Journal of Information Systems Engineering and Management
2025, 10(29s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

An Interactive Autonomous Forklift Robot Based on Large Language Models

Breno U. De Angelo[1], Andre G. C. Pacheco[2], Rafael F. Freitas[1], Italo J. O. Silva[1], Eduardo P. De Abreu[1], Luiz F. S. C. Cardoso[1], Thiago Oliveira-Santos[2], Claudine Badue[2], Alberto F. De Souza[2].

[1]*Undergraduate Student, Departamento de Informática, Universidade Federal do Espírito Santo, Brazil*
[2]*Doctor, Departamento de Informática, Universidade Federal do Espírito Santo, Brazil*

ARTICLE INFO

Received: 29 Dec 2024

Revised: 15 Feb 2025

Accepted: 24 Feb 2025

ABSTRACT

This paper introduces Hercules, an autonomous robot designed to simulate a specialized cellulose bale forklift, integrating cutting-edge technologies in natural language processing (NLP) and speech synthesis. Equipped with advanced sensors and communication capabilities — such as 2D cameras, LiDAR, and a microphone array — Hercules interacts with its environment and operators using its sensors, actuators, and natural language supported by text-to-speech and speech-to-text functionalities. The robot's architecture enables real-time interaction with non-expert users, allowing it to autonomously navigate, detect objects, and perform precise grasping tasks. This work outlines the development and integration of state-of-the-art large language models into a robotic system to enable two-way communication and execution of commands, demonstrating the robot's efficiency, accuracy, and versatility in a simulated warehouse environment. Our experiments and evaluations exhibit Hercules' capabilities to understand, respond to, and execute natural language commands effectively, presenting promising prospects for enhanced human-robot interaction in practical industrial settings.

Keywords: Autonomous Robot, Cellulose Bale Forklift, Human-Robot Interaction, Natural Language Processing, Speech Synthesis, Speech Recognition, Large Language Models.

INTRODUCTION

Over the past few years, we have witnessed a significant transformation in the way robotics permeates our daily lives and impacts industrial processes (Garrido-Jurado et al., 2014). Autonomous robots have found applications in a variety of sectors, such as automotive (Benotsmane et al., 2020), logistics and storage (Bernardo et al., 2022), agriculture (Rose et al., 2021), self-driving cars (Badue et al., 2021), among many others (Ma et al., 2020), (Alatise & Hancke, 2020). As robots advance in their capabilities, they are increasingly deployed in environments governed by people who may not be experts in robotics (Tellex et al., 2020). In these contexts, the ability to interact with robots naturally is a key factor in the successful deployment of robots in the real world (Zucker et al., 2015). Using abstractions such as visual programming (Coronado et al., 2020) provides some flexibility; however, it still requires training. Hence, building robots that are capable of interacting with humans through natural language is highly desired in the area.

Recently, the field of Artificial Intelligence (AI) has undergone a revolution due to the advent of the Transformer neural architecture (Vaswani et al., 2017) which is the foundation of today's Large Language Models (LLMs) — such as GPT-4 (Achiam et al., 2023), Llama-2 (Hugo Touvron et al., 2023), and Mistral (Jiang et al., 2023).

Also, the Transformer architecture has been used as the basis for robust speech-to-text (STT) models such as Whisper (Alec Radford et al., 2022) or has inspired relevant aspects (attention mechanism) of text-to-speech (TTS) models such as Tacotron 2 (Shen et al., 2018). These models achieve human-level performance in a variety of tasks, such as text generation, question answering, and code generation (Achiam et al., 2023), (Hugo Touvron et al., 2023), (Jiang et al., 2023). Naturally, the LLM's success has also been extended to the field of robotics. Since Natural language processing (NLP) is fundamental for human-robot interaction, there are multiple applications where robots can benefit from LLMs, including task instruction, navigation, and information retrieval (Vemprala et al., 2023). The integration of LLMs has emerged as a game-changer in the robotics domain, redefining the scope and potential of the use of autonomous robots and human-robot interaction.

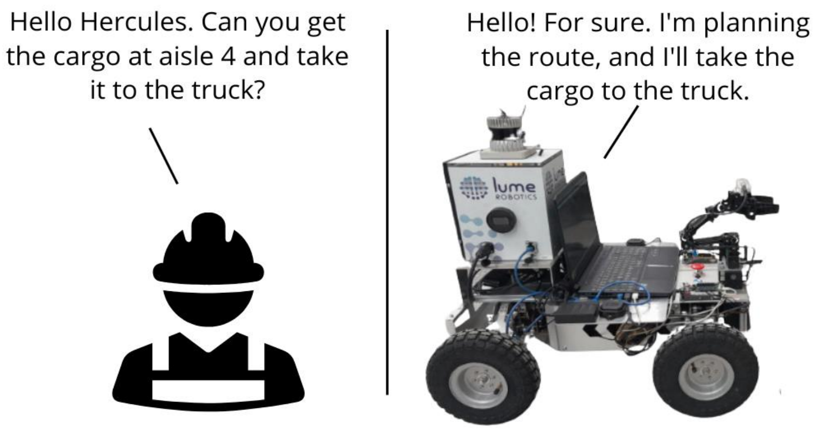

Hello Hercules. Can you get the cargo at aisle 4 and take it to the truck?
Hello! For sure. I'm planning the route, and I'll take the cargo to the truck.
lume ROBOTICS

Figure 1: Illustration of Hercules interacting with a non-expert robotics operator, understanding natural language commands, formulating responses, planning, executing tasks, and reporting back through speech.

In this paper, we introduce Hercules, an autonomous robot equipped with a variety of sensors — including 2D cameras, a LiDAR, and a microphone array — that simulates a specialized cellulose bale forklift, enabling the comprehensive study of the environmental perception and interaction capabilities required within industrial settings. As illustrated in Figure 1, Hercules' main goal is to interact with the environment through the use of natural language, i.e. Text-to-Speech (TTS), and Speech-to-Text (SST) capabilities, to perform complex tasks, including autonomous navigation, precise object detection, and object grasping. To do so, Hercules is designed to integrate state-of-the-art neural models to perform NLP and speech synthesis tasks. The integration empowers not only two-way communication in natural language, but also the robot's ability to understand, respond to, and execute commands efficiently and accurately, significantly enhancing its versatility and usefulness in varied environments. The main contributions of this paper are:

- We build a robot that integrates state-of-the-art technologies to perform NLP and speech synthesis tasks. The robot simulates a real-world task of cargo transportation, which has multiple applications in the industry.
- We present an architecture enabling real-time interaction between the robot and non-expert users. The robot operates autonomously, identifying tasks and executing them without human intervention. This includes navigation, obstacle avoidance, object detection, and grasping.
- We carry out experiments to assess the robot's performance in a warehouse simulation. Results demonstrate its ability to understand, respond to, and execute natural language commands efficiently and accurately.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the robot's hardware, software architecture, and capabilities. Section 4 presents the robot's performance in a real-world scenario. Finally, Section 5 concludes the paper and discusses future work.

RELATED WORK

While there has been a recent increase in enthusiasm for integrating natural language into robotics, this field of research has been active for a long time (Chen & Mooney, 2011), (Tellex et al., 2020). Historically, language processing has been used in a variety of applications such as search and rescue tasks (Murphy et al., 2007), general object description (Morales et al., 2006), and assistive robotics (Fasola & Mataric, 2012). However, its integration into robotics has been limited by the lack of robust and efficient NLP models. The recent development of Transformers (Vaswani et al., 2017) and the emergence of LLM models (Achiam et al., 2023), (Hugo Touvron et al., 2023), (Jiang et al., 2023) have been leading to a new wave of research in the field, with the integration of LLMs in a variety of applications aiming to enable seamless communication, sophisticated understanding of human commands, and intelligent decision-making by autonomous agents.

Different studies have showcased the integration of language models in robotics. Recently, (Vemprala et al., 2023) introduced the ChatGPT (Achiam et al., 2023) for robotics, which creates a set of functions that can be called by the

GPT model, being instructed using prompt engineering. In summary, the authors propose a collaborative framework named PromptCraft to allow the creation of high-level commands using the ChatGPT API. A similar idea was proposed by (Singh et al., 2023), who introduced ProgPrompt, a framework that uses programming language structures as prompts for performing the robot task planning. Experiments show promising results; however, the framework demands basic knowledge in programming to create the prompts. (Wu et al., 2023) introduced the TidyBot, a robot that employs an LLM to learn the user's preferences — via prior interactions — and autonomously tidy up a given room. The authors claim that their approach integrates language-based planning and perception alongside the few-shot summarization capabilities of LLMs to deduce generalized user preferences that hold broad applicability for future interactions.

Other works propose the use of LLMs to perform navigation tasks. (Wang et al., 2023) use few-shot prompts collected from the physical environment to allow a Large Language Model to autoregressively generate low-level control commands for robots without task-specific fine-tuning. Their experiments demonstrated the effective capability of their model in guiding a quadruped robot's locomotion, enabling it to walk proficiently. (Bucker et al., 2022) trained an LLM model to reshape robot trajectories using natural language commands. In brief, when a robot is performing a task, the user may intervene and provide a command to change the robot's trajectory. (Lin et al., 2023) proposed Text2Motion, a planning framework rooted in language that empowers robots to tackle sequential manipulation tasks demanding long-horizon reasoning. This framework, when provided with a natural language directive, generates comprehensive plans at both task and motion levels, ensuring the attainment of inferred symbolic goals.

Applying LLMs to robotics is currently an area of intense interest. Nonetheless, most of the studies are still in the early stages of development, and there is still substantial progress required before the models can be effectively deployed in real-world applications. While most of the studies are focused on prompting a network to obtain trajectories/navigation instruction, in this paper we focus on the user-friendly interaction between the robot and the humans. As previously mentioned, the proposed architecture aims to integrate state-of-the-art technologies to allow the robot to interact with non-expert users and perform the requested tasks without human intervention. In addition, the robot interacts with the environment through speech, which is not explored in the related works.

MATERIAL AND METHODS

This section provides a detailed overview of the key elements supporting the development and functionalities of our autonomous robotic system. Subsections cover the conceptual design, hardware architecture, Autonomy System, and the integration of the Large Language Model (LLM), Text-to-Speech (TTS), and Speech-to-Text (STT) systems.

A. Conceptual design

The problem studied was inspired by the work of forklift operators in cellulose transportation at. Employees are typically assigned to transport piles of bales from a warehouse to a truck. Once the truck is full it departs. Usually, the industrial environment is controlled. Workers are restricted to use sidewalks; and the piles and the truck are placed in predefined positions.

The designed robot is fully autonomous and emulates a specialized cellulose bale forklift used for transporting materials within warehouse environments. Figure 2. illustrates both the designed robot and the real forklift it aims to emulate. Note that this forklift type picks up bales by pressing them on both sides instead of using forks from the bottom. Like the real forklift, Hercules employs an Ackermann steering mechanism, like a car's articulated steering, enabling precise maneuvering and navigation in confined warehouse spaces.

B. Architecture overview

To enable Hercules to operate autonomously and interact seamlessly with the environment and individuals, a comprehensive array of sensors, actuators, and controllers was integrated into its system, represented in Figure 3 as green, yellow, and blue, respectively.

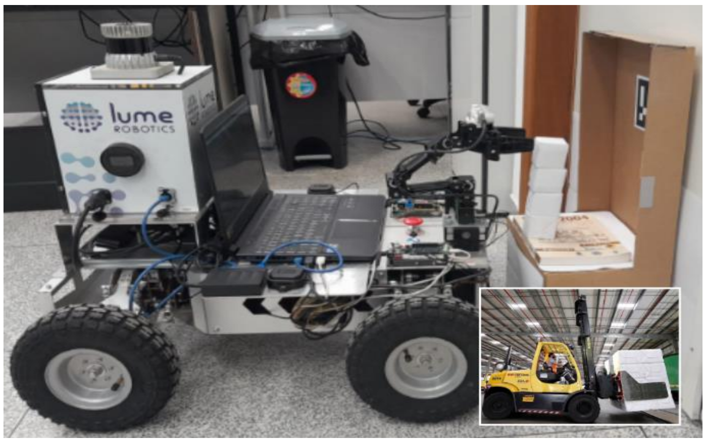


Figure 2: Illustration of the designed robot Hercules and the real forklift it aims to emulate.

Hercules' Autonomy System runs on a laptop equipped with an Intel i7 processor, 16GB of RAM, and an NVIDIA RTX 3070 graphics card. The system integrates data from various sensors, including encoders on wheel motors, steering wheel angle measurements, Inertial Measurement Unit (IMU), Global Positioning System (GPS), and Light Detection and Ranging (LiDAR). This fusion of inputs is vital for accurately determining Hercules' position and velocity. Movement commands are sent to specific actuators, including DC, Servo, and Step motors, as well as a 5-degree of freedom (DOF) robot arm designed for grasping and manipulating objects.

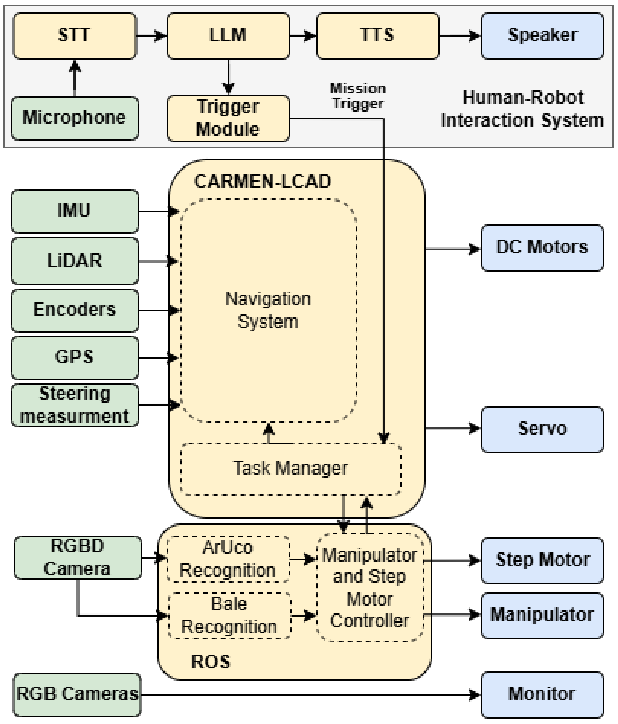


Figure 3: A high-level schematic diagram that shows an overview of Hercules' hardware and software architectures.

Despite the manipulator having more degrees of freedom than a standard cellulose bale forklift, its movements are deliberately restricted through software. This is done to mimic the operational limitations of a traditional forklift, thereby ensuring it remains compatible with and adheres to the usual handling procedures found in warehouse settings.

The robot is also equipped with a total of five cameras. Four of these cameras operate in RGB mode and mimic human viewing in remote operation scenarios. The fifth camera, an RGBD Intel RealSense D435i, is strategically positioned at the gripper of the manipulator. It aims to accurately estimate the pose of objects to be picked and realize the recognition of ArUco (Garrido-Jurado et al., 2014) markers used for fine localization.

C. CARMEN-LCAD

Hercules' Navigation System follows the architecture commonly found in self-driving cars (Badue et al., 2021). It relies on a version of the Carnegie Mellon Robot Navigation Toolkit1 named CARMEN LCAD2, which is a modular open-source software dedicated to mobile robot control, and it is the main software of IARA (an acronym for Intelligent Autonomous Robotic Automobile), an autonomous vehicle designed to navigate in complex large-scale urban environments (Cavalcante et al., 2021). As shown in Figure 3, CARMEN LCAD is the Hercules' main controller block.

Hercules Navigation System closely mirrors IARA's architecture, which is composed of two main systems: perception and decision-making systems. The perception system utilizes onboard sensors such as LIDAR, cameras, GPS, IMU, and more to estimate the robot's state and build an internal representation of the environment. The decision-making system guides the robot from its starting point to the user-defined destination by analyzing the robot's state and internal environment representation (Badue et al., 2021).

Hercules closely mirrors this architecture, primarily encompassing the following functional blocks (see Figure 4):

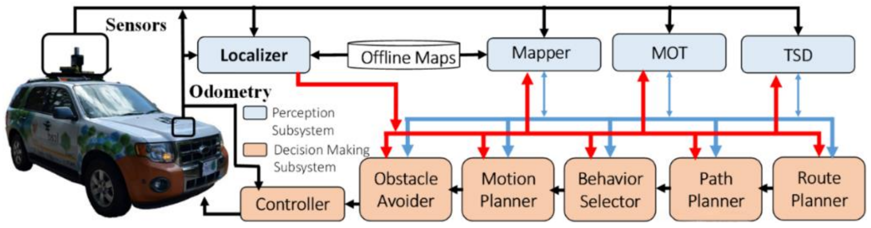


Figure 4: Illustration of the Intelligent Autonomous Robotic Automobile (IARA). For further details, please refer to (Badue et al., 2021).

• Localizer: it is responsible for estimating the robot's state (pose, linear velocities, angular velocities, etc.) in relation to the static map of the environment. The static map, also referred to as the offline map, is generated prior to autonomous operations, employing the identical set of sensors integrated into the robot.

• Mapper: this module takes the offline map and the robot's state as inputs, producing the online map as output. This online map combines data from the offline map with an occupancy grid map computed in real time using sensor data and the present state of the car.

• Moving Obstacles Tracker: this component utilizes the offline map and the current state of the robot, enabling the detection and tracking of nearby moving obstacles, such as other vehicles and individuals. It calculates both the pose and velocity of these obstacles for real-time monitoring and assessment.

• Route Planner: it computes a route, in the offline map, from the current robot's state to the final goal. A route is a sequence of waypoints, where each waypoint is a coordinate pair in the offline map.

• Path Planner: this component calculates a series of paths by evaluating the robot's present state and its internal representation of the environment. A path is essentially a sequence of poses, wherein each pose denotes a coordinate pair within the offline map, specifying the desired orientation of the robot at that particular location.

• Motion Planner: it is tasked with computing a trajectory from the robot's current state to the designated goal, ensuring adherence to the defined path while accommodating the robot's kinematic and dynamic constraints. This trajectory is composed of a sequence of commands, each specifying a desired velocity, a desired steering angle, and the duration for which each command should be applied. A trajectory navigates the robot from its current state to the intended goal in a smooth and safe manner.

• Obstacle Avoider: this module receives the computed Trajectory from the Motion Planner and, when necessary, adjusts it -- often by reducing the velocity -- to prevent potential collisions.

[1] https://carmen.sourceforge.net

[2] The project that is available on https://github.com/LCAD-UFES/carmen_lcad

● Controller: this module takes the Motion Planner's trajectory, possibly modified by the Obstacle Avoider module, and calculates effort commands directed to the steering wheel servo and traction DC motors. These commands are intended to guide the robot in executing the modified trajectory as closely as possible, considering real-world physical constraints.

An extra module is introduced for the control of Hercules, the Task Manager. The module is responsible for high-level control of the robot. Using a custom created language, the developers can define missions, which are sequence of operations that need to be performed by the robot to achieve a final goal. The summarized mission used by Hercules was the following:

```
# hercules_mission.txt
get_first_bale:
park at 7757696.37, -363595.74, 1.557
yolo get bale from pile

put_first_bale_on_truck:
park at 7757695.97, -363593.13, 3.128
put bale on the truck

get_second_bale:
park at 7757696.37, -363595.74, 1.557
yolo get bale from pile

put_second_bale_on_truck:
park at 7757695.97, -363593.13, 3.128
put bale on top

# Retorna a posicao inicial
park at 7757691.46, -363593.36, -3.086
announce "Mission completed!"
```

The language supports recovery in case of failed tasks, which were omitted in the paper for conciseness, but are implemented in Hercules. It also supports flow control so that the approach would generalize for an arbitrary number of bales, but for the sake of simplicity it was not implemented in Hercules.

As shown in the mission, Hercules is asked to move the bales one by one from the pile to the truck in a way that a new pile is created in the truck. The park commands used in the mission trigger the Navigation System to autonomously navigate the robot to a specified pose. The commands such as yolo get bale from pile trigger task manager routines that communicate with the Manipulator and Step Motor Controller via ROS topics to acquire the position of ArUco makers and bales and send simple commands, such as:

● Move forward: x (meters)

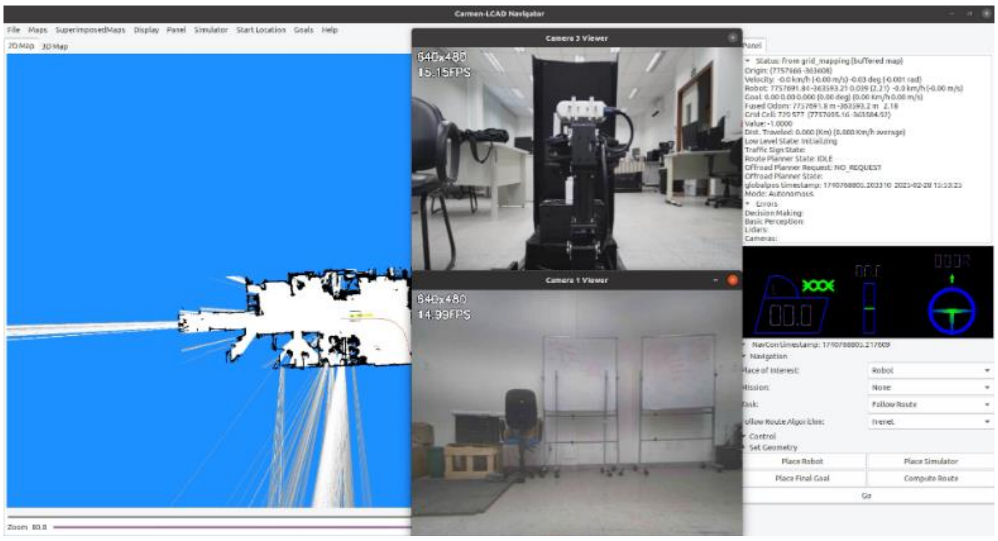
● Move horizontally: y (meters)

● Move vertically: z (meters)

● Close gripper: c (%)

By doing this, the Task Manager can perform all the calculations necessary to determine the individual movements the manipulator need to perform, considering its own dimensions and the environment.

With all the necessary integrations, a graphical user interface is available for the user to monitor the data from the sensors and visualize the goals and paths determined by CAMEN-LCAD's navigation framework, as illustrated by Figure 5.



```
Status: from grid_mapping (buffered map)
Origin: (7757666 -363608)
Velocity: -0.0 km/h (-0.00 m/s) -0.03 deg (-0.001 rad)
Robot: 7757691.84 -363593.21 0.039 (2.21) -0.0 km/h (-0.00 m/s)
Goal: 0.00 0.00 0.000 (0.00 deg) (0.00 Km/h 0.00 m/s)
Fused Odom: 7757691.8 m -363593.2 m  2.18
Grid Cell: 720 577 (7757666.16 -363584.57)
Value: -1.0000
Dist. Traveled: 0.000 (Km) (0.000 Km/h average)
Low Level State: Initializing
Traffic Sign State:
Route Planner State: IDLE
Offroad Planner Request: NO_REQUEST
Offroad Planner State:
globalpos timestamp: 1740768800.203310 2025-02-28 15:13:23
Mode: Autonomous
Errors
Decision Making:
Basic Perception:
Lidars:
Cameras:
NavCon timestamp: 1740768800.217609
Navigation
Place of Interest: Robot
Mission: None
Task: Follow Route
Follow Route Algorithm: Frenet
Control
Set Geometry
```

Figure 5: Autonomous navigation using CARMEN-LCAD software: simultaneous visualization of the robot-generated map, onboard cameras, and sensor data for environment mapping and localization.

D. Bale Recognition

This study's problem consists of autonomously transporting bales from a pile in a warehouse to a truck. To achieve this, the robot must identify the center of the bale with an accuracy of a few centimeters. Similarly, the forklift must detect the position where the bales need to be placed.

A YOLOv8 model, based on the You Only Look Once (YOLO) object detection framework (Redmon et al., 2016) was fine-tuned to detect the bales. To do so, a dataset of 330 images was collected using a cell phone camera and labeled using Roboflow, a web-based tool for computer vision tasks. An example of a labeled image used for training is presented in Figure 6. To enhance the dataset and improve model robustness, 529 additional images were generated through data augmentation techniques. These augmentations included horizontal flipping, random rotations between -15° and +15°, and brightness adjustments ranging from -15% to +15%. The final dataset was divided into 795 images for training, 32 for validation, and 32 for testing. The trained model was then integrated into a ROS-based vision pipeline to enable real-time object detection.

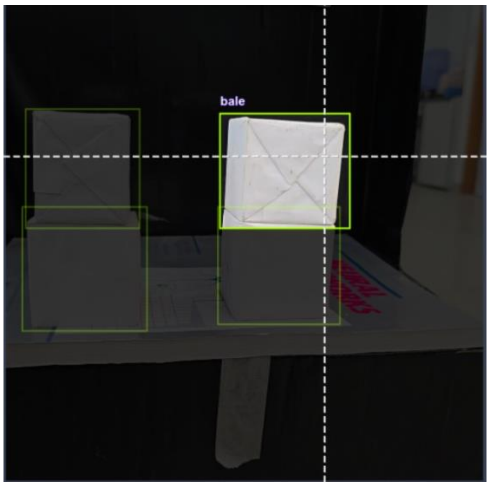


Figure 6: Example of labeled bale used for training.

The detection process utilizes an Intel RealSense depth camera, which provides both RGB and depth images. The YOLOv8 model, running on a GPU for reduced latency and higher frame rate, processes the RGB frames to detect bales based on the pre-trained dataset. The bounding boxes of detected objects are extracted, and the topmost and leftmost bale are selected for further processing.

Once the center of the topmost and leftmost bale is detected, the 3D position of the pixel with respect to the camera is calculated using the camera's intrinsic parameters and the corresponding pixel's depth. The equations that describe the system are:

$$x_{3D} = \frac{x_{pixel} - c_x}{f_x} \cdot z_{pixel}$$

$$y_{3D} = \frac{y_{pixel} - c_y}{f_y} \cdot z_{pixel}$$

$$z_{3D} = z_{pixel}$$

The position (x_{3D}, y_{3D}, z_{3D}) is given with respect to the optical frame. The optical frame uses the z-axis pointing forward, the x-axis to the right, and the y-axis downwards. The position is then included as a child of the "camera_optical_link" in the ROS2 TF-tree, which can be seen in Figure 7. This transformation allows the system to determine the bale's position relative to the robotic manipulator's base, enabling precise grasping and placement of the bales during the transport operation.

Detected bales are further grouped into stacks based on their horizontal alignment. The system selects the topmost bale from the leftmost stack to be picked by the manipulator.

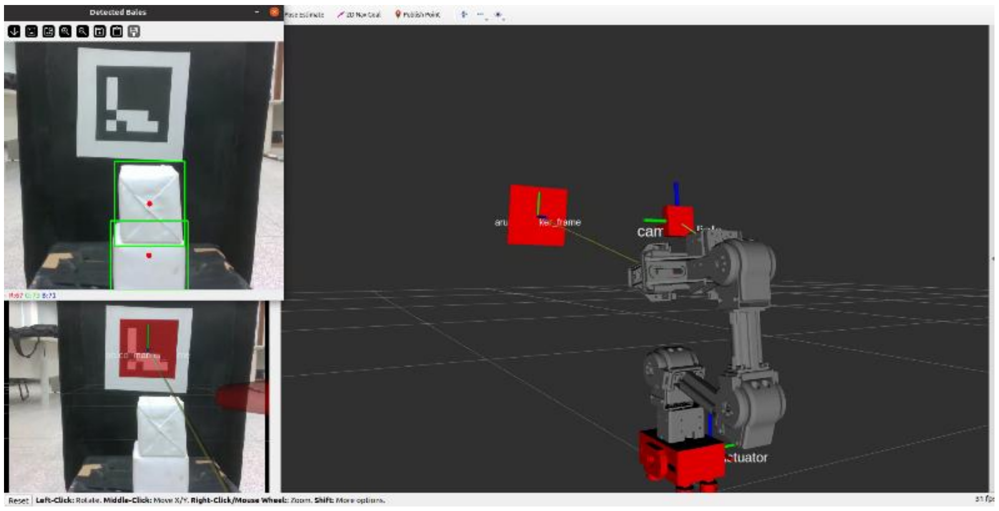


Figure 7: Representation of the use of the Open Manipulator with the YOLO Bale Detector and ArUco Detection.

E. ArUco Recognition

Once Hercules picks the first bale, it is required to place it on top of the truck at a specified position. To accurately determine the position in the truck, an ArUco (Garrido-Jurado et al., 2014) marker is positioned on top of the platform and the bales must be placed at a determined pose relative to the marker. Figure 8 shows examples of ArUco markers.

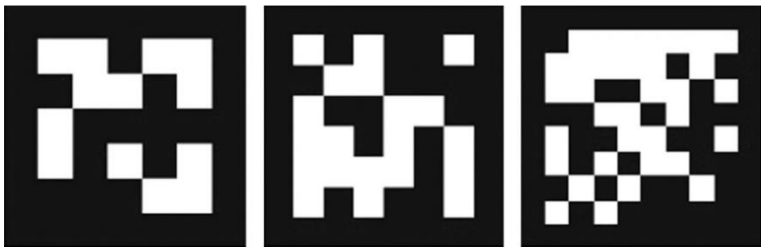


Figure 8: Representation of the use of the Open Manipulator with the YOLO Bale Detector and ArUco Detection.

ArUco markers are fiducial markers that allow accurate 3D pose recognition using an Intel RealSense D435i depth camera attached to the robotic arm. It was used the aruco_ros package (Bence Magyar et al, 2024), which integrates ROS with an OpenCV ArUco recognition. As a result, the package publishes a frame transformation from the coordinate system placed on the camera, denoted “camera_link”, to the coordinate system at the position of the ArUco Marker, shown as “aruco_marker_frame” in Figure 7.

After the first bale is placed on top of the truck, the ArUco marker is no longer used, and the model uses the placed bale as reference for putting the next bale on top.

F. Manipulator and Step Motor Controller

The Manipulator inherently has 4 degrees of freedom. However, its movements are limited by software to be like a forklift. The robotic arm is mounted over a step motor rail, which enables it to move sideways, also to resemble the structure of the industrial forklift.

The first step for picking up or delivering a load is adjusting the step motor so that the Manipulator is aligned with the bale. To do so, the position of the bale is detected using the Bale Recognition module and the distance (in meters) that the manipulator is required to dislocate horizontally is determined by the Manipulator and Step Motor Controller.

The step motor movement command is sent via a CAN interface to an ESP-32 installed at Hercules for interacting with its sensors and actuators. The microcontroller determines the number and directions of steps to be taken by the mechanism and sends the signal to the Step Motor. The system repeats the alignment once again to ensure precision.

After that, the Bale Recognition is used a second time, and the Manipulator and Step Motor Controller computes the required upwards and forwards movement for correctly grasping the bale. The path and trajectory computed for the Manipulator is calculated using the open-source OpenManipulatorX software to send commands to an OpenCR board installed at Hercules, which sends the voltage signals to the manipulator’s servo motors.

G. Human-Robot Interaction System

As shown in Figure 3, Hercules incorporates a Human-Robot interaction system seamlessly integrated with CARMEN LCAD. The main goal of this module is to enable Hercules to receive and understand commands given by individuals with no expertise in robotics, using natural language via speech. This module achieves this by utilizing speech-to-text (STT) and text-to-speech (TTS) neural models, which are integrated with a Large Language Model (LLM).

The Human-Robot Interaction system within Hercules follows a structured flow of operation that begins when a person communicates to the robot verbally. The Human-Robot Interaction system in Hercules works as the following structured process. First, verbal communication from a person prompts the robot to transform speech into text via STT. This text is then sent to the LLM, pre-instructed through few-shot learning prompts about Hercules' tasks. The LLM comprehends the command and forwards it to the Trigger Module and TTS model. The Trigger Module assesses feasibility and, if feasible, communicates with CARMEN LCAD for navigation and task execution. Simultaneously, the LLM’s response is converted into voice by the TTS model, enabling the robot to communicate back to the user. This flow is depicted in Figure. 9.

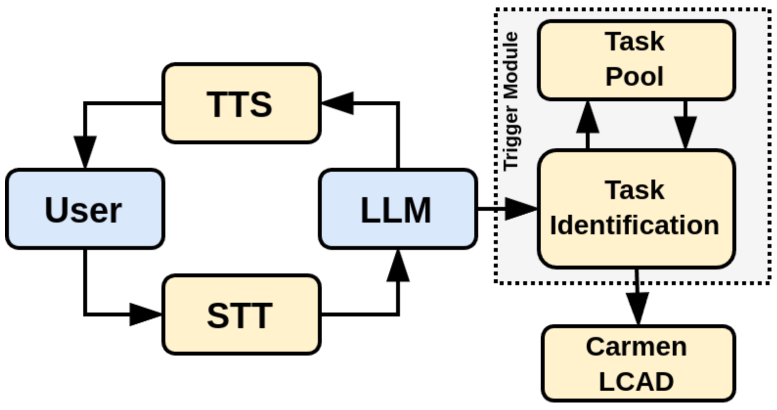


Figure 9: Human-Robot Interaction System flow.

The Human-Robot Interaction System is accessible via a web-based interface for users to communicate with Hercules. It is agnostic to the specific architecture or model used for Text-to-Speech (TTS), Speech-to-Text (STT), and Large Language Model (LLM) functions.

H. STT

The STT component was implemented using Whisper (Radford et al., 2022), an advanced open-source model tailored for Automatic Speech Recognition (ASR) tasks. Trained on a comprehensive dataset that includes over 680,000 hours of audio in various languages, it offers efficient performance across a wide range of accents and can effectively handle background noise. We utilized the whisper small variant, which comprises 244 million parameters. This version demonstrated strong ASR capabilities for Brazilian Portuguese and provided rapid transcription, essential for maintaining conversational fluidity.

Other STT models were evaluated by the authors, such as Kaldi and DeepSpeech. However, they showed suboptimal results for Brazilian Portuguese.

I. TTS

The available Text-to-Speech (TTS) models for Brazilian Portuguese are not as developed as their Speech-to-Text (STT) counterparts. The open-source TTS models either lack support for Brazilian Portuguese or exhibit substandard performance. For this reason, we utilized Amazon Polly (Werchniak et al., 2021), a proprietary online technology designed for human-like speech synthesis. It enables the transformation of written text into clear audible speech, offering high-quality vocal output that closely resembles human speech, including Brazilian Portuguese.

J. LLM

We configured the Large Language Model (LLM) with specific prompts to enable it to act as a robot for stacking and unstacking bales in a warehouse. Then, we developed a response pattern to efficiently process the results from the LLM's inferences. Essentially, the LLM is programmed to generate three distinct types of responses based on its input:

● RESPONSE TO THE USER: <sentence>: this type is used when the model provides a generic response to the user, such as when the user asks about the robot's name after introducing themselves. It is activated when there is no specific mission-related query.

● COMMAND: <command>: this type is used when the response indicates a mission for the robot to undertake, like when the user instructs the robot to unload a block at a specified position. It is activated when there is a clear mission outlined in the input.

● CONFIRMATION REQUEST: <sentence>: this type activates when the model requests confirmation of a command, like when the user instructs the robot to execute a mission. The model prompts the user to confirm the mission, and the subsequent response falls under the COMMAND type.

The raw answer is sent to the Trigger Module and the RESPONSE TO THE USER is sent to the TTS block to be played to the user, ensuring a smooth flow before vocalizing the answer to the user.

The system prompt (translated to English) consists of the following:

Your name is Hercules, and you are the virtual operator of a forklift in the company's warehouse. In the warehouse, there is a truck to where the bales need to be moved with move_pile_to_truck. You must follow the instructions to operate the forklift. To do this, you can invoke the following commands:

MOVE_PILE_TO_TRUCK: moves the cargo from the truck to aisle in stack;

If the user's speech is not a command, respond normally.

If the user's speech contains a command, you must respond as follows:

USER RESPONSE: spoken response to the user before executing the command

COMMAND: command to be invoked

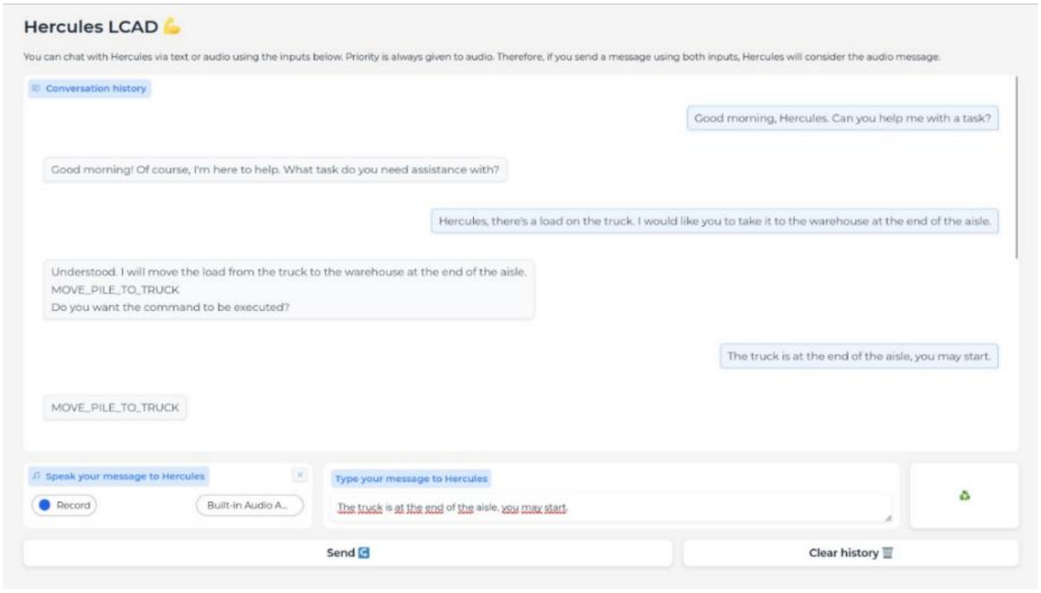
CONFIRMATION REQUEST: Do you want the command to be executed?

You must invoke only one command at a time.

If the user confirms the command, respond only with:

"COMMAND: command to be executed"

An example of interaction with the Human-Robot Interaction System is presented in Figure 10.



Hercules LCAD 💪

You can chat with Hercules via text or audio using the inputs below. Priority is always given to audio. Therefore, if you send a message using both inputs, Hercules will consider the audio message.

Conversation history

Good morning, Hercules. Can you help me with a task?

Good morning! Of course, I'm here to help. What task do you need assistance with?

Hercules, there's a load on the truck. I would like you to take it to the warehouse at the end of the aisle.

Understood. I will move the load from the truck to the warehouse at the end of the aisle.
MOVE_PILE_TO_TRUCK
Do you want the command to be executed?

The truck is at the end of the aisle, you may start.

MOVE_PILE_TO_TRUCK

The truck is at the end of the aisle, you may start.

Figure 10: Human-Robot Interaction System

The model's response shown in the web chat interface is the result of a filter applied to remove the control tags output. This result is also the sentence sent to the TTS module to be played to the user. The raw response from the LLM in the case shown was the following:

USER RESPONSE: Understood. I will move the load from the truck to the warehouse at the end of the aisle.

COMMAND: MOVE_PILE_TO_TRUCK

CONFIRMATION REQUEST: Do you want the command to be executed?

K. Trigger module

The trigger module processes the LLM's response to determine the robot's task, especially if it involves executing a command. It analyzes contextual information from the LLM, extracting and defining the specific task based on the response. Using regular expression methods, it extracts the command and assesses if it should trigger a task. The LLM does not generate code for the robot; instead, all tasks are predefined as missions available to the Task Manager, represented as a Task Pool in Figure 9. If the extracted command corresponds to a valid task, it is forwarded to the Task Manager for execution, represented by the CARMEN LCAD block in Figure 9.

As explained in the previous modules, in industrial controlled environments, the sequence of operations necessary to complete the tasks assigned to the forklift operators is fully described by hercules_mission.txt. The mission is made available to the robot as the single task in the Task Pool, being called MOVE_PILE_TO_TRUCK. Whenever the LLM confirms this command, the mission is initiated in the Task Manager.

The architecture is proposed with a Task Pool to allow flexibility of implementation and scalability for other robotics application. The accuracy of the models to identify the desired action in situations where extra actions are available is evaluated in one of the experiments presented in the next section.

EXPERIMENTS

We conducted experiments to evaluate our proposed architecture's feasibility and performance in two phases. First, we assessed the human-robot interaction system's efficiency. Next, we tested the fully integrated robot in a real-world scenario resembling a cellulose bale warehouse with a Task Pool containing only the MOVE_PILE_TO_TRUCK mission. All experiments were conducted in Brazilian Portuguese, reflecting the robot's intended use among Portuguese speakers[3].

A. Human-Robot Interaction

This phase assessed the Human-Robot Interaction System's effectiveness in a hypothetical warehouse with five aisles and ten cellulose bale stack locations per aisle. The aisles and stack locations were numerically identified from 1 to 5 and 1 to 10, respectively.

B. Experimental setup

To conduct this experimental phase, various Language Models were used and evaluated.

We evaluated five LLMs which were popular and performant at the time of the experiment: Llama 2 chat 70B and Llama 2 chat 13B, open-source models developed by Meta (Hugo Touvron et al., 2023); Mixtral-7x8B, an open-source model from Mistral AI; and OpenAI's advanced models for chat tasks, GPT-3.5-Turbo (Mann et al., 2020) and GPT-4 (Achiam et al., 2023). These LLMs were chosen for their ability to understand user queries, recognize commands, and produce appropriately formatted responses.

Performing inference of these models on Hercules is impractical due to their computational demands. Instead, they are accessed via Wi-Fi through API calls, with open-source models running on local high-performance computers. For Llama 2 and Mixtral, the deployment was on a system equipped with 3 NVIDIA A100 GPUs, utilizing vLLM (Kwon & Woosuk, 2023), an efficient LLM serving system designed to minimize KV cache memory waste and to enable flexible sharing of KV cache within and across requests, thereby enhancing the throughput. GPT-3.5-Turbo and GPT-4, on the other hand, are proprietary models and were used via OpenAI's chat completions API. All models were configured with a temperature setting of 0.0 and prompted to respond in the format outlined in Section human-interaction-system.

To simulate a warehouse scenario, the Task Pool comprised two possible actions:

- LOAD(x, y): Move the topmost bale from aisle x, pile y to the truck
- UNLOAD(x, y): Move a bale from the truck to the top of aisle x, pile y

C. Experimental results

To evaluate the Human-Robot Interaction System, we assessed the LLMs' performance with various inputs to the robot[4]. Two datasets of common commands that the robot may receive were created, and the responses generated by the models were assessed.

The first dataset includes feasible tasks, representing actions the robot is expected to execute. In contrast, the second dataset consists of unfeasible tasks, denoting actions the robot must decline as it cannot perform them. Examples of feasible and unfeasible tasks:

- Feasible: Hello Hercules, can you take the bale in aisle 2 and pile 3, and put it in the truck?
- Unfeasible: Hello Hercules, can you take the bale in aisle 10 and pile 3, and put it in the truck?

It is worth noting that while both tasks are similar, the second task is unachievable because the simulated warehouse lacks an aisle 10 – it only includes aisles numbered from 1 to 5.

[3] Code available at https://github.com/i2ca/tts-stt

[4] The performance results of the TTS and STT models are not included for two primary reasons: (1) their error rate is quite low, consistently nearing 100% accuracy; (2) their influence on the overall effectiveness of the proposed system is quite low compared to the impact of the LLM.

Our evaluation started with testing the models on feasible tasks. Each model underwent 30 standard feasible questions, with 60 different follow-up responses — evenly split between 30 approvals and 30 disapprovals. For each question, the expected model response involved suggesting a command for task execution and requesting user confirmation. If approved, the corresponding task was activated; otherwise, the robot initiated a new interaction cycle to determine the correct task. An example of an interaction with a feasible task is illustrated below:

- User: Hey Hercules, could you take the cellulose placed in the 3rd pile in aisle 4?
- Hercules: Hello, for sure. I'm going to take the cellulose in aisle 4 and pile 3. Do you confirm this request?
- User: Yes, do it!

Note that the confirmation serves as a security mechanism to prevent the robot from executing incorrect requests. The model must wait for human approval to generate an action command following the rules previously described -- for example, COMMAND:

LOAD(4, 3). If the user disapproves the request, the model cannot generate a task command.

In Table 1, it is presented the performance for each model, considering both approval and disapproval. An approval pipeline is correct if the model generates the accurate task command after positive human feedback. Similarly, the disapproval pipeline is considered correct if the model does not generate a task command after negative human feedback. In Table 1 is shown that the overall performance is better for the disapproval pipeline than the approval one, which is expected since denying a command is generally easier than extracting the correct answer from a natural language specification. GPT-4 achieves the best performance, with 100% accuracy in the approval pipeline and 93.3% in the disapproval pipeline. Among the open-source models, Mixtral stands out, with 53.3% accuracy in the approval pipeline and 90.0% in the disapproval pipeline.

Table 1: Performance of each LLM considering feasible tasks for both approval and disapproval pipelines. All values represent the percentage of correct interactions.

Pipeline	**Model**				
	GPT-3.5-turbo	**GPT-4**	**Llama-2-13b-chat**	**Llama-2-70b-chat**	**Mixtral-8x7b-instruct**
Approval	96.7%	100%	26.7%	6.7%	53.3%
Disapproval	53.3%	93.3%	70.0%	70.0%	90.0%

Next, we evaluated the models' capabilities considering the unfeasible tasks. All models were evaluated with 19 questions, and refusals to perform the tasks were considered correct answers. Consequently, answers that resulted in commands to the robot are labeled as incorrect. In Table 2 is shown the performance of the LLMs for the unfeasible tasks. Remarkably, GPT-4 stands out once more, successfully identifying unfeasible tasks in 83.3% of the requests, followed by Mixtral at only 23.3%.

Table 2: Performance of each LLM considering unfeasible tasks. All values represent the percentage of correct interactions.

Metric	**Model**				
	GPT-3.5-turbo	**GPT-4**	**Llama-2-13b-chat**	**Llama-2-70b-chat**	**Mixtral-8x7b-instruct**
Accuracy	13.3%	83.3%	0%	0%	23.3%

D. Discussion

The experimental results show varying performance of the LLMs across different evaluation scenarios. For feasible tasks, GPT-4 emerged as the top performer, displaying a higher percentage of correct interactions in both approval and disapproval pipelines, reflecting its sophistication and effectiveness in generating accurate task commands following user feedback. Notably, among the open-source models, Mixtral presented promising results, highlighting its competitive performance in this context.

However, challenges emerged when evaluating the models' capabilities with unfeasible tasks. Despite this complexity, GPT-4 demonstrated a notable success rate, correctly identifying unfeasible tasks in the majority of requests. While dealing with unfeasible tasks is inherently more complex, GPT-4's performance in this context showcased a notable advantage over other models. Mixtral presented better performance compared to the Llama models — which failed to identify all unfeasible tasks —, but it was much lower when compared to the GPT-4. These findings underscore the nuanced nature of language understanding in robotics applications and emphasize the significance of robust language models for effective human-robot interactions.

E. Real-World Emulation of a Cellulose Bale Forklift Robot

The goal of this experiment phase was to demonstrate Hercules' capabilities in a simulated real-world setting, utilizing the entire system, including autonomous navigation and human-robot interaction.

A laboratory room was modified to simulate a warehouse, featuring cellulose bales and a platform serving as a truck for unloading. This setup allowed us to evaluate Hercules' ability to navigate autonomously, engage in natural language interactions with users, and effectively handle cellulose bales, providing insights into its overall functionality and practical utility.

The simulation utilized GPT-4 as the LLM, which achieved the best performance in the experiments, along with the TTS and STT described earlier. Hercules was configured to engage in conversations with individuals and activate a single task upon identification of the request. The task consisted of moving a pile of two bales from the warehouse to the truck, as described in the section 3.

The robot can initiate the task from any random location within the map and execute the following steps: (1) Navigate to the pile location; (2) Retrieve the topmost bale; (3) Proceed to the unloading area; (4) Unload the bale; (5) Return to the rest position. The result of one of these experiments was documented in a video available as supplemental material[5].

Overall, Hercules successfully completed the task, demonstrating its seamless navigation in the simulated warehouse environment and its effective response to user-initiated commands. Through voice and natural language interactions, Hercules accurately identified requests and initiated tasks from random locations within the map. This indicates promising results for the integrated Navigation and Human-Robot Interaction Systems.

CONCLUSION

In this paper, we developed and evaluated Hercules, an autonomous bale handling robot with advanced navigation and human-robot interaction systems. Using Text-to-Speech (TTS), Speech-to-Text (STT), and Large Language Models (LLMs), Hercules effectively responds to natural language commands and operates in dynamic settings. Experiments showed GPT-4 as the top-performing LLM, followed by Mixtral. Hercules successfully executed voice-based tasks in a simulated warehouse environment, demonstrating its practical utility. Future fields of improvement include reducing model latency for better conversation, refining prompts to improve the LLM performance. With the further developments in LLM research, newer models have surpassed the models used at these studies and future projects should observe an accuracy gain when using open-source models like DeepSeek-v3 and llama-3.2 and proprietary ones such as GPT-4o and Gemini-2.0.

ACKNOWLEDGEMENTS

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – (CAPES, Brazil) – Finance Code 001; Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq; and Fundação de Amparo à Pesquisa do Espírito Santo (FAPES, Brazil) - grant 2021-07KJ2.

[5] The video is available on https://drive.google.com/file/d/1S_YY728cVe0ZI0z2gd5wSdTuoRPzd-NB

REFRENCES

[1] Achiam, J, et al. (2023). GPT-4 technical report. arXiv.
[2] Alatise, M. B., & Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods (Vol. 8). IEEE.
[3] Alec Radford, et al. (2022). Robust Speech Recognition via Large-Scale Weak Supervision. arXiv:2212.04356.
[4] Badue, C. (2021). Self-driving cars: A survey (Vol. 165). Elsevier.
[5] Benotsmane, R., Dudás, L., & Kovács, G. (2020). Survey on new trends of robotic tools in the automotive industry. Springer.
[6] Bernardo, R., Sousa, J. M., & Paulo JS, G. (2022). Survey on robotic systems for internal logistics (Vol. 65). Elsevier.
[7] Bucker, A., Figueredo, L., Haddadinl, S., Kapoor, A., Ma, S., & Bonatti, R. (2022). Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers.
[8] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision. arXiv:2212.04356.
[9] Werchniak, A., Chicote, R. B., Mishchenko, Y., Droppo, J., Condal, J., Liu, P., & Shah, A. (2021). Exploring the application of synthetic audio in training keyword spotters. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing, 7993–7996.
[10] Cavalcante, T. G, et al. (2021). Visual Global Localization Based on Deep Neural Netwoks for Self-Driving Cars.
[11] Chen, D., & Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations (1st ed., Vol. 25).
[12] Coronado, E., Mastrogiovanni, F., Indurkhya, B., & Venture, G. (2020). Visual programming environments for end-user development of intelligent and social robots, a systematic review (Vol. 58). Elsevier.
[13] Fasola, J., & Mataric, M. J. (2012). Using socially assistive human--robot interaction to motivate physical exercise for older adults (8th ed., Vol. 100). IEEE.
[14] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 2280-2292.
[15] Guizzo, E., & Ackerman, E. (2012). The rise of the robot worker (10th ed., Vol. 49). IEEE.
[16] Hugo Touvron, et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.
[17] Jiang, A. Q., et al. (2023). Mistral 7B. arXiv:2310.06825.
[18] Kwon, & Woosuk. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180.
[19] Lin, K., et al. (2023). Text2motion: From natural language instructions to feasible plans. arXiv:2303.12153.
[20] Ma, Y., Wang, Z., Yang, H., & Yang, L. (2020). Artificial intelligence applications in the development of autonomous vehicles: A surve (2nd ed., Vol. 7). IEEE.
[21] Mann, B., et al. (2020). Language models are few-shot learner. arXiv:2005.14165.
[22] Morales, A., Asfour, T., Azad, P., Knoop, S., & Dillmann, R. (2006). Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands.
[23] Murphy, R. R., et al. (2007). Search and rescue robotics. Springer.
[24] Rose, D. C., Lyon, J., de Boon, A., Hanheide, M., & Pearson, S. (2021). Responsible development of autonomous robotics in agriculture (5th ed., Vol. 2). Nature Publishing Group UK London.
[25] Shen, J., et al (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions.
[26] Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2023). Progprompt: Generating situated robot task plans using large language models.
[27] Tellex, S., Gopalan, N., Kress-Gazit, H., & Matuszek, C. (2020). Robots that use language (Vol. 3). Annual Reviews.
[28] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Llion, J., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need (Vol. 30).
[29] Vemprala, S., Bonatti, R., Bucker, A., & Kapoor, A. (2023). ChatGPT for robotics: Design principles and model abilities (Vol. 2).
[30] Wang, Y.-J., Zhang, B., Chen, J., & Sreenath, K. (2023). Prompt a robot to walk with large language models. arXiv:2309.09969.
[31] Werchniak, A., et al. (2021). Exploring the application of synthetic audio in training keyword spotters. 10.1109/ICASSP39728.2021.9413448

[32] Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. arXiv:2305.05658.

[33] Zucker, M., Joo, S., Grey, M. X., Rasmussen, C., Huang, E., Stilman, M., & Bobick, A. (2015). A general-purpose system for teleoperation of the DRC-HUBO humanoid robot (3rd ed., Vol. 32). Wiley Online Library.

[34] Joseph Redmon, et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640.