

# Supplementary: Human-like Navigation in a World Built for Humans

Anonymous Author(s)

Affiliation

Address

email

**Abstract:** In the supplementary material, we provide more details on the simulation environment and ReasonNav’s implementation, along with additional experiments and visualizations. We begin with a deep dive into the simulation environments in Sec. 1. Next, we provide additional implementation details in Sec. 2, including hardware design (2.1), object detection parameters (2.2), and VLM prompts (2.3). Finally, we showcase additional visualizations in both simulation and real-world scenarios in Sec. 3, with an additional demo of multi-floor navigation (Sec. 4). Additionally, we provide a supplementary video, *supplementary\_398.mp4*, which includes better visualizations of our experiments.

## 1 Simulation Environment Details

Our simulation experiments were conducted using IsaacSim with customized robot and environment assets. The robot is controlled via ROS2 with the same topics/interfaces as the real-world robot to ensure easier sim-to-real transfer.

**Simulated Robot Setup** Our simulation robot asset starts with a URDF exported from Fusion 360 CAD. The core sensors and actuators, including base control, pan-tilt, and sensor data (RGBD camera, 3D Lidar, and 2D Lidar), are then simulated using IsaacSim action graphs. Coordinate frames/topics are also aligned with the real-world robot system.

**Simulated Environment Setup** We modify one of the provided IsaacSim hospital environments to include room numbers, signs, and human NPC actors for our task. They are arranged realistically in the scene as shown in Fig. 1. Specifically, we divided the hospital into several functional zones: consultation, support, examination, reception, waiting, public service, and stair areas. Each room within these zones was assigned a unique room number ranging from 3001 to 3041. Additionally, four prominent directional signs were placed at key locations to indicate the relative directions of each zone.

**NPC Information** To make the robot’s interaction with the environment more realistic, we classified NPCs into three types—doctors, nurses, and patients—each with its own ‘knowledge base’: **Doctors** know the exact locations of all doctors and approximate orientations of each functional zone. They can also answer medical questions and schedule consultations. **Nurses** know the exact locations of all doctors and all functional zones. They can also help with patient registration, check appointment times, and provide detailed directions to specific rooms. **Patients** know the approximate directions to the public service and waiting areas. Furthermore, every NPC can give relative route planning instructions (e.g., ‘The room is on your left’) when asked for directions and will respond ‘I don’t know’ if they are uncertain about the answer.

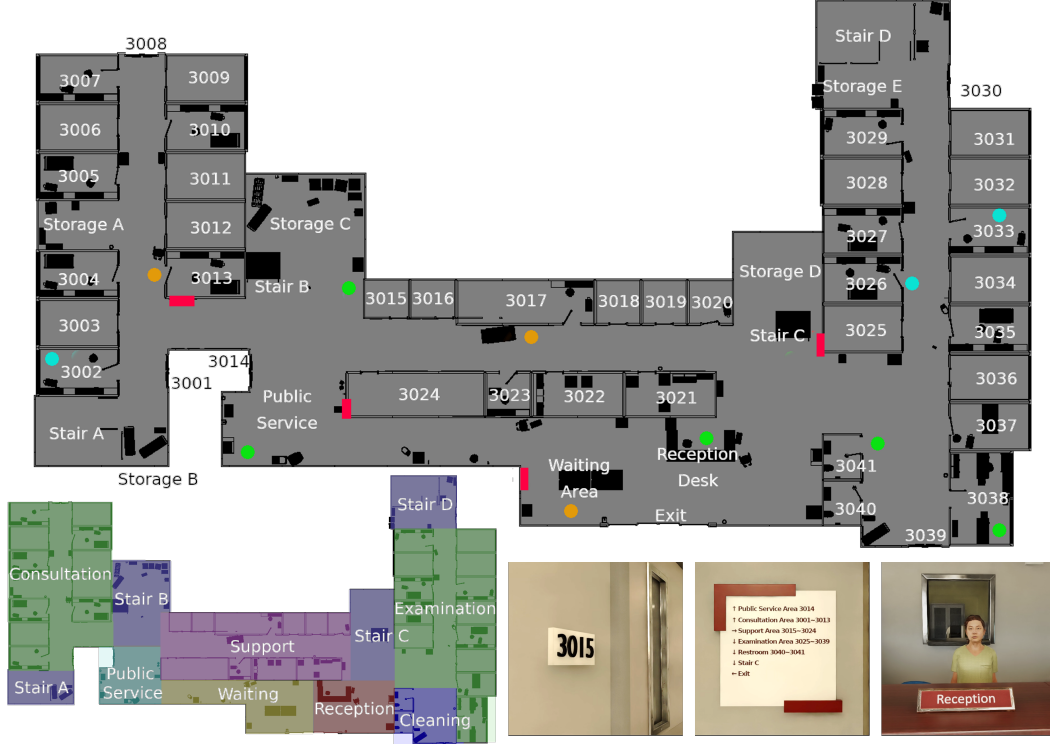


Figure 1: **Door Numbering, Signs and NPC Placement in Hospital Environment.** Blue dots represent doctors, orange are patients, and green are nurses. Red rectangles represent signs.

## 2 ReasonNav Implementation Details

### 2.1 Hardware Design Details

Our robot platform is custom-built to be both affordable and versatile. The four-wheel steering base allows for omnidirectional motion in flat indoor and outdoor spaces. A 7-DoF UFACTORY xArm7 is mounted at a height ideal for manipulation on tables/door handles in the real world. Using a laptop with a powerful GPU rather than an embedded ARM device like a Jetson also allowed for easier software setup and debugging. For sensors, both 2D and 3D LiDARs are used for obstacle avoidance and mapping; the two RealSense cameras for object detection and VLM perception; and the omnidirectional microphone for interacting with humans. The entire system is integrated using ROS2 and Docker for easy setup on any PC.

Overall, the system costs  $\sim \$35k$ . We plan on open-sourcing the entire CAD and hardware setup documentation in the future as well.

### 2.2 Object Detection Details

The detection runs on a 15 FPS stream of  $1280 \times 720$  RGB-D images from the RealSense camera. In the real-world, additional filtering is performed based on bounding box corners unprojected to 3D. For doors, the NanoOWL confidence threshold is 0.3, and we only keep detections with box widths in the range (0.5, 2.5) and heights in the range (0.5, 3). For room labels, the confidence threshold is 0.04, and the width range is (0, 0.4) and the height range is (0, 0.15). For directional signs, the confidence threshold is 0.03, and the width range is (0.35, 0.5) and the height range is (0.2, 0.5). For people, the confidence threshold is 0.3 and we do not do any other filtering. The unprojected box center is taken as the position of each object, and the approach orientation is acquired by taking the cross product of vectors from the center to unprojected points 20 pixels to the left and below. For each incoming frame, the filtered detections are greedily matched with existing objects based

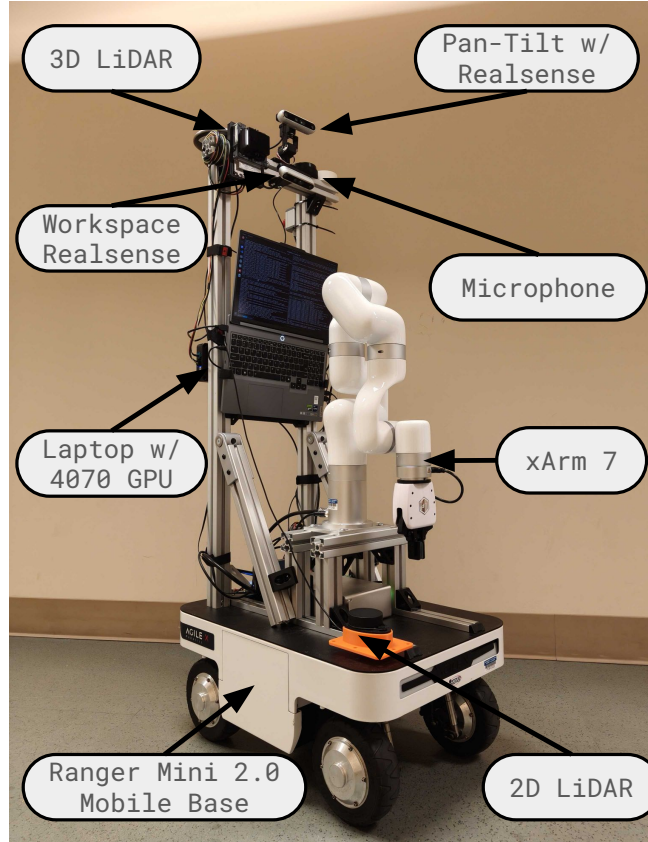


Figure 2: **Hardware System Overview.**

on center distance and aggregated if the distance is within 1m. Only the objects with 3 or more  
detections within the last 20 frames are added to the memory bank.

### 2.3 VLM Prompts

We provide the exact prompts for the various VLM calls we make through our framework: 1)  
Choosing the next landmark to visit given a top-down map image and JSON landmark information  
2) Reading the room number of a door given a real-world image 3) Reading a directional sign given  
a real-world image 4) Recording a note given the speech-to-text transcription of a human interaction.  
All of them use the same system prompt.

#### System Prompt

You are ChatGPT, a large language model trained by OpenAI. Follow the user's  
instructions carefully. Respond concisely, informatively, and helpfully. If you're  
unsure about an answer, say so. You have strong reasoning capabilities.

#### 1) Choose Next Landmark Prompt

You are a robot (🤖) trying to find target. You will receive an image of a map de-  
picting the environment you've explored so far with doors (🚪), signs (📍), people  
(😊), and frontiers (★). The map is North up, and the borders are labeled with  
cardinal directions N, S, E, W.

You will also receive JSON below that maps each landmark index on the map to  
its name and, if applicable, additional info such as directions (from signs).

Choose the next landmark to go to based on the provided map and JSON. Reason over how to most efficiently find the target, like asking a person nearby for directions, going to directions signs to narrow down regions, following ascending/descending door number patterns, visiting nearest doors to see their room number, or exploring frontiers to find more landmarks.

Visited doors are named in the format "Visited\_door\_X", where X is the room number. Use these room numbers to identify door numbering patterns. Use patterns efficiently. For example, if you are at door 100 and looking for door 110 and find there is an ascending pattern, skip doors 101, 102, etc.

Keep in mind that all information from signs, people, etc. is relative to the position of that particular landmark and is probably only applicable in a local region of that landmark.

If an unvisited person is nearby and you want to ask for more information on where to go, choose the person. Information provided by a visited person will be listed in 'info' using cardinal directions. Use this information. Example: Input: target='Room 3339', info='I should go North to find the door' Output: You will pick a landmark to the north of the person.

If directions are available in the JSON, follow those directions. This will take precedence over looking for things nearby. Make sure your range is correct. Example: Input: target='Room 3339', directions='North':['Room 3326-3340'], 'North-East':[], 'East':[], 'South-East':[], 'South':['Room 3101-3307'], 'South-West':[], 'West':[], 'North-West':[] Output: You will pick a landmark to the north of the sign because 3339 is in the range 3326-3340.

**\*\*Don't visit landmarks that are already visited.\*\***

First, think carefully step by step about where the target room might be and decide which landmark to visit next. Then, print out your reasoning followed by the chosen landmark index in brackets. Example: I am looking for Room 110. The directions from the sign say room 100-120 are south. Landmark 3 is an unvisited door to the south, so it might be the target; Chosen landmark: [3]

JSON: {landmark\_dictionary}


## 2) Read Door Label Prompt

What is the door number in the image? Return only the number and confidence score (to 2 decimal places), separated by a semicolon. If you can't see a sign or you can only read half of the sign, return -1; confidence score. If you can see the sign but you can't read the numbers, return -2; confidence score. Your confidence score should be between 0 and 1. Be more conservative with your estimates.

## 3) Read Directional Sign Prompt

Break down the directions sign you see in the image. Return your answer in the form 'left': [content], 'right': [content], 'forward': [content], 'backwards': [content]. Return only the dict (no comments or formatting).

## 4) Record Conversation Information Prompt

You are a robot  trying to find {self.goal}. You are facing {robot.facing}. Given the map and the directions by Person, write a short note for your future self to refer to later on how to reach your goal. Return only the output in the format "Note: note to self". Do not include quotations. The image is already aligned with relative directions, so left means left on the image. In your note to self, use the cardinal directions in the map. Also ignore the numbers above each landmark as they will be updated.

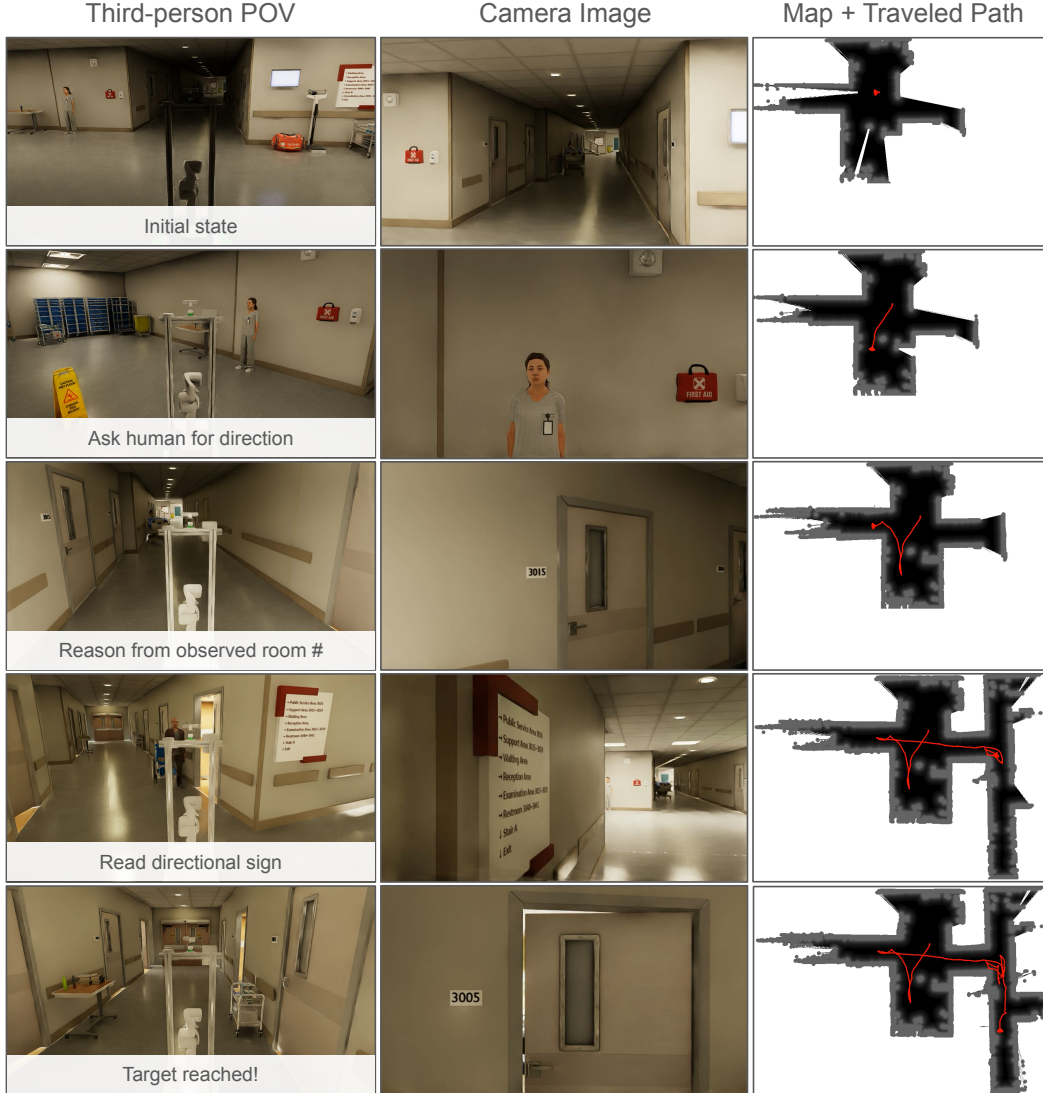


Figure 3: **Qualitative Simulation Results:** We present full step-by-step episode visualizations of our framework in simulation with exact path traveled highlighted in red.

124 Conversation: {conversation\_history}

### 125 3 Additional Visualizations

126 **Simulation Example** We present a step-by-step qualitative result for our simulation environment  
127 in Fig. 3, with more examples in the supplemental video.

128 **Step-by-step VLM reasoning example** We further showcase ReasonNav’s reasoning capabilities  
129 in a step-by-step example in Fig. 4. Given the landmark map (on the left), the VLM is prompted  
130 to choose a waypoint for the robot to follow. The text boxes (on the right) represent the VLM’s  
131 reasoning and decisions. Please refer to the video for more examples.



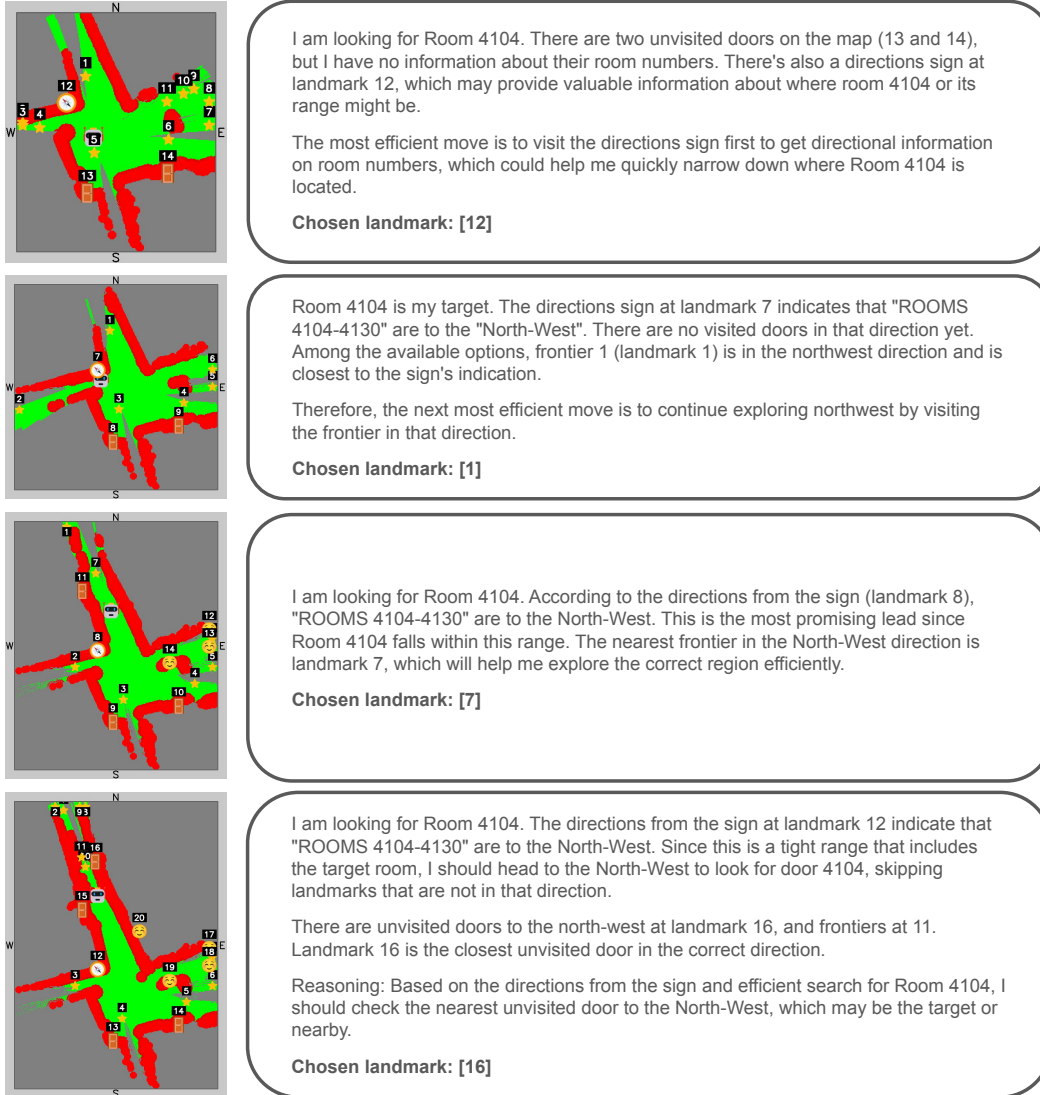


Figure 4: **Real-world VLM reasoning:** We present a step-by-step example of VLM’s reasoning and decisions to navigate to room 4104. ReasonNav exhibits spatial reasoning capabilities given the direction guidance from direction signs, as showcased in the third and fourth rows.

#### 132 4 Extension to Multi-Floor Scenarios

133 We show ReasonNav can also work with multiple floors through the elevator demo shown in the  
 134 supplementary video at the 2:00 timestamp. We choose a goal room on the fourth floor and initialize  
 135 the robot with the information that it is currently on the third floor and can take elevators. We add  
 136 elevators and elevator button panels to our detection module as well, and design a simple behavior  
 137 primitive to take the elevator. When the VLM chooses to go to the elevator, we detect the button  
 138 panel and use our onboard robot arm to push the up button. After the button is pressed, the robot  
 139 moves to face the door and uses the center depth values from the pan-tilt camera to sense when the  
 140 elevator door has opened based on a threshold. The robot then moves forward and asks a human  
 141 (assuming someone can help push the interior buttons) to push the button to go to the fourth floor.  
 142 Using the pan-tilt camera facing backwards, the robot once again waits till the elevator door has  
 143 opened before backing out. The SLAM and landmark maps get reset and the robot continues to  
 144 search for the door as usual on the new floor.