Language-Facilitated Robot Task Planning

A dissertation submitted in partial fulfilment of the requirements for the degree of

MSc Artificial Intelligence
Supervisor: **Dr. IOANNA GIORGI**

By

SANJAY JACOB (sj429)

17.09.2024

UNIVERSITY OF KENT

ACCESS TO A MASTER'S DEGREE DISSERTATION

In accordance with the Regulations, I hereby confirm that I shall permit general access to my dissertation at the discretion of the University Librarian. I agree that copies of my dissertation may be made for Libraries and research workers on the understanding that no publication in any form is made of the contents without my permission.

Notes for Candidates: by submitting your dissertation, you agree with the following:

1 Where the examiners consider the dissertation to be of distinction standard, one copy may be deposited in the University Library and/or uploaded into Moodle as an example of good dissertation for future students.

2 If a copy is sent to the library, it becomes the property of the University Library. The copyright in its contents remains with the candidate. A duplicated sheet is pasted into the front of every thesis or dissertation deposited in the library. The wording on the sheet is:

> "I undertake not to use any matter contained in this thesis for publication in any form without the prior knowledge of the author."

Every reader of the dissertation must sign and date this sheet.

3 The University has the right to publish the title of the dissertation and the abstract and to authorise others to do so.

.....

SIGNATURE **DATE**

.....

FULL NAME:

(Please print, underlining surname, in order to assist the cataloguing of theses deposited in the library.)

CERTIFICATE ON SUBMISSION OF DISSERTATION

I certify that:

1 I have read the University Degree Regulations under which this submission is made.

2 In so far as the dissertation involves any collaborative research, the extent of this collaboration has been clearly indicated; and that any material which has been previously presented and accepted for the award of an academic qualification at this University or elsewhere has also been clearly identified in the dissertation.

.....

SIGNATURE **DATE**

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to all those who have supported and guided me throughout the journey of this project.

I express my profound gratitude to my supervisor, **Dr. Ioanna Giorgi**, for your invaluable guidance, encouragement, and insightful feedback. Your expertise and patience have been instrumental in shaping this research.

I am also grateful to the University of Kent for providing the resources and environment necessary for this project.

To my family and friends, your unwavering support and encouragement have been a constant source of motivation. A special thanks to my parents for their understanding, patience, and belief in me throughout this process.

This dissertation is the culmination of the support and encouragement from all of you, and I am sincerely grateful.

Thank you.

ABSTRACT

Robotic applications have long since been an integral part of the industry. Humanoid robots are finding more and applications in the healthcare. More so in the care for elderly as the populations are ageing and there is always a severe shortage of human care givers world over. Mobility becomes a huge concern among the aged as they lose the ability to walk unassisted. A wheelchair is often imperative for them to move around a house or even outside of the house.

This paper gives a brief introduction on the history and evolution of robots, and thereafter provides an outline on the general design aspects of robots. Moving forward, the paper describes the specific design details of the **Robotic Wheelchair (RWC)** and how it is modelled in Webots. Finally an explanation is given on the programming code written in python, to demonstrate that the code can achieve the above desired results.

This project explores, conceptualises and implements a robotic wheelchair model, in the open-source 3D robot simulator, called **WeBots** augmented by the integration of a **Large Language Model (LLM)** for intelligent interaction and decision-making. The primary objective is to create a wheelchair capable of autonomous navigation, obstacle avoidance, and voice-command interpretation, thereby enhancing user experience and accessibility. From being just, a mechanised wheelchair the design of the RWC goes beyond to make the user's daily life easier by incorporating AI based control methods.

The Webots platform is used to simulate the wheelchair's environment and mechanical functionality. Sensors are used for real-time environment mapping and object detection. Movement of the RWC based on voice commands and ability to find and move to a destination along an optimum path inside the house, by using the **A* Algorithm**. This makes it much easier for the user to navigate the RWC in the house environment. Also, an LLM based interactive system, which can infer the user's intent from a general statement, instead of a specific command, is also incorporated. The LLM processes natural language commands from the user, translating them into specific navigation or control actions for the RWC.

Final test results demonstrated that the RWC's controlled system successfully interpreted and executed high-level instructions. This proved the feasibility of integrating LLMs in assistive robotics for natural, human-like interaction. This work highlights the potential of combining advanced AI models with robotics for improved mobility and quality of life.

ABBREVIATIONS

ML	Machine Learning
RWC	Robotic Wheelchair
NLP	Natural Language Processing
AI	Artificial Intelligence
LLM	Large Language Model
ROS	Robot Operating System
SLAM	Simultaneous Localization & Mapping
IMU	Inertial Measurement Unit
GPS	Global Positioning System

Table of Contents

CHAPTER 1

Introduction

1.1 Mobility

Mobility is critical for functioning well and living independently for human beings. As people age, they experience changes in their mobility including changes in gait, physical strength and balance. Gait disorders in the elderly group lead to a loss of personal freedom, frequent falls and injuries and result in a marked reduction in the quality of life and restrict personal independence. By addressing physical, environmental and psychological factors, it is possible to enhance the quality of life in the elderly population who forms around 12% of the world population.

1.2 Assistive and mobility devices for the Elderly

Assistive devices refer to physical equipment like a robot, or a digital tool like **Artificial Intelligence (AI)** based support system, which improves an individual's functioning. Mobility devices are designed to help people who have mobility issues and can help them move around without support. Both these devices create a sense of independence, reduced pain, and increased confidence and self-esteem in the elderly. Mobility devices are largely available to meet the needs of people, assistive robots, wheelchairs and stairlifts being some of them.

Application of robotics in healthcare industry, specifically in the field of elderly care, is advancing, but relatively at a slower pace when compared to the tremendous advancement in other fields. Societies world over are aging and availability of trained professionals for giving care is becoming increasingly limited. This calls for implementation of innovative and cost-effective ideas and solutions to make life easier for the elderly. The ability to move around freely and independently is as important as medications and other support systems. As an assistive technology, Robotics combined with AI has great potential in bringing about a better quality of life for our seniors.

Basic wheelchairs are readily available for elderly with mobility issues, but often would require personnel assistance to move the wheelchair around. In such cases, an advanced wheelchair support system becomes highly imperative in providing both emotional and physical support to this group. A robotic wheelchair would provide complete independence, and a **Large Language Model (LLM)** based interactive voice command system will greatly improve the ease of use of the wheelchair.

1.3 Objectives

This project aims to develop a robust language-facilitated task planning system that enables robots to understand and execute tasks based on natural language instructions. The objectives include understanding of **Natural Language Processing (NLP)**, enhancing the accuracy of translating natural language into executable tasks and developing interactive protocols that can take care of input variability of the users. Hence the primary goal is to model a cost effective and easy to use **Robotic Wheelchair (RWC)** for elderly care and integrate NLP and LLM capabilities for task execution.

CHAPTER 2

Literature Review

2.1 History of Robots

Basic Robots have been in use in industry in the late 19th century, and since the beginning of 20th century, programable robots have evolved at a fast pace. They were designed and built to replace humans in those tasks which were considered dangerous, monotonous and repetitive in nature.

Now complex robots have been designed and produced, with controls powered by Artificial Intelligence. Humanoid robots, as they are called, can interact with humans and can produce human like actions / movements and can more effectively do jobs in service industry.

2.2 Evolution of Robots

The rudimentary origins of the modern-day robots can be said to have begun in the 20th century. In 1950s, George C Devol invented and patented a reprogrammable manipulator called **Unimate**. This can be considered as the first Industrial robot and was put into use in General Motors assembly line. In 1958, another robot called '**Shakey**" was born in the Stanford Research Institute, developed by a team led by Charles Rosen. Shakey was more advanced that Unimate, could move around and could even observe surrounding using its television "eye".

Later in 1960s and 1970s more and more advanced robots came into being. One such robotic arm called Programmable Universal Machine for Assembly (PUMA) was developed in 1978 by Victor Scheinman of Standford. In 1990s, small robotic arms capable of precisely handling equipment in research laboratories were introduced.

21st century has seen a remarkable advancement in the development of more sophisticated robots. Collaborative robots that could interact with humans and perform tasks, and autonomous vehicles, were some of them.

During the past few years, the advancement in the field of artificial Intelligence and machine learning techniques, and their integration into robots, has led to a paradigm shift in the field of robotics.

2.3 Future of Robots

Intelligent autonomous Robots have already become an indispensable element in manufacturing industry, Logistics, medical research, space exploration etc. to name a few. Its application in the healthcare industry is also advancing at a fast pace.

In the times when the populations are aging world over, it is the need of the hour for a significant advancement in the field of robotic support for the elderly care. While the number of older individuals needing care increases, there is limited availability of trained professionals for giving the care.

Mobility is a crucial aspect of health and independence for elderly people. As people age, mobility often becomes more challenging due to a variety of physical, medical, and environmental factors. Improving mobility for the elderly through robot task planning is an emerging field that combines robotics, artificial intelligence, and healthcare to enhance the quality of life for older adults.

In highly unstructured human environments, a robot-controlled wheelchair can significantly enhance mobility and independence for individuals with physical disabilities, particularly in a home setting. With voice commands and automated navigation, operating the wheelchair becomes intuitive, especially for individuals with limited motor skills.

2.4 Introduction to Machine Learning

The field of machine learning emerged in the 1940's/50's. **Machine learning** (**ML**) is a subset of AI that enables software applications to improve their predictive accuracy without being explicitly programmed to do so. The idea behind machine learning algorithms is to use previously known data to train a machine (i.e., a model) to predict the output in relation to the data provided, allowing this model to be deployed in production, where it will be fed with unknown data and should output predicted values in a timely manner based on the previously known data training. Humans acquire knowledge through exposure to something, either through personal experience or through hearing about it from someone else. Machines acquire knowledge through shared experience from the past.

Deep learning has evolved as a subset of machine learning, and it uses multilayered neural networks. Neural network is a model used in machine learning algorithms. The term Neural has been taken from the biological neuron. The human brain consists of billions of neurons which are interconnected to each other to form a network. Neural networks in artificial intelligence (AI) are also inspired from this structure of the human brain.

Neural networks in AI essentially will have layers of neurons, one input layer, one or more hidden layer and one output layer. The number of neurons in each layer and the number of hidden layers depends on the complexity of the computation to be performed. Each of the input neurons is connected to each of the hidden neurons and these connections bear weights. By this way the value in the input node gets multiplied by the respective weight value of the connections and then reaches the hidden neuron. Same way the output of the hidden neurons is connected to each of the output neuron and the connections bear weights.

As mentioned above, the input layer neurons value gets multiplied by the respective weights, and each hidden neuron thus gets a modified value at its input. The values from other input neurons also come to each of the hidden neuron. These values are added, and then a sigmoid activation function is applied to this sum. The resultant value is passed on to the output layer neurons, through connection bearing a different set of weight values. At the output neuron, these values are summed and another sigmoid activation function applied to the resultant value. This activity is termed as **Feed forward.**

The weights mentioned above are initially assigned randomly and hence the resultant value at the output neuron may not necessarily be anywhere near to the target output. So, for this we need to perform another activity, which is termed **Back propagation**. This is fundamental to the concept of machine learning, to train the neural network to achieve a more accurate output

2.5 Robot Task Planning

Robot task planning involves creating a sequence of actions or steps that a robot must perform to achieve a specific goal or complete a task considering the environment, constraints, and goals, while ensuring efficiency and adaptability. The key components of robot task planning are:

- Goal Specification - The task planning process begins with defining the goal or objective that the robot needs to achieve
- Environment Mapping - The robot needs to perceive and understand its environment, which includes mapping the surroundings, identifying obstacles, and recognizing objects.

- Path Planning - Determining the best and the shortest path the robot should take to move from its current location to the goal location, avoiding obstacles along the way.
- Hierarchical Planning – Planning at different levels from high level goals to low-level motor commands.
- Error detection – Identifying when something has gone wrong for e.g. Collision with an obstacle.

2.6 Natural Language Processing (NLP)

NLP is a field of AI focused on the interaction between computers and human language. It involves teaching machines to understand, interpret, and generate human language in a way that is both meaningful and useful. NLP combines elements of linguistics, computer science, and machine learning to process and analyze large amounts of natural language data.

NLP plays a crucial role in Language-Facilitated Robotic Learning, where robots learn tasks, behaviors, or skills through human interaction using natural language. This process typically involves several steps and techniques that integrate NLP with robotics, machine learning, and computer vision. Robots can assist in patient care by understanding and following verbal instructions from healthcare professionals or patients themselves.

Below are the key aspects of Natural Language Processing in Robotic learning:

- Understanding the Language – Robots need to convert natural language instructions into a form they can act upon. This involves parsing sentences to understand the tasks and the objects involved. Robots must also have a contextual understanding of the environment, the relationship between the objects and the goals of the task.
- Response Generation – Robots should be able to generate natural language responses to the instruction given. They might be able to provide feedback to the human instructor to refine their understanding of the task.
- Learning from instructions: Robot can listen to human demonstrations of the tasks and understanding the associated language instructions. Robots can receive language-based feedback during their learning process which helps them refine their actions.
- Multi-modal integration: Integrating NLP with speech recognition enables robots to understand spoken commands making the interactions more natural.

Interactive Learning: NLP allows robots to learn in stages, building on previous knowledge and refining their understanding through repeated interactions

2.7 Webots

Webots is free and open-source robotics simulation software developed by Cyberbotics Ltd. It provides a virtual environment for designing, modeling, and simulating the behavior of robots in complex environments. The software is widely used in robotics research, education, and industrial development due to its versatility and ease of use. Webots simulate realistic physics, including gravity, friction, and collisions, allowing users to test how robots interact with their environments. It offers a 3D environment where users can create and manipulate robot models, sensors, actuators, and objects. The software can simulate various sensors (like cameras, GPS, etc.) and actuators (like wheels, arms, grippers, etc.), enabling users to test complex robot behaviors.

Webots support Robot Task Planning in various ways

- Path Planning Algorithms: Webots supports the integration of various path planning algorithms like A*, Rapidly exploring Random Trees (RRT), or Dijkstra's algorithm. These can be used to navigate a robot through an environment, avoiding obstacles and reaching target locations.
- Multi-Robot Task Planning: Webots supports multi-robot task planning. This includes coordination between robots, task allocation, and collision avoidance, which can be implemented using custom algorithms or integrated with frameworks like ROS (Robot Operating System).
- Integration with External Libraries – Webots allows integration with various external libraries that support advanced task planning, such as ROS or PyRobots. This integration enables the use of more sophisticated planning frameworks, including those that involve machine learning or advanced optimization techniques.
- Webots provides a wide range of simulated sensors (like cameras, LIDAR, GPS) and actuators (like wheels, arms) that can be used in task planning. The availability of these sensors allows for implementing sensor-based planning strategies, such as SLAM (Simultaneous Localization and Mapping) or vision-based planning.

2.8 A* Algorithm for Path Planning

Various path planning methods are in use, like A* Algorithm, Dijkstra's Algorithm, Breadth-First Search etc. A star algorithm being the most widely used in Robotic planning and this has been considered for this project.
It is effective in identifying an optimum path from a start location to a goal location. It has the following components. A heuristic is introduced into a regular graph-searching algorithm, essentially planning ahead at each step so a more optimal decision is made.

- Cost Function – The cost from the start location to the current location
- Heuristic Function – The estimated cost from the current location to the goal location
- Total Cost Function – The sum of the cost function and the heuristic function.

A* Algorithm basically finds the shortest path between an initial point to a goal point. It is used widely in the field of robotics and autonomous vehicles. It finds application in natural language processing and Artificial Intelligence, where this algorithm can be used to optimize decision making processes.

The formula used in the algorithm is as shown below.

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

Where f (n) = total estimated cost of the path through the node n

g(n) = cost so far to reach node n

h(n) = estimated cost from n to target node.

The execution of the algorithm begins with the 'start node', where the adjacent cells are checked whether it contains obstacles. The adjacent cells having obstacles will have higher cost value and one without obstacles will have lower cost value. Total estimated cost (fn) is calculated for these adjacent cells and the cell with the lowest cost is selected. The process is recursively repeated until the shortest path has been found.

2.9 Large Language Models (LLM)

Large language models have extended the horizons of Natural language processing far and wide. It is designed to recognize and interpret human language and can even generate text like how a human do. These models are trained on a large amount of data, and hence the named as "large". They can infer the intent from the conversation and can generate a coherent and relevant response. Most of the LLM

models are trained on data gathered from the Internet and are built on a type of neural network called Transformer model.

Integrating LLMs into robots is a breakthrough towards developing machines which can think and interact with humans in a more intelligible way. It can learn from human instructions and interact naturally with users.

The application of LLM in facilitating a more interactive and easy way of controlling the RWC, was investigated and successfully implemented in this project. This was done by using an Open AI in the control code of the RWC.

OpenAI is a nonprofit research organisation, aiming to develop more and more “safe and beneficial” AI systems to public. It was founded in 2015 with a focus on developing AI and machine learning tools for various activities. Its first offering was an open-source toolkit for developing reinforcement learning algorithms (OpenAI Gym), which prompted it to focus on AI research for more general purposes.

In 2018, OpenAI released the concept of a Generative Pre-trained Transformer (GPT), which is a neural network (a machine learning model) that simulates human brain and is trained on data sets. The type of Neural networks used in LLM models is called as Transformer neural networks.

Conventional machine learning (ML) models applied similar technology among the different models. The relationship frequency between different word pairs or word groups were checked in their training data set. A predictive output of the next word was guessed based on the patterns and relationship of the words in the data set. However, a meaningful output could not be achieved whenever the input text gets longer. Earlier ML models couldn’t generate a full paragraph meaningfully, since it couldn’t retain context between the first and last sentence in a paragraph.

Transformer type of neural networks brought about a fundamental change in this regard and was able to handle long -range dependencies in text. They carry out computation parallelly, thereby reducing processing time of long sequences. The models are trained on huge volumes of datasets, containing billions of parameters, and thereby able to capture a vast amount of human language and knowledge

2.10 Robotic Control Systems

Control systems should be designed in a way that it would regulate and command the functions of the robot to achieve the desired result in an optimum way. In case of a **Robotic Wheelchair (RWC)**, the control mechanism should focus primarily on the safety of the user, and ease of commanding the controller to perform according to the need of the user. Speech recognition for voice-based commands and obstacle avoidance for safety of the user, are some of the control mechanisms built into the RWC. Path planning function and LLM based interaction with the user, enhances the ease of use of the RWC. These are detailed in the below sections.

2.11 Obstacle Avoidance Using Sensors

For any moving robot, obstacle avoidance is an unavoidable feature to be built into its design. It eliminates the risk of collision and damage. This is more important in the case of robots in the healthcare field. A robotic wheelchair for assisting the elderly should have very robust obstacle avoidance system integrated into its controller.

Various kinds of sensors like laser radar sensors, visual devices, ultrasonic and infrared sensors, are available for this purpose. Ultrasonic sensor is used in this project for the control of the RWC.

2.12 Speech Recognition and Voice Based Commands

To enable the robot to respond to verbal commands, a speech recognition application that converts speech into a textual form is needed. This is particularly challenging because of the variations of human voice tone, pattern and speed of talking which, widely varies from person to person. And it varies considerably based on linguistic and cultural differences among populations.

Different algorithms and computational techniques have evolved to overcome the above challenges in speech recognition. Natural language processing (NLP) has advanced as a subset of the wider technology, Artificial Intelligence (AI). NLP focusses on understanding human language from speech, and then convert it into a textual form. A combination of computational linguistics, machine learning algorithms and deep learning is used to achieve this. Specifically designed AI models need to be trained using large amounts of labelled data, and for this human intervention is largely required. Self-supervised models have been able to overcome this difficulty.

There are generally three different types of NLP: Rule based NLP, Statistical NLP and Deep learning NLP. Rule based NLP applications were modelled based on decision trees and relied on pre-programmed rules. Statistical NLP which evolved later, relied on machine learning technology, and could automatically extract, classify and label voice and text data. It then looks for a statistical likely hood of the possible meaning of the voice and text data.

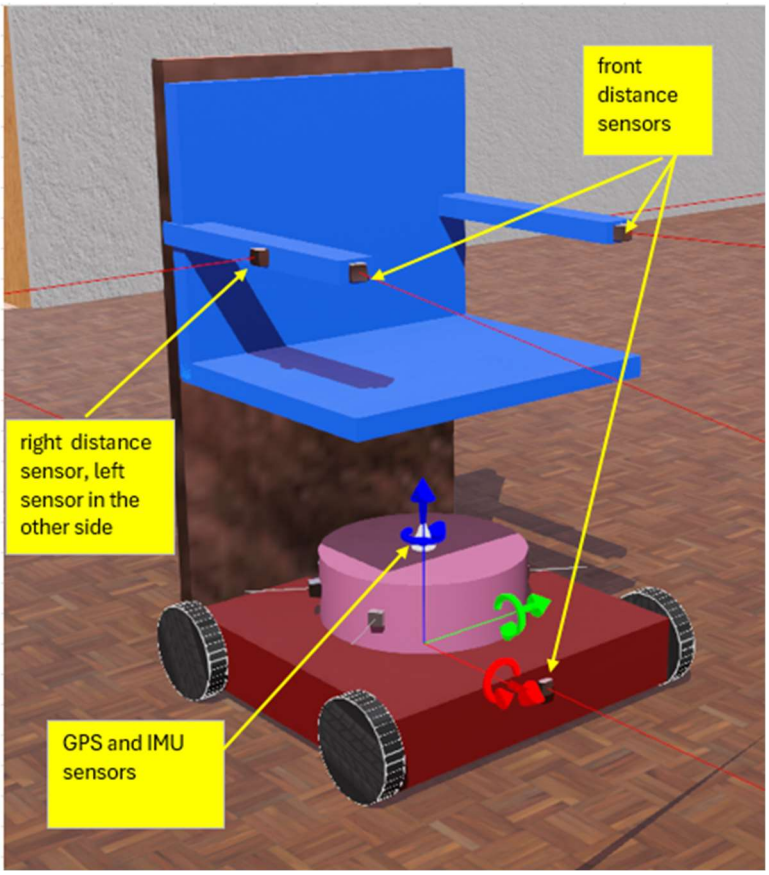
Currently Deep learning NLP has become the most widely used NLP

CHAPTER 3

Implementation of a Robotic Wheelchair model in Webots

3.1 Design of the Robotic Wheelchair (RWC)

The wheelchair is to be used by an elderly patient residing in the house, with limited home care support. The patient is not completely immobile and is able to place himself on to the wheelchair after getting up from the bed. But he is unable to walk around the house unassisted, and so the wheelchair would be used always for movement.



Picture -1 Model of the RWC created in Webots

A four-wheeled robotic wheelchair is designed in WeBots as shown above. It is equipped with 4 wheels powered by 4 different motors, which can be controlled separately. Six ultrasonic distance sensors, are mounted on the RWC as below:

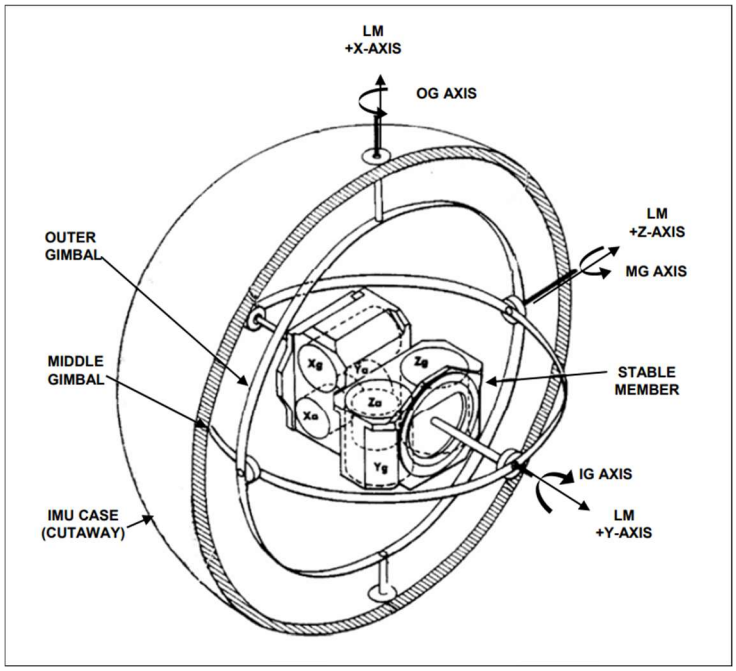
- 3 Front sensors, 2 on the arm rests and one on the wheelchair base, to detect obstacles in its direct path.
- 2 Side sensors looking left and right
- 1 Rear sensor, mounted on the wheelchair base backside to avoid obstacles while moving backward.

A **Global Positioning System (GPS)** and an **Inertial Measurement Unit (IMU)** device are also attached to the RWC, which are mounted on the base

In Webots, the wheelchair is to be designed as a solid node with child nodes of 4 wheels connected through 4 hinge joints. A drive motor and a position sensor are also added to the wheels. The 4 individual drive motors are given unique names so that the controller code can identify each of them and individually drive them. Distance sensors will be built into the wheelchair, and these are used for obstacle avoidance.

3.2 Sensors and Third-Party Applications Used in RWC

a) Inertial Measurement Unit



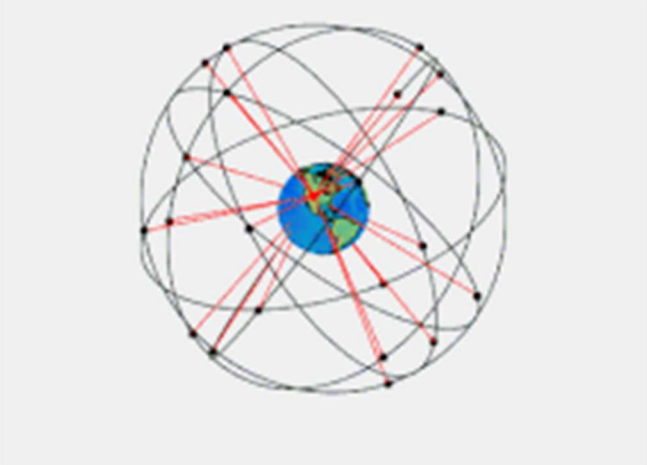
Picture -2 Sketch of an IMU. **image courtesy: Wikipedia**

Using a set of accelerometers, gyroscopes and magnetometers, IMUs help a moving object measure its own velocity, acceleration and orientation with respect to the environment. In Webots there are models of sensors such as the above mentioned IMU, which can be integrated into the robot model being designed.

An IMU device is integrated into the robotic Wheelchair. It helps the wheelchair to find its orientation with respect to the house environment. At a given time, the RWC can be anywhere inside the house, like the bedroom or kitchen or another place. Its orientation is derived from the IMU device position of the Wheelchair with respect to the house environment.

The IMU model in Webots returns the '**roll**', '**pitch**' and '**yaw**' angles if called for in the code.

b) Global Positioning System (GPS)



Picture -3 Sketch of a GPS. **image courtesy: Wikipedia**

GPS is a navigation system used worldwide, owned by the United States government. It uses a network of satellites and enables the users to accurately know where they are on the surface of the earth. A GPS receiver would give the exact location by coordinate system based on the latitude and longitude of earth.

In Webots, a GPS sensor module is available to be used in moving objects like a car or a moving robot. Such a GPS module is integrated into the robotic wheelchair, and it helps know the current position.

c) Distance Sensors

Different types of generic distance sensors are available depending on their principal of functioning, as listed below:

- Ultrasonic
- Infrared
- Laser
- Proximity

The Ultrasonic Sensor is the most common distance measuring sensor. It detects the distance to objects by emitting high-frequency sound waves and measuring the time taken for those sound waves to return. Ultrasonic distance sensor is modelled in this project, for using on the RWC to detect obstacles in front and back.

3.3 AI based functions built into the robotic wheelchair

The following AI based functions are built into the wheelchair control, to make it easier for the user to navigate around the house environment.

a. An obstacle avoidance system for the wheelchair which would prevent collision with any obstacle around.
b. A voice-based command system – Based on the voice commands given, the wheelchair moves in respective directions for a short distance.
c. A voice-based system by which the wheelchair moves from the current position to the destination desired by the patient. A defined path would be followed by the wheelchair to move to these locations.

d. A large language model to interact with the patient to understand the needs of the patient, and to move the wheelchair to a location where the patient's needs can be met.

A voice-based command function would enable the patient to move the wheelchair in different directions and stop it. It would also move the wheelchair to a location in a predetermined path. An LLM built into the controller would try to understand the need of the patient and would take the wheelchair to a specific location.

Using a set of accelerometers, gyroscopes and magnetometers, IMUs help a moving object measure its own velocity, acceleration and orientation with respect to the environment. In Webots there are models of sensors such as the above mentioned IMU, which can be integrated into the robot model being designed.

An IMU device is integrated into the RWC. It helps the wheelchair to find its orientation with respect to the house environment. At a given time, the RWC can be anywhere inside the house, like the bedroom or kitchen or another place. Its orientation is derived from the IMU device position of the Wheelchair with respect to the house environment.

The IMU model in Webots returns the 'roll', 'pitch' and 'yaw' angles if called for in the code. In the code made for RWC, only the 'yaw angle' is called for since the wheelchair is having movement only in the horizontal plane (XY plane). The angle at which the wheelchair turned with respect to the 'downwards direction', is calculated by the function: get heading in the python code, and this angle is used to determine the heading of the wheelchair.

in moving objects like a car or a moving robot. Such a GPS module is integrated into the robotic wheelchair, and it helps know the current position.

3.4 Implementation of the RWC in Webots, version R202b

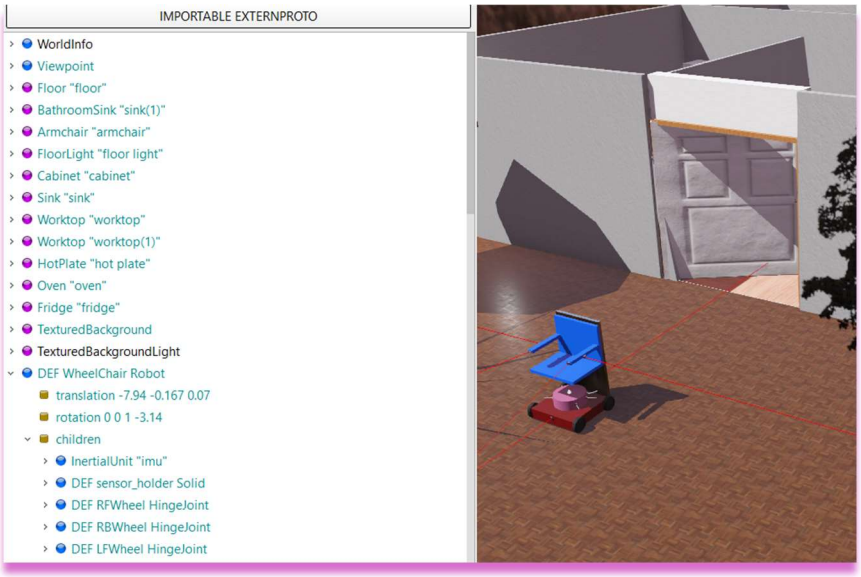
Picture -4 – House environment for the RWC as implemented.

3.4.1 Design of a House Environment

3.4.1.1 Scene Tree

In Webots, the **scene tree** contains the information that describes the simulated world. The scene tree is composed of a list of nodes, each containing fields. The nodes contain information about the outside environment like the sky and lighting, the house, garden, walls, furniture, domestic appliances etc. The information about the RWC, is also held in the node named as “Wheelchair Robot” in the scene tree representation in the below screen shot picture 5.

The node named “Worldinfo” forms the basis of the simulated world, and contains information like the position of the objects, general appearances, how objects interact with each other etc. In the real-world scenario, properties like gravity, friction, masses of the objects act upon individual objects and among each other. To simulate such a real-world and real object, the features and parameters required are defined in the world info node and in each of the individual nodes representing the various objects.



Picture -5 – Scene tree in Webots

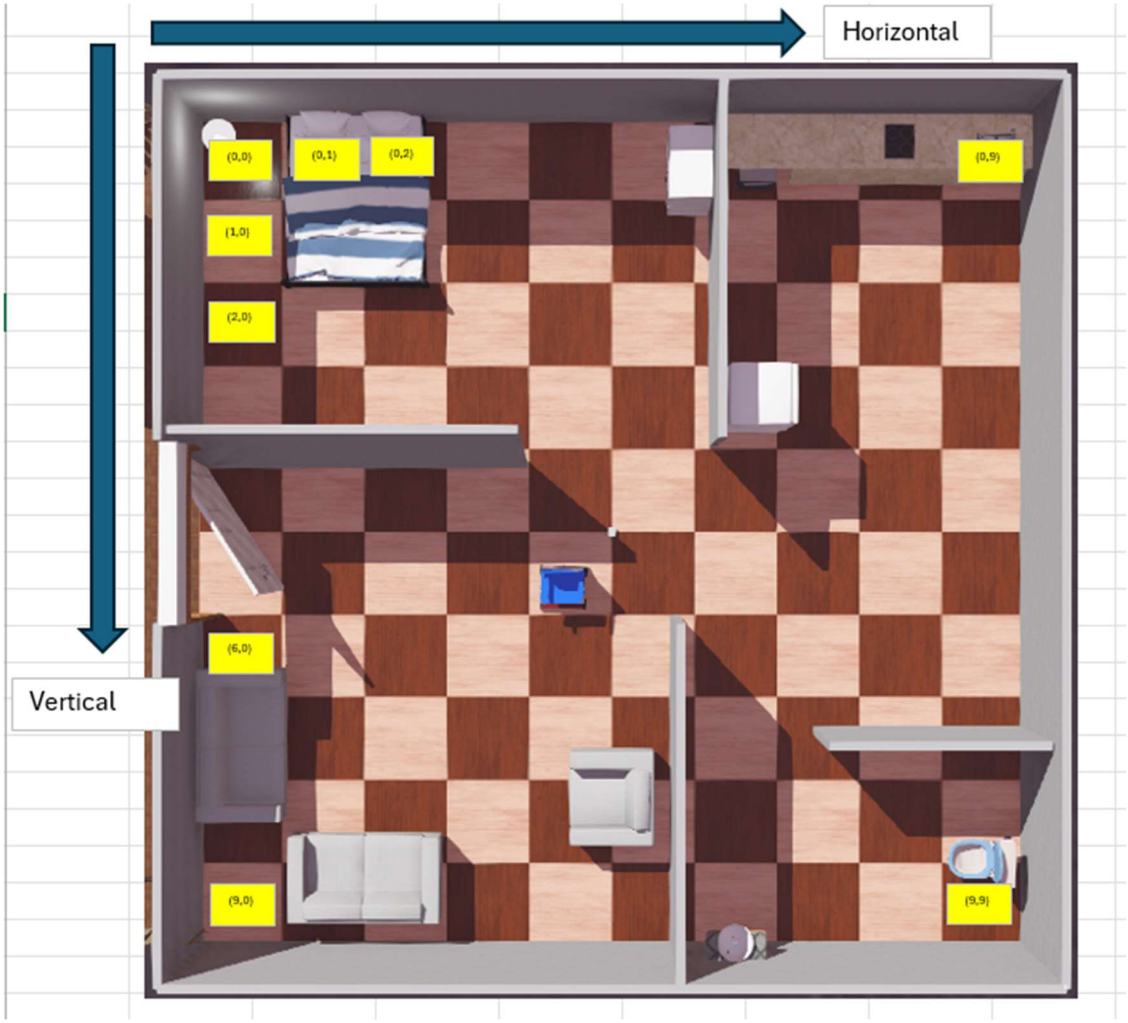
The scene tree of Webots is structured like a VRML97 file. As mentioned above, it is a list of nodes, each containing fields which contain the information of the objects they represent. Fields can contain values (text strings, numerical values) or other nodes. The nodes can be cut, copied and pasted, which makes the building of the environment easy. For example, one wall can be made defining the required properties and then the node can be copied to create more walls having similar properties.

The node named “Wheelchair robot” contains all the information about the wheelchair, its shape and geometry, its physical design, color, type of material etc. There are sub nodes, or the ‘children’ nodes created for the wheelchair node, and this contains all the information about the individual components that make up the complete wheelchair. These include the various sensors (which are described in section 3.2), wheels, hinge joints and the wheels. The four wheels built into the wheelchair’s base structure are responsible for the movement of the RWC inside the house environment.

The ‘controller’ field in the Wheelchair Robot node holds the directory location of the implementing code. The implementing code is written in Python and is saved in the default directory of Webots, from where the code is executed each time the simulation is run.

3.4.1.2 The Floor of the House

In Webots, the term ‘arena’ represents the floor on which the objects are placed. A rectangular arena is created which forms the floor. The floor size is 10 meters X 10 meters and is divided into tiles of size 1 meter X 1 meter. Different walls divide the floor area into four tooms, Bedroom, Living room, Kitchen and washroom. To form a grid-based coordinate system, the total floor area is divided into cells represented by the squares in the below picture. Such a grid based is floor has been required for implementing a path planning algorithm for the RWC. The cells need to be identified by a coordinate system, so that the algorithm can identify the location of the RWC at any given time and to navigate to a desired location. This coordinate system designed is detailed below in picture 6.



Picture -6 – Coordinate system for floor explained

Each cell (square) can be identified by a pair of coordinates, like (x, y), where x would represent the position in one direction and y, in the other direction. The horizontal cells have been considered as rows and the vertical lines as columns. The numbering of the cells starts from the origin point which is the top left corner cell. The numbering starts from “0” to match with the common practice in coding. The term zero- indexed, where in data structures such as lists, the first element is at index 0, the second at index 1 and so on.

As indicated in the above picture, the top left corner cell would be (0,0), and the next cell in the horizontal (rows) would be (0,1) and the next one (0,2) and so on.

Similarly, the cell next to the (0,0) cell in the vertical direction (column) would be (1,0), and the next one (2,0) and so on.

3.4.2 Design of voice command and obstacle avoidance mode (Operating mode – 1)

In the operating mode-1, the RWC takes command to move based on voice commands. The user gives voice commands like, 'forward', 'left'. 'right', 'backward', 'stop' etc and the RWC controller code converts the voice command to textual form Using Google API. The recognised text command is used in the code to control the speed of the wheels and thereby move or turn the RWC. By setting the speed of all the four wheels in the same positive direction, it moves forward. For turning right, the speed of the right wheels is made negative, meaning they rotate in the opposite direction to that of the left wheels. Similarly to turn left, the left wheels are set at negative speed and thereby the turn in the opposite direction. This causes the RWC to turn left.

When obstacle is encountered in the front, the front sensors measure the distance to it. If the distance measured by the front sensors is < 0.5 meters, the RWC turns to either the left or the right direction.

In an obstacle avoidance robotic movement, when an obstacle is sensed in the front, the direction the robot should turn is decided based on the specific application of the robot. In the case of RWC, it is equipped with two distance sensors on either side. This measures the distance to the nearest obstacle towards the sides. Direction to turn is based on the distances measured by the left and the right sensors to obstacles towards left and right directions. If measured distance to the right is more than the distance measured to left, then the RWC turns right. Likewise, if distance measured by the left sensor is more than the distance measured by the right sensor, the RWC turns left.

3.4.3 Design of AI based control for the RWC (*Operating* mode – 2)

The below Artificial Intelligence based control mechanisms are implemented in the RWC

3.4.3.1 Path planning using A * Algorithm

Path planning mode typically refers to a feature or algorithm used in robotics, autonomous vehicles, and similar systems to determine an optimal or feasible route from a starting point to a destination while avoiding obstacles. The goal is to find a path that minimizes some cost, such as distance, time, or energy, while adhering to constraints like avoiding obstacles or following specific rules.

The robotic wheelchair (RWC) conceptualised in this project should focus on the movement inside the house with minimal commands. As explained in the previous section, in the operating mode -1, the RWC responds to the verbal commands of the user and make basic movements like 'moving forward', turning right, turning left etc. But moving from one place to the other inside the house may involve many turns and hence may need the user to give more voice commands. To make such movements inside the house easier to the user, a path planning mode (operating mode-2) is also built into the RWC.

For this purpose the following things shall be established:

- Representing the environment – (forming a grid map)

In the context of this project, house layout where the RWC is expected to move shall be mapped in a digitised form. The floor l shall be divided into grid of cells, each cell representing a small portion of the area. This forms the grid map. The path through which the RWC must move is planned by identifying the cells forming the optimum route.

- Pathfinding Algorithm

Identification of the optimum route is achieved by A* algorithm.

- Identification and representation of the obstacles in the map.

The obstacles in the map shall be identified and represented in the above-mentioned grid-map.

In this RWC project, the wheelchair user needs to move from one location in the house to another, depending on the needs. The house environment has walls and other household objects which poses as obstacles in the path of the wheelchair. The algorithm should identify an optimum path to the goal location by avoiding these obstacles. As detailed in the previous section, the floor of the house is divided into square shaped cells and each of these cells are assigned a pair of X, Y coordinates. These coordinates are based on their position from the top left corner of the floor map.

An obstacle map is where the known obstacle in an environment is mapped into a horizontal plane, in this case, the floor of the house. The below is the obstacle map created for the Robotic wheelchair project. It is a rectangular array of numbers (Matrix) with elements representing each cell of the floor plan of the house. The entry in the matrix is either 0 or 1. 'Zero' value represents that the cell is empty, ie not having any obstacles and the One' value indicates that there is an obstacle in the cell. The A* Algorithm must avoid the cells having obstacles to reach upon the shortest path to the destination.

```
[1, 1, 1, 0, 0, 0, 1, 1, 1, 1],
[0, 1, 1, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0],
[1, 1, 1, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 1, 0, 1, 1, 1],
[1, 1, 1, 0, 1, 1, 0, 0, 0, 1],
[1, 1, 1, 0, 1, 1, 0, 0, 0, 1]
```

Picture -7: - Obstacle map

A* Algorithm calculates the path having the smallest 'cost', which is usually the path having the least distance to reach the target, or shortest time etc. The path consists of nodes, which are basically the intermediate points in the path. In the context of the RWC project these nodes are considered as the square cells of the house floor, which has been explained in the previous section. So the path identified by the algorithm will consist of a start node and a target node, with many intermediatory nodes as shown in the below picture 8.

Picture - 8 – identified path with nodes

3.4.3.2 LLM based voice interactive user commands

To make the interaction with the RWC more humanistic for the user, an LLM model is implemented in the control. Open AI is a non-profit organization focusing on the development of AI models. GPT stands for Generative Pre-Trained Transformer. GPT models act like intelligent and interactive voice assistants. They can create human-like responses to natural language queries. They also can perform a wide range of other tasks like creating original textual and visual content, understand and summarize textual data, and even create software codes.

Essentially GPT models are natural language-based prediction models. They are trained on billions of parameters on massive language data sets, and use the patterns and styles used in them to create new content. GPT models are based on a specific type of neural network which is based on Transformer architecture. Some of the other types of neural networks are ‘recurrent” and “convolutional’.

The transformer neural network architecture uses self-attention mechanisms to focus on different parts of the input text during each processing step. A transformer model can capture more context and performs better in natural language processing (NLP) tasks.

Open AI GPT services can be integrated into robotic applications to create humanistic robots. In the RWC control, it gets a human-like response to a query from the elderly user. OpenAI's GPT-3.5-turbo model is used to interpret user’s commands and provide corresponding actions for the wheelchair. For example, if the person needs some milk and doesn’t recall where it is stored in the house, the user can speak a query into the microphone. The RWC’s code sends this query to the open AI GPT and gets a response with the word “Kitchen” in it. This key word captured is used to navigate the RWC to the Kitchen in the house. The path it takes to reach the house is determined by the A*Star algorithm already implemented in the code (as described in section 3.4.3.1).

The responses obtained from Open AI GPT model is based on certain configurable parameters, like **‘system roles’**, ‘**max tokens’**, ‘**temperature**’ etc.

‘**System role’** in the code provide setup information of context that informs the model on the expected

responses. It sets the behaviour and guidelines for the assistant. The responses can adapt its responses to cater to the needs of the conversation. In the case of RWC, the expected responses is in the context of home environment, and hence the 'system role' has been defined in the code as "You are a helpful assistant that understands and interprets user commands for a wheelchair in a house environment". This enables interpretation of commands and responses related to moving a wheelchair within a home setting.

The parameters **max_tokens** determine the maximum length of the output text generated by the model. Increasing max tokens allows the model to produce longer responses. Such a response will have more content and detail. But on the other hand, they will contain irrelevant and out of the context text. So an optimum value is often chosen to strike a balance. In the **RWC**, the max token is set as 1000 tokens in the code.

The parameter **"temperature"** influences the randomness and creativity of the model's responses. It is usually a number between 0 and 1. Lower the value, the responses would tend to be more straightforward and predictable. It would be more deterministic in that; the responses are likely to be consistent. As the 'temperature' value get higher, the responses to tend to have the opposite behaviour. In the RWC design, a 'temperature' setting of **0.5** is adopted. It provides a mix of creativity.

CHAPTER 4

Conclusion

4.1 Contribution

Mobility issues faced by elderly was addressed in the initial part of this thesis. The limited availability of trained health care professionals for the care of elders poses a real challenge in the healthcare industry.

While self-powered wheelchairs have existed for the past few decades, ones with Artificial Intelligence built into their controls have not really developed at a necessary pace.

In this thesis, the integration of NLP, LLM model and a path planning techniques were explored. The findings demonstrated that leveraging on natural language as a means of communication with any robotic application, and more so with an assistive wheelchair, significantly improves the ease of operating and achieves a humanistic interaction with an elderly user.

Simulated functional testing of the Robotic Wheelchair, as mentioned in section 4.2, was carried out in the home environment, and achieved successful results. The simulated model demonstrated successfully it's capability in performing the desired tasks. The author is confident that the RWC model can be converted into a real-world Wheelchair, and that it would perform satisfactorily and meet its objectives.

4.2 Testing the simulation

4.2.1 Testing of the Voice Command Recognition and Processing

Accurate recognition of the voice commands was the key to the success of this project. The voice inputs for operating mode -1 included simple movement commands like, 'forward', 'left', 'right', backwards etc. More complex natural language queries were used for testing the operating mode -2, which were processed by the LLM.

The success rate which was calculated by the number of times the RWC responded correctly to the total number of trials, was more than 90%. However, the time taken for processing the voice commands were sometimes high, this is likely due to the response delay from the Google API server.

Overall, the test results met the objectives of this project.

4.2.2 Natural Language Queries

Commands like "Where can I find the milk?" or "I am feeling sleepy" were interpreted using the LLM. Here, the recognition accuracy was slightly lower, with an overall success rate of around 70%.. Misinterpretations occurred at times, this is assumed due to the 'temperature' settings of the OpenAI GPT, which needs to be fine-tuned.

Overall, the test results met the objective of the project.

4.3 Limitations and future enhancements

While integration of NLP and LLM into a self-powered wheelchair looks promising, the cost of manufacturing such a robotic wheelchair should not be exorbitant and beyond the reach of a common man. Using more cost-effective physical components like variable speed-controlled driving motors, sensors etc. and government regulations in subsidizing such components would be a solution. Voice recognition and LLM models should continue to be open sourced and freely available for development of such robotic applications.

Based on the insights gained during the implementation and testing of the RWC, the following improvements could be made to the controls.

a. **Voice Recognition:** The delayed response to the voice commands, as mentioned in section 4.2.1, can be investigated and a solution to this explored.

b. **LLM Limitations:** The occasional misinterpretations while more using complex sentence structures, as mentioned in section 4.2.2, can be improved by fine tuning of the LLM model.

c. **Enhanced obstacle avoidance:** Obstacle avoidance system was designed for a static environment, but to suit a real-world scenario, need to be designed for dynamic environments. More sophisticated sensors and integrating advanced algorithms like SLAM (Simultaneous Localization and Mapping), can make the RWC work effectively in dynamic environments.

References

1. Language in Our Brain: The Origins of a Uniquely Human Capacity, *by Angela D. Friederici, Cambridge, MA: The MIT Press (2017)*
2. Artificial intelligence - A modern approach *by Stuart Russel*
3. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation *by O. Michel*
4. Robot Task Planning Based on Large Language Model Representing Knowledge with Directed Graph Structures *by Yue Zhen, Sheng Bi, Lu Xing-tong*
5. Boosting Robot Intelligence in Practice: Enhancing Robot Task Planning with Large Language Models *IEEE*
6. Deepbots: A Webots-Based Deep Reinforcement Learning Framework for Robotics - Conference paper *by M. Kirtas, K. Tsampazis First Online: 29 May 2020*
7. A review of methodologies for natural-language-facilitated human–robot cooperation by *Rui Liu1 and Xiaoli Zhan*
8. Natural Language Processing: A Text book with Python Implementation *by Raymond S.T.Lee*
9. A Comprehensive Overview of Large Language Models *by Humza Naveed, July 2023*
10. Evaluating Large Language Models Trained on Code *by Mark Chen, et al 07/07/2021*
11. Incremental Learning of Humanoid Robot Behaviour from Natural Interaction and Large Language Models *by Leonard Bärmann, Rainer Kartmann 08/09/2023*
12. https://www.coursera.org/articles/
13. https://cs.stanford.edu/people/eroberts/courses/
14. https://www.simplileam.com/future-of-robotis-article

Annexure 1

Explanation of the Python code for Operating mode - 1

The code controls a Wheelchair robot to move forward, backwards, and turn left or right based on voice commands. Also, the robot detects obstacles in front and back, through distance sensors (ultrasonic sensors) mounted on the Wheelchair. Distance sensors are also mounted on the sides. They measure the distance to the obstacles on the sides. When the front sensors detect an obstacle within 0.5 meters, and if the left sensor value is more than the right value, the wheelchair turns left. Likewise, if the right distance is sensed more, it turns towards right.

The following is a brief explanation of the program code written in Python language, for achieving the Voice control & obstacle avoidance mode.

Code lines 9 - 11,
the required library functions given below, are imported.
speech_recognition
threading
robot controller

Code line 14,
Creating a recognizer instance

Code lines 17-18
Initializing the Robot instance
Retrieves the basic timestep in Webots simulation.

Code lines 22-32
This function initializes motors (robot): creates a dictionary of motor devices using their names and sets them to velocity control mode.

Code lines 35-42,
The function capture voice adjusts the ambience noise and records the audio bit using microphone.

Code lines 46-57
Pre-processes the recorded audio bit and convert into text using Google API

Code lines 60-62
Wheel control functions are defined.

Code lines 69-80
The 6 distance sensors are initialized and enabled.

Code lines 69-80
Retrieves the value of the sensors and saves it in the variables front_value, frontright_value, frontleft_value, left_value, right_value, back_value.

Code lines 91-126
If the voice command is recognised as ‘forward’, it sets the left and right wheels speed at the set value, and the RWC moves forward. If an obstacle is sensed by any of the front sensors, frontleft_value, front_value, frontright_value, then it turns left or right, depending on whether left value or the right value is greater.

Code lines 130-124
If the voice command is recognised as “slow’, it makes the RWC move forward at a very low speed. This command is introduced to enable very slow movements if required, especially when the wheelchair is near walls or other obstacles.

Code lines 136-147
If the voice command is recognised as ‘backward’ at a slower speed. If any obstacle is detected behind, the RWC is commanded to stop immediately and take no further action.

Code lines 150-164
If the voice command is recognised as either ‘right’ or ‘left’, the RWC makes the turn accordingly.
If recognised as ‘stop’, the wheels of the RWC are set to speed zero and stops.

Code lines 168-175
Deines a function with two parameters: motors (which controls the robot) and shared command (a dictionary that tracks the current command).
Captured voice is assigned to a variable ‘audio’.
The audio which has been preprocessed and recognized earlier, is assigned to variable, “new_command”.
The condition *if new_command and new_command != shared_command['current']*: is used to check whether a new command has been recognized and whether it differs from the previously stored command.
Update the current command stored in the shared command dictionary with the new command that was just recognized.
The new command is used to control the motors.

Code lines 178-188
Main function to execute the below.
Initializes the robot’s motors and sets up a dictionary to track the current command
Concurrently manage voice command recognition while running the main control loop of the robot.
Within this loop, control_robot(shared_command['current'], motors) is called to process and execute the current command.

Annexure - 2

Explanation of the Python code for Operating mode - 2

This code achieves the control of the wheelchair with the aid of a large language model, OpenAI GPT 3.5. The wheelchair user speaks into the microphone indicating the intent. For example: "Where can I find the milk in the house". This speech is processed by Open API to make out the user's intent. The response given by Open API would contain the 'key word' which will indicate the location in the house where the wheelchair is to be taken. For example, "Kitchen", "Bathroom" etc. The shortest path to the destination is derived using the A star algorithm. The floor of the house is divided into grids which divide the floor into cells. The chequered floor in Webots indicates each cell. The cells are numbered from left-top corner starting as (0,0), (0,1)....(0,9) in the horizontal direction and (0,0), (1,0).....(9,0) in the vertical downwards direction.

The current position of the wheelchair in the house is obtained from the GPS device built into the wheelchair. An IMU device is also integrated into the wheelchair to determine the direction, by calculating the yaw angle.

The following is a brief explanation of the program code written in Python language, for achieving the LLM and path planning mode. Please refer to Annexure -2 for the full python code

Code lines 15 - 20,

the required library functions given below, are imported.

math
heapq
time
speech_recognition
openai
robot controller

Code lines 23 – 29,

Some parameters are used in the code, like VELOCITY, GRIDWIDTH, GRIDHEIGHT, DIRECTIONS, CELL SIZE etc., for the movement of the RWC and for the grid formation on the floor to achieve path planning.

Code line 32

A valid openai GPT 3.5 API key is used to make use of the LLM model for interaction with the user.

Code lines 36-40

The grid position of each room is defined and will be a ‘Dictionary map’ from which matched commands can retrieve the position.

Codes lines 44-59

A Robot controller class is created, retrieves the basic timestep for simulation, and this would determine how often the control loop should be executed. The RWC’s 4 motors are initialized and set the motors to velocity control mode.

Code lines 62-65
Inertial measurement unit (IMU) and Global positioning system (GPS) are integrated into the RWC to identify its orientation and locate its position in the house environment. These sensors are initialized in these codes.

Code lines 68-69
Speech_recognition and microphone library functions are used in the program to capture the users voice and convert them into text, so that it can be used in the command line of the RWC for achieving voice based control for the wheel chair.

Code lines 72-74
Ensures the motors are stopped at start, clears the list that keeps track of the RWC's path., and clears the list of positions the RWC has visited.

Code lines 78-82
The GPS coordinates returned by the GPS sensor is converted into a matrix coordinates.

Code lines 84-86
Size of each cell is defined based on GPS range and grid size.

Code lines 89-90
GPS coordinates converted to Grid coordinates.

Code lines 93-96
checks if the gps x value is within the range from -5 to -4 (excluding -4). If it is, it sets grid x to 0. This could be used to handle cases where gps x is very close to the left edge of a grid or map.
Similarly, it checks the same for gps y values too, in cases where it is very close to the top edge of the floor map.

Code lines 99-103
Ensures grid coordinates are within bounds and returns corrected grid coordinates.

Code lines108-121
Get the current position from GPS and map it to grid coordinates.

Code lines 123-131
Converts the voice input 'command' to its lower case, and checks against a predefined set of voice commands, and returns the associated position if a match is found.

Code lines 133-135
Computes the heuristic function between two points and returns the Manhattan distance between the points.

Code lines 133-135
Sets the velocities of the RWC's 4 motors.

Code lines 146-148
Defines a function to stop all motors by setting their velocities to 0.

Code lines 151-161
The Move forward function is defined which is called for, would make the RWC move forward for a specified duration. The duration is specified by the MOVE_DURATION parameter.

Code lines 164-176

An Inertial Measurement unit (IMU) is integrated into the RWC, and this code uses it to get the 'yaw' angle from it. The function, get_heading, determines the RWC's orientation based on the yaw angle. It then converts the yaw angle in radians, to degrees. The value in degrees is then categorized as N,S,E,

W, if it fall within a certain range corresponding to the directions.

Code lines 179-186
The ‘turn to direction’ function adjusts the RWC’s heading towards a specific desired direction.

Code lines 189-191
It calculates the number of 90deg turns required for the RWC to turn to the desired direction from the current direction.

Code lines 195-214.
Turn duration, that is time taken for turning left, right, and 180 degrees, is directly inputted into the program. turn_durations dictionary is converted into a turn_to_direction method is a great way to make your robot's turning logic more flexible and adaptable to different turn durations.

Code lines 219-245
Turning and fine turning are executed with a specific time duration.

Code lines 248-257
Makes the RWC turn and align towards a specific direction and then move forward. The earlier defined functions get_heading, turn_to_direction, and move_forward are called during the execution of these lines of code.

Code lines 261-278
A function get_coordinates_from_voice is defined. The code captures and processes the voice commands by using the speech_recognition library function. Audio input is converted into text with Google’s Speech recognition API. Microphones are used as the audio input, and the background noise is adjusted.

Code lines 281-307
The defined function ‘interpret_command_with_llm’ uses the LLM gpt-3,5-turbo to comprehend the user’s intent and then give a suitable response to it. It uses OpenAI's GPT-3.5-turbo model and the response key word and looks up in an earlier defined VOICE_TO_POSITION dictionary.
Max_tokens set the number of words the model can generate in response.
The temperature parameter defines the level of randomness in the response. A lower temperature makes the response more deterministic and consistent, but on the other hand it less creative or varied. A temperature setting of 0.5 is used in this project.

Code lines 310-349
The defined function in this code, ‘find_and_execute_path’ is to find a execute a path from a start node to a target node. The A* algorithm is used to find the shortest path.
In A* algorithm, open list is a priority queue that stores nodes to be evaluated. Each entry in the list is a tuple which has the f value, g value, current node and the path.
(please refer to section 2.2.3).
The heapq.heapify() function is used to convert a list into a heap in-place, where the smallest element is always at the root of the heap. It is useful for maintaining a priority queue efficiently.
The while loop continues to execute if there are nodes in the open list.
The neighbours list represents the possible moves or directions from the current node.
Code then contains a loop which iterates over each neighbouring node, extracts the coordinates of the neighbor, checks the boundaries and whether it is an obstacle. Then calculate the costs and add the neighbour to the open list.

Commands to move the RWC based on the path found by the A* algorithm is then executed.

Code lines 357-383

The main() function creates and instance of Robotcontroller.

The while ‘controller.robot.step(controller.timestep’ loop is continuously executing the simulation until it ends of encounters an error.

It listens for a voice command and determines the goal node from it.

Once the goal node is found, the current position is determined as the start node.

The ‘board’ represents the grid map of the house floor, with 0s indicating empty space and 1s indicating obstacles.

Finally, the find_and _execute_path function is executed to determine the optimal path from the start node to the goal node, with the board (obstacle map) configuration.