Mastani, Fetch for Me

Eman Nabeel, Zaeem Mohtashim Khan, Sulaiman Ahmed

***Abstract*—We present Mastani: Fetch, a novel robotic framework leveraging large video language models (LLMs) for real-time multimodal command interpretation and autonomous object retrieval. The system is implemented on an NVIDIA Jetson Nano platform integrated with a robotic arm, Intel RealSense depth camera, and LiDAR sensor, forming a compact and cost-effective perception and actuation suite. At the core of the architecture lies a two-tier LLM pipeline: (1) Gemini 2.0 Flash Lite handles multilingual audio command processing, converting spoken instructions into actionable semantic representations and object targets; (2) Gemini 2.0 Flash Vision Language Model (VLM) then performs environment scanning analysis to detect and localize relevant objects. The output is coupled with a lightweight autonomous navigation module—Go2Goal—which executes the physical fetch-and-retrieve behavior. This approach uniquely integrates audio-language understanding, visual perception, and motion planning in a decentralized pipeline while maintaining real-time performance. Our system supports commands in multiple languages (supported by Gemini) and demonstrates robust object identification and path planning in cluttered indoor environments. Mastani: Fetch exemplifies a novel and effective application of foundation models in embodied AI, pushing the boundary of interactive, intelligent, and low-cost service robotics.**

I. Introduction

HUMAN-ROBOT interaction (HRI) is entering a transformative phase, driven by the integration of large language models (LLMs) into robotic systems. Traditionally, programming robots required domain-specific knowledge and complex interfaces, limiting accessibility and responsiveness. However, recent advancements in natural language processing (NLP) and the emergence of general-purpose LLMs—such as OpenAI's ChatGPT—are enabling a paradigm shift. These models offer intuitive, zero-shot or few-shot learning capabilities that allow robots to interpret and execute unstructured human language commands with minimal prior task-specific programming.

A compelling example of this integration is ROSGPT [1], a ROS2-based framework that uses prompt engineering and ontology-based structuring to translate natural language into executable robotic commands. The system demonstrates how ChatGPT can serve as a middleware layer between human intent and robotic execution, converting phrases like "Go to the kitchen and bring me water" into JSON-serialized ROS commands. Such developments highlight the feasibility of using LLMs as robust translation brokers between humans and robotic platforms.

Industry insights also reflect this momentum. According to Louis Dumas, CEO of Inbolt [2], LLMs are reducing friction between end-users and robotic systems by enabling "real-time decision-making and flexibility on the factory floor" through natural language interfaces. Similarly, Acrome's research [3] demonstrates that integrating LLMs into control loops not only improves usability but introduces the potential for real-time adaptive behavior and mission planning.

This paper explores the architecture, capabilities, and limitations of using LLMs for controlling robots through natural language. By examining systems like ROSGPT and comparing them to emerging industrial applications, we outline the foundational strategies—prompt engineering, ontology design, and hybrid architecture—that are propelling HRI toward more general, adaptive, and user-friendly interfaces. Ultimately, we argue that the fusion of LLMs with robotic systems is not just a technical innovation—it represents a step toward democratizing robotics, empowering non-experts to program and interact with robots in profoundly human ways.

II. Literature Review

A. Large Language Models for Robotics: A Survey

[4] The integration of large language models (LLMs) into the domain of robotics marks a transformative shift in how intelligent systems perceive, reason, and interact with their environments. This survey paper by Zeng et al. provides a comprehensive and timely review of the evolving synergy between LLMs and robotics, underscoring how advancements in natural language processing have begun to reshape the capabilities of autonomous systems. The authors trace the development of LLMs—from early models like Eliza and LSTM to more sophisticated architectures such as GPT-4 and PaLM-E—highlighting their escalating capacity for semantic understanding, multimodal processing, and decision-making. By embedding LLMs into robotic frameworks, new paradigms in human-robot interaction, task execution, and adaptive behavior are being realized. The literature reviewed showcases several key robotic systems enhanced by LLMs, such as PaLM-SayCan for task decomposition and LM-Nav for language-driven navigation, which collectively demonstrate the ability of LLMs to interpret complex instructions, plan multi-step actions, and adjust dynamically to contextual feedback. The paper also evaluates emerging Transformer-based architectures like RT-1 and RT-2, which show promising generalization across diverse robotic tasks without requiring fine-tuning. The authors underscore the benefits of LLMs in robotics, including enhanced natural language interfaces, personalized and multimodal interaction, and advanced reasoning capabilities. Nevertheless, they also provide a critical lens on unresolved challenges such as the high computational costs of LLM deployment, the need for diverse and multimodal datasets, and ethical concerns regarding safety, privacy, and the potential societal implications of embodied intelligence. Overall, the literature reviewed in this survey illustrates not only the technical potential of LLM-driven robotics but also calls attention to the multidisciplinary dialogue needed to address the socio-technical complexities of their integration.

B. ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS [1]

The intersection of large language models (LLMs) and robotics has opened transformative avenues for human-robot interaction (HRI), and Koubaa's ROSGPT framework represents a pioneering contribution in this domain. In contrast to earlier approaches that often relied on rigid command structures and pre-programmed responses, ROSGPT introduces a dynamic and natural communication interface between humans and ROS2-based robotic systems by leveraging the capabilities of ChatGPT. This work situates itself within a growing body of research exploring the utility of LLMs—like GPT-3 and GPT-4—in enabling intuitive and context-aware robot behavior through natural language. Drawing inspiration from prior studies that utilized LLMs for task planning, dialogue systems, and language grounding, ROSGPT advances the field by implementing a real-time, ontology-guided translation mechanism from human commands to structured, executable robotic actions. Through ontology-based prompt engineering and JSON-formatted command schemas, ROSGPT overcomes the brittleness and ambiguity often associated with natural language interpretation. The system's architecture, comprising the GPTROSProxy for prompt handling and ROSParser for command execution, exemplifies a modular and scalable design that can be extended to diverse robotic tasks beyond navigation. Notably, the paper underscores the few-shot learning capabilities of ChatGPT, demonstrating how minimal examples enable generalization to previously unseen instructions—an attribute highly valuable in dynamic environments. While comparable frameworks like RobotGPT offer theoretical explorations, ROSGPT distinguishes itself through its open-source implementation and rigorous testing. The author's experimental observations also highlight critical issues like hallucinations and the necessity of domain-specific ontologies to constrain LLM behavior effectively. This literature underscores an emerging consensus: that the fusion of LLMs and robotic middleware such as ROS is a promising path toward achieving more intelligent, adaptable, and user-friendly robotic systems. As such, ROSGPT not only builds on foundational NLP and robotics work but also sets the stage for future research in multimodal AI, collaborative autonomy, and generalized robotic intelligence.

C. AI Robotics Case - Controlling Robots with LLMs (Large Language Models) [3]

The application of Large Language Models (LLMs) in robotic control represents a cutting-edge intersection of natural language processing (NLP) and embedded systems engineering. Recent literature has increasingly focused on the role of LLMs, such as ChatGPT, Gemini, and LLaMA, in interpreting human prompts and mapping them to precise robotic actions. This project exemplifies that convergence by integrating Google's Gemini LLM with ACROME's Smart Motion Devices (SMD), providing an efficient and modular framework for real-world human-robot interaction. Drawing on core concepts such as prompt engineering, function abstraction, and RESTful communication architectures, the system allows users to issue commands in natural language, which are then translated into pseudo-functions (e.g., linear movement, turn) and converted to actual robotic instructions via a Flask-based API on a Raspberry Pi controller. This approach is consistent with and extends upon prior work such as ROSGPT, where ChatGPT was used with ROS2 to transform unstructured commands into structured JSON payloads. Unlike some earlier implementations, this project places significant emphasis on hardware-level integration, leveraging SMD modules to bridge the semantic gap between human intent and electromechanical action. Notably, the use of pseudo-functions provides a semantic layer that simplifies interaction with the robot's API, enabling even non-technical users to control complex behaviors such as circular motion or conditional navigation (e.g., "move until you see an obstacle"). The project's architecture also reflects trends in the literature advocating for two-tiered systems—client-side LLM interpretation and robot-side execution—which enhances modularity, reusability, and real-time responsiveness. Furthermore, the incorporation of a user-friendly Streamlit interface for prompt entry underscores a growing interest in democratizing robotic systems through natural language. Overall, this work contributes a practical and scalable template for LLM-driven robotics, reinforcing the view that semantic understanding, function abstraction, and hardware-software co-design are central pillars of next-generation human-robot interaction systems.

D. Exploring the potential of LLMs in robotics: an interview with Louis Dumas [2]

The integration of Large Language Models (LLMs) into robotics is increasingly viewed as a pivotal development in bridging the gap between human intent and robotic execution. As Louis Dumas, CTO of Inbolt, explains in a 2024 interview, LLMs such as GPT, PaLM, and LLaMA provide a natural language interface that significantly lowers the barrier for human-robot interaction. These models enable users to issue complex commands using everyday language, thereby eliminating the need for specialized programming knowledge. This shift aligns with broader trends in the field, where LLMs are not only seen as tools for communication but also as accelerators for robotic system development—automating code generation, streamlining workflows, and facilitating rapid prototyping. However, Dumas emphasizes a crucial distinction: while LLMs enhance linguistic comprehension, they are inherently limited when it comes to executing or interpreting physical and spatial tasks. This has prompted the emergence of Vision-Language Models (VLMs) and Foundation Models, which combine the linguistic capabilities of LLMs with perceptual and contextual understanding, enabling robots to interpret their environment and act on commands like "pick up the blue cup." Despite these advancements, significant challenges remain. LLMs are resource-intensive, requiring powerful hardware and large datasets, and their use raises safety concerns when robots act on probabilistic language interpretations in unpredictable environments. As such, Dumas advocates for a cautious yet optimistic view of LLMs in robotics: while they revolutionize the interface layer, true autonomy will depend on multimodal systems capable of integrating vision, language, and physical reasoning. The literature suggests that the future of robotics lies in this hybridization—melding LLMs with perception-based AI to enable robust, context-aware, and user-friendly robotic systems.

E. Introducing Gemini 2.0: our new AI model for the agentic era [5]

In December 2024, Google DeepMind unveiled Gemini 2.0 Flash, a significant advancement in its AI model lineup, designed to usher in the "agentic era" of artificial intelligence. This model emphasizes enhanced multimodal capabilities, enabling it to process and generate text, images, and audio outputs natively. Gemini 2.0 Flash is tailored for real-time applications, offering low-latency responses and improved performance over its predecessors. Developers and trusted testers gained early access through platforms like Google AI Studio and Vertex AI, with broader availability planned for early 2025.

A notable feature of Gemini 2.0 Flash is its integration into various Google products, including the Gemini app, providing users with a more responsive and capable AI assistant. The model supports multimodal input, allowing it to understand and generate content across different formats seamlessly. Additionally, Google introduced experimental variants like Gemini 2.0 Flash Thinking, which focuses on reasoning by breaking down complex problems into manageable steps, enhancing the model's problem-solving abilities.

The development of Gemini 2.0 Flash aligns with Google's broader vision of creating AI models that can perform complex tasks with minimal human supervision. By incorporating features like native image and audio output, tool use, and improved reasoning capabilities, Gemini 2.0 Flash represents a step towards more autonomous and versatile AI systems.

In summary, Gemini 2.0 Flash marks a pivotal moment in AI development, offering enhanced multimodal processing, real-time responsiveness, and advanced reasoning capabilities. Its integration into Google's ecosystem and availability to developers signal a move towards more agentic AI applications, capable of performing complex tasks across various domains.

III. METHODOLOGY

A. Components and Design Choices

The robotic system is constructed using a suite of carefully selected hardware components optimized for real-time perception, navigation, and human-robot interaction. At its core lies the Jetson Nano, a compact yet powerful edge-computing module from NVIDIA that enables onboard execution of neural models, computer vision, and control logic with low latency. The system uses the Jetson Nano's built-in microphone for capturing voice commands; however, due to the inherent noise sensitivity and limited audio quality of onboard microphones, the design allows for seamless integration of external microphones to improve voice capture robustness. For visual perception, the robot is equipped with an Intel RealSense D435 camera, which provides both RGB and depth data essential for environmental mapping, object detection, and frame-based inference using a vision-language model (VLM). This camera facilitates six-frame panoramic scanning by rotating the robot to capture 60-degree segments of its surroundings. Spatial awareness and obstacle detection are supported by a 2D LiDAR sensor, which provides precise range measurements used for navigation, dynamic obstacle avoidance, and object alignment during the grasping phase. These components work in synergy under the control of the Jetson Nano, which manages data acquisition, model inference, and actuator control, forming a cohesive and autonomous robotic system.

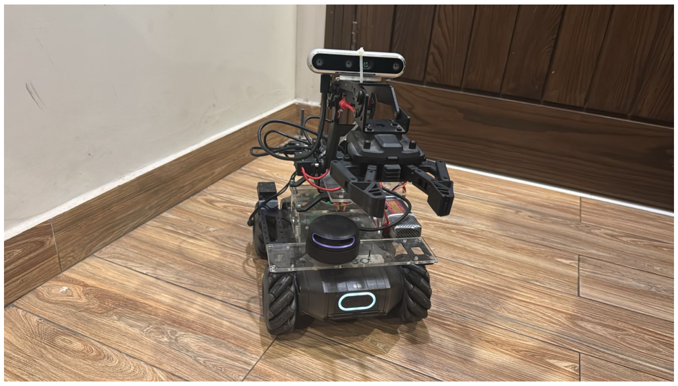


Fig. 1: Overall Setup

Initially, the system was designed to utilize the Jetson Nano's built-in camera for visual perception tasks, including environmental scanning and object detection. This decision was driven by the convenience of onboard integration and reduced hardware complexity. However, during early testing phases, it became evident that the built-in camera suffered from significant limitations in both frame rate and image quality. The low frame rate led to motion blur and delayed frame capture during panoramic scanning, while the poor resolution and color fidelity adversely affected the performance of the vision-language model (VLM), resulting in inconsistent object detection and frequent misinterpretations. These deficiencies directly impacted the robot's ability to scan the environment effectively and detect target objects reliably. As a result, the decision was made to transition to the Intel RealSense D435 camera, which offers high-resolution RGB-D imaging and superior frame rate performance. This upgrade significantly improved the reliability of both environmental mapping and object detection. Additionally, the Jetson Nano robotic arm, fixed to the base of the platform, remained a critical component for object manipulation throughout these upgrades, enabling precise pick-and-place operations after visual and positional alignment with the target object.

B. Audio-Based Command Input and Schema Generation

The system initiates its interaction cycle through a voice-based command acquisition module designed for natural and intuitive human-robot communication. This process is managed by the script capture_audio.py, which captures audio input using the device's default microphone and stores it in the WAV file format for further processing. Recognizing the

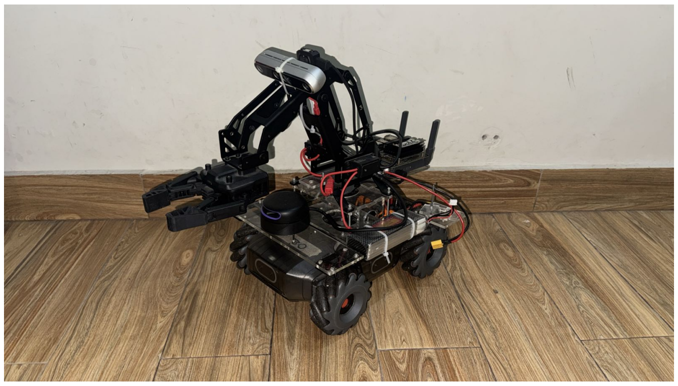


Fig. 2: Overall Setup

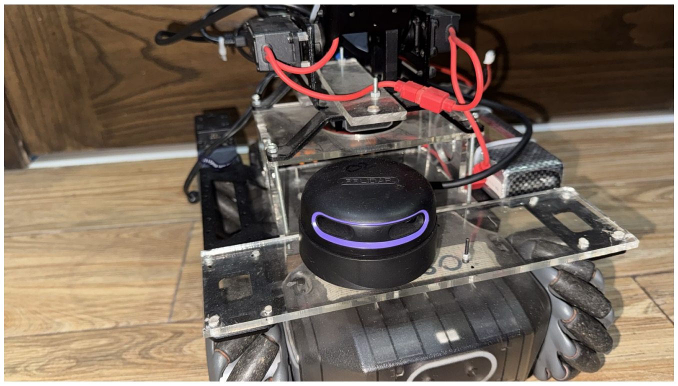


Fig. 3: Lidar

Fig. 4: RealSense Camera

susceptibility of in-built microphones to ambient noise and distortion—especially in uncontrolled environments such as laboratories or real-world deployment areas—the system also supports the use of pre-recorded audio files for consistent demonstration purposes. However, this limitation can be effectively mitigated in practical applications through the integration of a high-fidelity external microphone, thereby enhancing the clarity and accuracy of audio input. The captured audio is

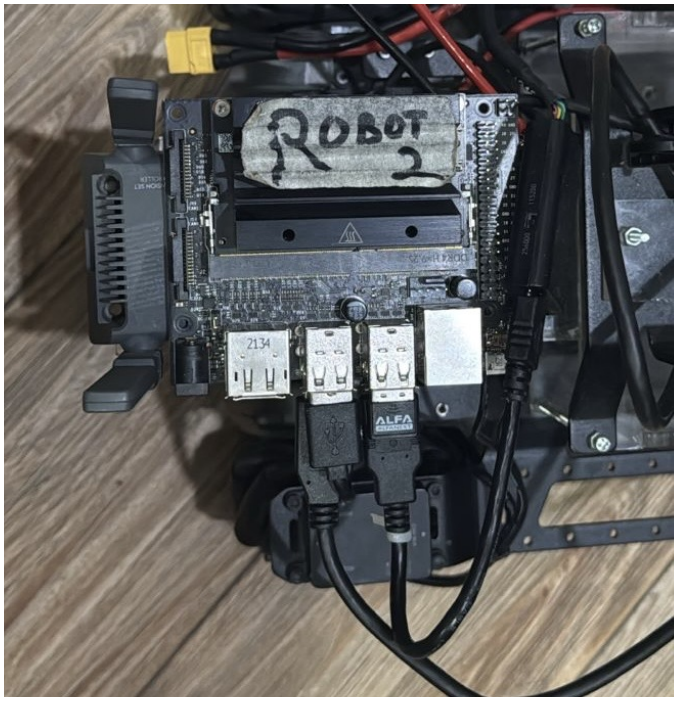


Fig. 5: Jetson Nano

processed using gemini_aud.py, which sends the WAV file to the Gemini 2.0 Flash Lite model. This large language and vision model (VLM) interprets the spoken instruction and extracts critical semantic elements, namely the script_name, object_name, and room_name. These elements are returned in a structured JSON format. This schema forms the fundamental task descriptor that drives the robot's subsequent decision-making, navigation, and interaction behaviors.

The following prompt is issued to the Gemini 2.0 Flash Lite model for interpreting spoken commands into structured robotic actions:

```
You are a robot command interpreter. The user will
    speak a command to the jetson nano robot.
Convert that audio command into structured JSON in
    English with the following format:

{
"commands": [

    { "type": "scanobj", "params": { "object": "..."
         , "warehouse number": ...} },
]
}

Only respond with valid JSON. Do not explain.
```

Listing 1: Prompt for Gemini Audio Model

```
{
  "commands": [
    {
      "type": "scanobj",
      "params": {
        "object": "red can",
        "warehouse number": 2
      }
    }
  ]
}
```

Listing 2: Sample JSON Output from Gemini

C. Autonomous Navigation to Target Room with Obstacle Avoidance

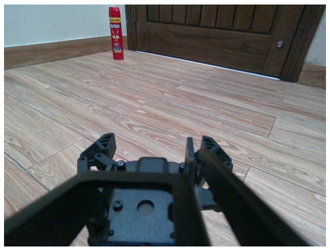
Upon successful schema extraction from the audio input, the robot proceeds to the locomotion phase, where it interprets the room_name from the JSON structure as a spatial goal within the environment. The robot employs a go2goal navigation algorithm that integrates obstacle avoidance mechanisms to safely traverse to the specified room. This navigation module utilizes a combination of sensor data and localization techniques to dynamically adjust the robot's path in real-time, ensuring collision-free movement even in the presence of static or dynamic obstacles. As the robot moves through potentially cluttered or human-populated indoor environments, it maintains an updated cost map and navigational trajectory using sensor fusion inputs, likely including LiDAR and inertial data. The robot determines its arrival based on localization estimates and map data, after which it prepares for environmental scanning. This transition marks the shift from spatial mobility to perceptual analysis, where the robot's primary objective becomes visual recognition of the specified object.

During the initial stages of development, the robot's navigation system was implemented using a unified go-to-goal algorithm with integrated obstacle avoidance, both for approaching the target warehouse and for returning to the drop-off zone after object retrieval. However, repeated tests revealed an unexpected behavior during the return phase—once the robot had picked up the object, it began to exhibit erratic motion and frequent halting. Upon closer inspection, it became evident that this anomaly was caused by a conflict between the obstacle avoidance system and the robot's updated LiDAR and odometry readings. After grasping the object, the LiDAR detected the object now held within close proximity as a persistent obstacle directly in front of the robot, causing the navigation logic to reactively attempt to avoid it. This internal conflict compromised the efficiency and stability of the return journey. To resolve this issue, the navigation pipeline was bifurcated into two distinct phases: the robot employed go-to-goal with obstacle avoidance while navigating toward the warehouse (pre-retrieval), and switched to go-to-goal without obstacle avoidance once the object was secured. This separation allowed the robot to treat the held object as part of its own frame, thereby ignoring it as an obstacle and ensuring a smooth, uninterrupted return to the drop-off zone.

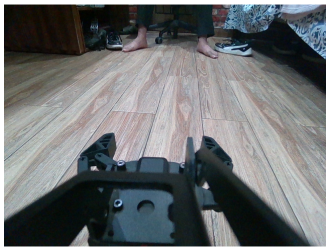
D. Environmental Scanning and Object Detection via Gemini VLM

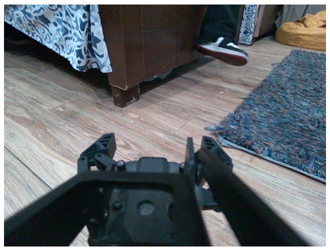
Following arrival at the target room, the robot initiates a comprehensive environmental scanning process aimed at detecting the object described in the command schema. This phase is primarily managed through the scripts check-camera.py, check-vision.py, and gemini_vlm.py. The robot, equipped with an Intel RealSense camera, captures six distinct frames by rotating in increments of 60 degrees, thereby achieving a 360-degree panoramic visual sweep of the environment. These frames are sequentially transmitted to the Gemini 2.0 Flash Lite model, which serves as the system's vision-language model (VLM). For each frame, Gemini performs multi-modal inference by correlating the visual content with the previously extracted semantic descriptors—script_name, object_name, and room_name. The output for each frame includes a boolean indicator flagging whether the target object is present in that frame. In the event of a positive detection, Gemini additionally returns the lower and upper HSV (Hue, Saturation, Value) bounds that characterize the object's appearance in color space. These HSV parameters serve as a critical input for color-based segmentation in the subsequent alignment and approach phase.

(a) Frame 0


(b) Frame 1

(c) Frame 2


(d) Frame 3


(e) Frame 4

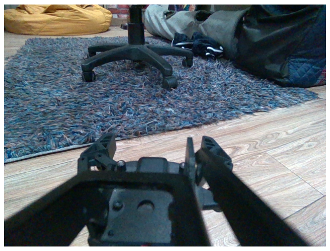

(f) Frame 5

Fig. 6: Six panoramic frames captured by the robot during a 360-degree environment scan.

```
You're an object detection model. Your goal is to:
- Respond if you see the mentioned object in the
    given picture. You should only respond with a '
    Yes' or a 'No'.
- If you see the mentioned object in the picture,
    return the lower and upper hsvs of the color of
    the object such that a segmentation mask can be
    effectively generated for it.
- If no object is detected in the frame then return
    [0, 0, 0] for lower_hsv and [0, 0, 0] for
    upper_hsv.

Return your answer strictly in the following JSON
    format:
{
  "is_object": "Yes" or "No",
  "lower_hsv": [H, S, V],
  "upper_hsv": [H, S, V]
}

The object to find is: red can
```

Listing 3: Prompt for Object Detection via Gemini VLM

```
{
  "frame_2.png": {
    "is_object": "No",
    "lower_hsv": [0, 0, 0],
    "upper_hsv": [0, 0, 0]
  },
  "frame_0.png": {
    "is_object": "No",
    "lower_hsv": [0, 0, 0],
    "upper_hsv": [0, 0, 0]
  },
  "frame_4.png": {
    "is_object": "No",
    "lower_hsv": [0, 0, 0],
    "upper_hsv": [0, 0, 0]
  },
  "frame_1.png": {
    "is_object": "Yes",
    "lower_hsv": [170, 100, 100],
    "upper_hsv": [180, 255, 255]
  },
  "frame_5.png": {
    "is_object": "No",
    "lower_hsv": [0, 0, 0],
    "upper_hsv": [0, 0, 0]
  },
  "frame_3.png": {
    "is_object": "No",
    "lower_hsv": [0, 0, 0],
    "upper_hsv": [0, 0, 0]
  }
}
```

Listing 4: Sample Multi-Frame Object Detection Output from Gemini VLM

E. *Precision Object Alignment and Grasping Using Visual and LiDAR Feedback*

With the object now visually confirmed and its HSV characteristics extracted, the system enters a fine-tuned alignment and object acquisition phase. This operation integrates both vision-based and LiDAR-based feedback to accurately position the robot relative to the object. The robot uses the HSV bounds received from Gemini to generate a binary color mask from the camera feed, which isolates the object of interest against its background. Through contour detection and centroid localization, the robot calculates the object's position within its field of view. Simultaneously, LiDAR sensors continuously scan the environment to provide depth data, enabling the robot to correct for both rotational and translational misalignments in its approach trajectory. A feedback control loop ensures that the robot iteratively adjusts its orientation and lateral displacement until the object is directly in front of it at a suitable range. The robot then advances toward the object, halting when a predefined LiDAR threshold indicates proximity sufficient for safe grasping. At this juncture, a robotic arm or actuator is activated to pick up the object, thereby completing the acquisition task with high spatial precision and minimal collision risk.

F. *Return Navigation to Drop-Off Zone via Odometric Localization*

Once the object has been successfully grasped, the robot transitions into the final phase of the operation—returning to the initial drop-off zone. This process relies heavily on the use of odometric tracking, implemented in the script odom.py. At the beginning of the operation, the robot logs its starting pose as the origin point (0, 0) in its coordinate frame. Throughout its navigation and object retrieval activities, the robot continuously updates its odometric position using wheel encoders and inertial measurements. When the return phase is initiated, the robot computes the optimal path back to the origin by leveraging its accumulated odometry data, effectively retracing its trajectory or navigating through a pre-mapped environment. The return path also includes obstacle avoidance and trajectory correction to ensure a safe and efficient movement. Upon reaching the origin, the robot deposits the object at the designated drop-off zone, thereby completing the full cycle from command intake to successful task execution. The robot then resets its state, ready to accept and process a new command, facilitating seamless, multi-cycle operation in an intelligent human-robot interaction loop.

IV. RESULTS

The proposed autonomous robotic system was evaluated in a controlled indoor setting at the Electrical Engineering Department of Lahore University of Management Sciences (LUMS). The environment was arranged to simulate a real-world warehouse retrieval scenario. The testbed consisted of a defined drop-off zone as the starting and return location, and two designated warehouse zones—Warehouse A and Warehouse B—which served as the possible destinations for object retrieval tasks. Objects included boxes and bottles of different colours, ensuring that they are long and light enough for the arm to grab.

At the start of each experimental run, the robot was positioned at the drop-off zone and remained idle until it received a voice command from the user. These commands specified the target object and its warehouse location (either A or B). The audio input was captured using the Jetson Nano's onboard microphone and processed through the Gemini 2.0 Flash Lite model, which returned a structured JSON schema containing the object_name, script_name, and room name. These parameters directed the robot's behavior for the remainder of the task cycle.

Upon receiving the task parameters, the robot initiated navigation toward the specified warehouse using a go-to-goal navigation algorithm integrated with real-time obstacle avoidance. This ensured smooth traversal through the dynamic and sometimes cluttered corridors of the department. The robot successfully interpreted the room name from the Gemini schema and autonomously planned a path to the correct warehouse zone, adjusting in real-time for static and dynamic obstacles such as chairs, tables, and passersby.

Once the robot arrived at the designated warehouse, it began a 360-degree environmental scan using the Intel RealSense D435 camera. It captured six frames, each spaced 60 degrees apart, and transmitted these to the Gemini 2.0 VLM. For each frame, Gemini returned a boolean indicating whether the desired object was detected, and in the affirmative case, also provided the HSV (Hue, Saturation, Value) bounds for the object. These HSV values were then used to create a segmentation mask, isolating the object in the frame.

Using this mask and LiDAR depth data, the robot engaged in a precision alignment routine. It dynamically adjusted its orientation and lateral position to align with the detected object's location, and then moved forward until the object was within a predefined LiDAR proximity threshold. Upon reaching this distance, the robot executed a grasping sequence using the Jetson Nano robotic arm mounted on its base, successfully picking up the object.

To complete the task, the robot employed odometry-based localization to return to the original drop-off zone. The system logged its initial coordinates at (0,0) and, upon object retrieval, calculated the shortest path back using the recorded odometric trajectory. For the return path, the robot implemented go-to-goal navigation without obstacle avoidance, assuming a relatively unobstructed corridor during the return phase. This simplification allowed for a faster return cycle and minimized computational load.

Across multiple trials and object types, the system demonstrated high reliability in command interpretation, warehouse identification, object detection, and return navigation. The integration of Gemini for both audio and vision-language processing proved robust, enabling end-to-end autonomy with minimal human intervention. These results validate the feasibility of using VLMs and multimodal perception in practical robotics applications for indoor logistics.

V. Limitations

Despite the successful implementation and operation of the system, a few limitations remain. The use of the Jetson Nano's onboard microphone introduces sensitivity to ambient noise, which can affect the accuracy of voice command recognition in uncontrolled environments. This can be mitigated by integrating a higher-quality external microphone for more robust audio input. Additionally, the robot's manipulation capabilities are constrained by the physical limitations of the robotic arm. Specifically, the arm can only grasp objects within a certain weight threshold and dimensional range, limiting the system to handling small to moderately sized items. These constraints must be considered when selecting target objects for retrieval tasks.

References

[1] A. Koubaa, "ROSGPT: Next-generation human-robot interaction with ChatGPT and ROS," *Preprints*, vol. 2023, no. 202304.0827.v1, Apr. 2023, preprint. [Online]. Available: https://doi.org/10.20944/preprints202304.0827.v1

[2] Inbolt. (2024, jan) Exploring the potential of llms in robotics: An interview with louis dumas. Accessed: 2025-05-05. [Online]. Available: https://www.inbolt.com/resources/exploring-the-potential-of-llms-in-robotics-an-interview-with-louis-dumas

[3] Acrome Robotics. (2024, mar) Controlling robots with llms (large language models). Accessed: 2025-05-05. [Online]. Available: https://acrome.net/post/controlling-robots-with-llms-large-language-models

[4] M. Chen, W. Xu, Z. Liu *et al.*, "Teaching robots to follow high-level instructions by leveraging world knowledge," *arXiv preprint arXiv:2311.07226*, 2023, accessed: 2025-05-05. [Online]. Available: https://arxiv.org/abs/2311.07226

[5] Google DeepMind. (2024) Gemini 2.0 flash: Bringing the agentic era to ai. Accessed: 2025-05-05. [Online]. Available: https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/gemini-2-0-flash