Project Report: GPT-4o Spatial Reasoning for VEX AIM Robot

Course: 15-494/694 Cognitive Robotics, Spring 2026
Author: Shun Jin & Weison Huang

---

1. Problem Statement

Traditional robot navigation relies on algorithmic path planners (RRT, wavefront, A*, etc.) that compute geometric paths from coordinates. These work well but are rigid - they cannot handle open-ended spatial questions, natural language commands, or reason about the scene semantically.

This project investigates whether a vision-language model (GPT-4o), operating in an agentic multi-turn loop, can serve as a practical alternative for spatial reasoning and navigation planning. Instead of computing paths algorithmically, the robot shows its world map to GPT-4o and lets the model decide where to go, how to get there, and what obstacles to avoid - all from visual understanding of the map image.

---

2. Approach

We built a system that: 1. Captures the robot's world map (a top-down 2D view maintained by the vex-aim-tools framework) 2. Processes the map image into a robot-relative perspective (rotated so UP = robot's forward direction) 3. Sends the annotated image to GPT-4o with a carefully engineered system prompt that includes a fixed pixel-to-millimeter scale 4. Parses GPT-4o's response for either spatial answers or movement commands 5. Executes those commands on the physical robot 6. Repeats in a multi-turn agentic loop if GPT-4o needs to explore before committing

Pixel-to-Millimeter Scale: The world map viewer renders at a consistent resolution. We measured the grid spacing from captured images: each 100 mm grid square corresponds to 168 pixels. This gives a fixed scale of 1 pixel = 0.595 mm, which is hardcoded in the system prompt so GPT-4o can convert its pixel measurements to real-world distances without any per-round recalculation.

Use of the Course Framework (Built-In Components):

We want to be transparent about what comes from the vex-aim-tools framework versus our contribution.

From the framework: - FSM (finite state machine) infrastructure for program structure - Low-level motor control: Forward, Sideways, Turn nodes - World map maintenance: the framework's perception pipeline detects ArUco markers,

walls, barrels, etc. and places them on the map - Qt-based world map viewer (which we capture screenshots from) - Speech recognition (Hear transition) and text-to-speech (Say node) - OpenAI client setup and connection

Where built-in navigation is used: The framework provides built-in navigation nodes like PilotToObject (path planning + driving to a named object), TurnToward (orienting toward a map object), and PickUp. These are used only as the **final execution step** - after GPT-4o has already decided what action to take. For example, when GPT-4o outputs `#pilottoobject OrangeBarrel.a`, the framework's built-in pilot handles the actual driving. The key distinction is that GPT-4o does all the planning and decision-making (which object to go to, whether the path is clear, whether to explore first), while the built-in nodes only handle the last-mile execution. During exploration rounds, where GPT-4o issues `#forward` and `#turn` commands to gather more information, the movements are executed directly through basic motor control with no built-in path planning involved.

Our contribution is the high-level spatial reasoning and decision-making layer: - The `ExploreWorldMapGPT` node implementing the agentic multi-turn exploration loop - Image processing pipeline (rotation to robot-relative frame, robot center marking, cropping) - The fixed pixel-to-mm scale calibration (168 px = 100 mm) - System prompt engineering that teaches GPT-4o how to read the map, measure distances, avoid obstacles, and format commands - The interactive exploration protocol ([EXPLORE]/[EXECUTE]/[DONE] tags) - Command parsing and dispatch from GPT-4o's natural language output

---

3. Design Iterations

Iteration 1: Basic Single-Shot Query

What we tried: Send the raw world map screenshot directly to GPT-4o with a simple prompt asking it to answer spatial questions.

What happened: GPT-4o could identify some objects but had no consistent way to estimate distances. It would give vague answers like "the barrel is somewhat far away" without actionable measurements. It also used absolute image directions ("upper-left corner") instead of robot-relative language, which made command execution impossible.

Lesson learned: The model needs explicit guidance on how to interpret the image coordinate system and convert visual distances to physical measurements.

Iteration 2: Added Pixel-to-MM Conversion and Robot-Relative Framing

What we tried: We added image processing to rotate the map so UP always equals the robot's forward direction, and we provided GPT-4o with a pixel-to-millimeter scale factor.

What happened: Distance estimates improved significantly. GPT-4o could now say "the barrel is approximately 300 mm ahead and 100 mm to my right." However, the model still sometimes hallucinated objects that weren't on the map or confused similar-looking objects (e.g., two barrels of the same color).

Lesson learned: The image processing pipeline (rotation, scale) is critical. But GPT-4o's object recognition from a stylized 2D map is not as reliable as from a real photograph.

Iteration 3: Multi-Turn Exploration Loop (Agentic Approach)

What we tried: Instead of a single image-question-answer exchange, we implemented an agentic loop where GPT-4o can request exploration movements ([EXPLORE] tag) before committing to an answer ([DONE]) or action ([EXECUTE]). Each round captures a fresh map image after the robot has moved. This is the core idea of the project - giving GPT-4o agency to gather information iteratively, similar to how an LLM agent uses tools in a ReAct-style loop.

What happened: This significantly improved reliability for tasks that require seeing different parts of the environment. For example, if asked "how many barrels are there?" the robot can turn around to check all directions before answering. However, it sometimes explored unnecessarily when the answer was already visible.

Lesson learned: Giving the model agency over when it has "enough information" is powerful but needs budget limits to prevent infinite exploration.

Iteration 4: Detailed System Prompt Engineering and Obstacle Avoidance

What we tried: We wrote a comprehensive system prompt covering: - Exact procedure for pixel measurement (step-by-step) - Obstacle avoidance rules (minimum gap = 131 mm for robot body + clearance) - Command syntax with strict formatting requirements - Rules against using markdown, math notation, or bullet points in output

What happened: Command parsing became much more reliable. The robot could navigate around obstacles in simple environments. However, in cluttered

environments with many objects, GPT-4o would occasionally issue commands that drove into obstacles it failed to notice.

Lesson learned: Prompt engineering has diminishing returns. The fundamental bottleneck is GPT-4o's spatial perception accuracy from a 2D map image.

Iteration 5 (Final): Fixed Scale and Robust Image Processing

What we tried: We measured the world map viewer's actual pixel-to-mm ratio by examining captured images: each 100 mm grid square is consistently 168 pixels. We hardcoded this fixed scale (1 pixel = 0.595 mm) directly in the system prompt, eliminating the unreliable autocorrelation-based grid detection from earlier iterations. We also improved the robot center detection, added a red dot marker, and cropped the image to a relevant region around the robot.

What happened: Distance estimates became more consistent since GPT-4o no longer received a different (and sometimes wrong) scale factor each round. The cropped, centered image gives GPT-4o a clearer view of the immediate surroundings. This is the final version submitted.

---

4. Results

What worked well: - The agentic exploration loop demonstrates that LLM-based planning can work as an alternative to traditional path planning for simple navigation tasks - Robot-relative image processing (rotation so UP = forward) makes GPT-4o's spatial reasoning much more natural and accurate - The fixed pixel-to-mm scale gives GPT-4o a reliable measurement system - Multi-turn exploration is effective for tasks requiring a broader view of the environment - Simple navigation tasks (go to a specific named object, answer "where is X") succeed reliably in uncluttered environments - The command parsing and dispatch system robustly translates GPT-4o's text output into robot actions

What did not work well: - GPT-4o sometimes misidentifies objects on the stylized 2D world map (confusing barrels of similar color, missing small objects, hallucinating objects) - Distance estimation errors of 20-40% are common, since GPT-4o is still eyeballing pixel distances from the image even with a fixed scale - In cluttered environments, GPT-4o struggles to plan multi-step navigation paths that avoid all obstacles - The system is slow due to API latency (each GPT-4o vision call takes 3-8 seconds) - not suitable for real-time reactive behavior

---

5. Key Bottleneck: GPT-4o's Spatial Perception Accuracy

The primary limitation of this approach is that GPT-4o was not specifically trained for precise spatial reasoning from top-down 2D map images. While it can understand the general layout, it struggles with: - Precise pixel-level measurements (even with a fixed scale and step-by-step instructions, it still eyeballs distances) - Distinguishing between similar objects in close proximity - Consistently following the measurement procedure across multiple rounds

Potential improvement with Gemini: A stronger multimodal model such as Google's Gemini could potentially improve recognition accuracy, as it has demonstrated stronger performance on visual grounding and spatial reasoning benchmarks. We did not have time to fully evaluate this alternative, but it represents a promising direction for future work.

---

6. Future Work

1. **Stronger vision-language models:** Evaluate Gemini or future GPT models for improved spatial perception from map images. The bottleneck of inaccurate pixel estimation could be significantly reduced with a model that has better visual grounding.
2. **Hybrid approach:** Use the vision-language model for high-level planning and object identification, while reading object coordinates directly from the world map data structure for precise distance measurements. This would combine GPT-4o's semantic understanding with exact positional data.
3. **Few-shot examples:** Provide GPT-4o with annotated example images showing correct pixel measurements and command sequences to calibrate its spatial judgments.
4. **Real camera integration:** Instead of (or in addition to) the 2D world map, send the robot's actual camera feed to the vision model for richer visual information.

---

7. Conclusion

This project demonstrates that an LLM operating in an agentic loop can serve as a practical alternative to traditional path planning for robot navigation in simple environments. The key insight is that by giving GPT-4o a processed robot-relative map image with a fixed pixel-to-mm scale, and allowing it to explore iteratively before committing to actions, we can achieve natural language-driven navigation without writing any explicit planning algorithms. The built-in framework nodes handle only the final execution of GPT-4o's decisions, not the

planning itself. The main limitation remains GPT-4o's imperfect spatial perception from 2D map images - a gap that newer multimodal models may close.