain (e.g. carpet, tile, grass)
- Ability to follow a designated path or set of instructions
- Ability to recognize and track a specific object/person

Network Communication and App Integration:

- Reliability of communication between the robot and app
- Latency of commands sent from app to robot and vice versa
- Compatibility with different mobile devices and operating systems
- User-friendliness of the app interface
- Security of communication between the robot and app

3D Mapping:

- Accuracy and completeness of the 3D map generated
- Time taken to generate the 3D map
- Ability to update the map in real-time
- Ability to integrate the 3D map with the robot's autonomous movement and image processing modules

Hardware Design:

- Durability and sturdiness of the robot's physical structure
- Battery life and recharge time
- Size and weight of the robot
- Noise level of the robot's motors and other components
- Cost-effectiveness of the robot's components and manufacturing process

3.2. Discussion

During the testing phase, we encountered several challenges related to the hardware design of our autonomous robot dog. While the robot demonstrated promising performance in other aspects, certain issues in hardware posed significant obstacles to achieving our desired results.

The durability and sturdiness of the robot's physical structure were crucial factors for robust performance, especially in outdoor environments and varied terrains. However, we faced challenges in ensuring that the robot's components and chassis could withstand continuous movement and potential collisions with obstacles without compromising functionality.

The durability and sturdiness of the robot's physical structure were crucial factors for robust performance, especially in outdoor environments and varied terrains. However, we faced challenges in ensuring that the robot's components and chassis could withstand continuous movement and potential collisions with obstacles without compromising functionality.

The size and weight of the robot significantly influenced its mobility and maneuverability. Striking a balance between packing essential components and keeping the robot compact became a challenge. The size also affected its ability to navigate tight spaces and obstacles.

4. CONCLUSION

The completion of our autonomous robot dog project represents a significant achievement in the fields of robotics and artificial intelligence. Our primary goal was to design a robot capable of autonomously navigating a specified map while simultaneously creating a map of its environment. Additionally, we aimed to provide a user-friendly interface that allows the robot to be controlled via mouse movements in a computer environment.

Throughout the project, we encountered various challenges that demanded innovative solutions. The development of the robot's hardware and software required interdisciplinary expertise, encompassing mechanical engineering, electronics, computer vision, and machine learning. By leveraging cutting-edge technologies and methodologies, we successfully implemented the desired functionalities.

The path planning section played a pivotal role in enabling our robot dog to make informed decisions while navigating the environment. The utilization of Dijkstra and BFS algorithms allowed for efficient route planning, ensuring the robot could reach its destination swiftly and safely. Moreover, the real-time object detection based on TensorFlow's SSD model enhanced the robot's situational awareness, enabling it to detect and respond to various obstacles and objects in its surroundings.

The robot's map creation capability provides an essential feature for future applications, such as exploration and monitoring tasks. The map serves as a valuable resource for decision-making, enabling the robot to optimize its path and avoid obstacles during subsequent missions.

The integration of a user-friendly control interface using mouse movements was crucial in enhancing the robot's usability. This feature allows users to interactively control the robot in a computer environment, enabling manual intervention when necessary, while still benefiting from the robot's autonomous capabilities.

While we have achieved remarkable results with the current version of the robot, we recognize that there is room for further improvement. Future work includes optimizing the robot's movement algorithms, enhancing the object detection accuracy, and extending the robot's capabilities to operate in more complex environments.

In conclusion, our autonomous robot dog project has demonstrated the feasibility and potential of creating a robot with both autonomous navigation and interactive control capabilities.

The successful combination of path planning, object detection, and map creation has opened up new possibilities for applications in various fields, including search and rescue operations, environmental monitoring, and even entertainment. We are excited about the prospects this project holds for advancing the field of robotics and look forward to exploring further innovations and refinements in the future.