Aim My Robot: Precision Local Navigation to Any Object

Xiangyun Meng, Xuning Yang, Sanghun Jung, *Graduate Student Member, IEEE*, Fabio Ramos, *Member, IEEE*, Sri Sadhan Jujjavarapu, *Member, IEEE*, Sanjoy Paul, and Dieter Fox, *Fellow, IEEE*

***Abstract*—Existing navigation systems mostly consider "success" when the robot reaches within 1 m radius to a goal. This precision is insufficient for emerging applications where a robot needs to be positioned precisely relative to an object for downstream tasks, such as docking, inspection, and manipulation. To this end, we design and implement *Aim-My-Robot* (*AMR*), a local navigation system that enables a robot to reach any object in its vicinity at the desired relative pose, with *centimeter-level accuracy*. AMR achieves high accuracy and robustness by leveraging multi-modal sensors, precise action prediction, and is trained on large-scale photorealistic data generated in simulation. AMR shows strong sim2real transfer and can adapt to different robot kinematics and unseen objects with little to no fine-tuning.**

***Index Terms*—Vision-based navigation, learning from demonstration, visual servoing.**

I. Introduction

NAVIGATION is a foundational skill that unleashes robots from confined workspaces and lets them interact with the open world. While there has been significant progress in learning-based systems that can navigate without a geometric map [1], [2], [3], [4], [5], understand semantics [4], [6], [7], [8], [9], and follow human instructions [4], [8], [9], [10], [11], these systems typically consider the goal reached when the robot is within 1 m radius to the goal [8], [12]. This lax definition of success hinders their applicability to the growing need for mobile robots to navigate to objects precisely. For example, in a factory, a forklift must position itself correctly to a pallet so that the fork can be inserted into the pallet without collision (Fig. 1(b)); an inspection robot can more clearly read gauges when facing instruments perpendicularly at a proper distance.

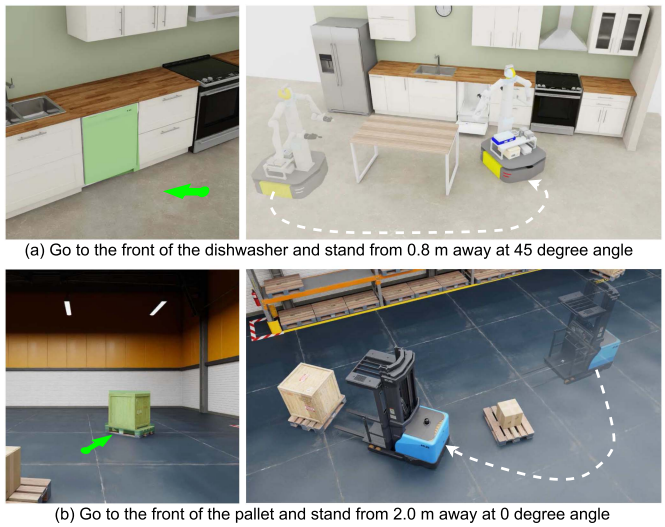

Fig. 1. Overview of AMR. Given a masked image describing the target object and an object-centric pose (relative position and orientation), AMR tracks the object while moving, avoids obstacles, and aligns the robot to the target object with centimeter-level precision without maps or object 3D models.

Similarly, a home robot needs to position itself properly to open a dishwasher (Fig. 1(a)), dock to a charging station, or place objects at accurate locations (e.g. *left* side of a table). Lack of precision in the final pose of the robot would incur failures due to collision, out-of-reach, or not adhering to task requirements.

Precision navigation requires a robot to understand the geometry of relevant objects (*where is the goal?*) and the local scene structure (*how to get there?*). One classical approach would be to estimate the object's pose, from which the desired robot pose can be derived. But this usually requires specific object information such as 3D models [13], and the object is initially visible. This limits its applicability when the object 3D model is unavailable, or the object is initially out of view. On the other hand, current learning-based navigation [2], [4], [5], [6], [7], [9], [14] is not quite applicable to real-world high-precision navigation tasks, as they lack precise goal conditioning, often assume discrete action spaces [2], [15], or trained on imprecise real-world data [5], [14].

In this work, we propose *Aim My Robot* (AMR), an end-to-end vision-based local navigation model that navigates to objects with *centimeter-level* precision. AMR does not require an object CAD model, and instead uses a reference image with an object mask and relative pose for precise goal specification. AMR takes streams of RGB-D and LiDAR data as inputs and outputs trajectories directly, eliminating the need of a metric map. We achieve high precision and strong generalization via two contributions:

Received 13 September 2024; accepted 27 January 2025. Date of publication 14 February 2025; date of current version 11 March 2025. This article was recommended for publication by Associate Editor D. G. Sorrenti and Editor A. Valada upon evaluation of the reviewers' comments. This work was supported by NVIDIA and Accenture. *(Corresponding author: Xiangyun Meng.)*

Xiangyun Meng and Sanghun Jung are with the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: xiangyun@cs.washington.edu; shjung13@cs.washington.edu).

Xuning Yang and Fabio Ramos are with the NVIDIA, Seattle, WA 98105 USA (e-mail: xuningy@nvidia.com; ftozetoramos@nvidia.com).

Sri Sadhan Jujjavarapu and Sanjoy Paul are with the Accenture LLP, Palo Alto, CA 94304 USA (e-mail: srisadhan.j@gmail.com; sanjoy.paul@gmail.com).

Dieter Fox is with the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA, and also with the NVIDIA, Seattle, WA 98105 USA (e-mail: fox@cs.washington.edu).

We use the word **precise** to indicate both low error and high consistency. This differs from the conventional meaning of "precise" which only refers to high consistency.

Website: https://sites.google.com/view/aimmyrobot.

Digital Object Identifier 10.1109/LRA.2025.3542323

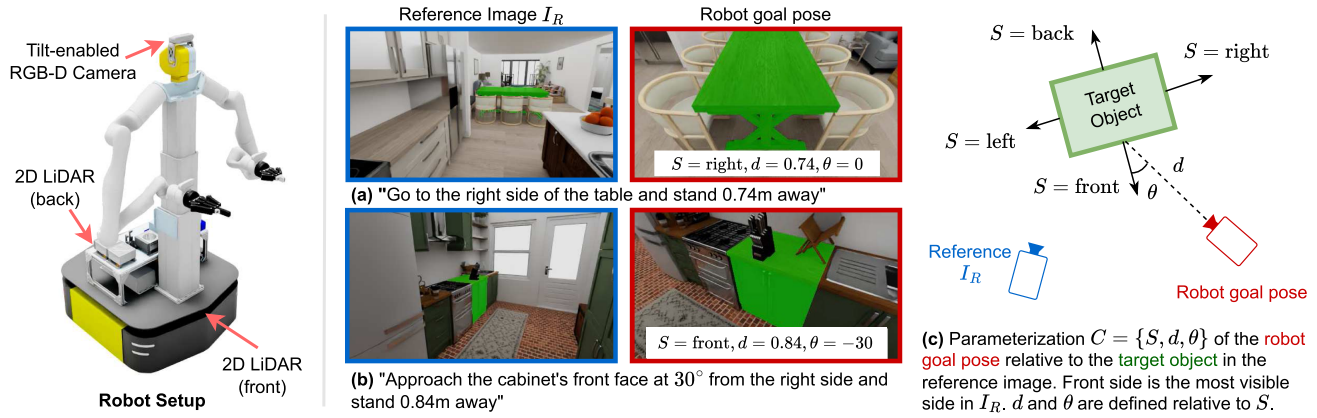

Fig. 2. Problem setup. We specify the target object via a reference image I_R taken in the scene and an object mask M (in green). The goal condition $\mathbf{C}$ is defined as the relative side and pose of the object in I_R. A robot needs to navigate to the object conditioned on $\mathbf{C}$ and tilt its camera to gaze at the object. Note the reference image *is not* the final image captured by the robot at the goal.

1) a data pipeline for generating large-scale, photorealistic, and precise trajectories to diverse objects; 2) a transformer-based model that takes multi-modal sensor inputs (RGB-D + LiDAR) and plan precise and safe trajectories. Experiments show that AMR achieves a median error of 3 cm and 1° on unseen objects, with little degradation when deployed on a real robot. It supports robots of different sizes and can be finetuned quickly to support more complex kinematics such as Ackermann steering vehicles. AMR is the first system for high-precision visual navigation with strong sim2real transfer. We hope it paves the way towards precision robot autonomy.

II. Problem Definition

We consider the scenario where the robot has reached the vicinity (<10 m, about a room size) of the object of interest and needs to perform a *local*, *object-centric* high-precision maneuver for docking, inspection, and manipulation. We formulate this as a local navigation task where the goal G consists of the target object and the relative base pose (in $SE(2)$) to the object. For generality, we assume G is provided by another system. The robot has a tilt-enabled forward RGB-D camera at 1.5 m high and a 2D LiDAR mounted on the base, providing 360° coverage (Fig. 2). Concretely, the robot is given past sensor observations $\{o_t, o_{t-1}, \ldots\}$ along with the goal G, and needs to move its base to reach the specified pose while tilting its camera to gaze at the object. The robot does not have the object's 3D model or a 2D/3D map.

Goal specification: Due to the lack of the object's 3D model and a map, one challenge is unambiguously defining the goal G. To address this, we assume the robot is given a reference image I_R with the target object mask M (Fig. 2(a) and (b)). I_R can be taken from the robot's long-term memory or recent observations. To make goal specification object-centric, we use the fact that common objects have four dominant sides (i.e. described by its bounding box). Denoting the most visible side in I_R as the *front*, then the relative pose can be defined as $\mathbf{C} = \{S, d, \theta\}$, where $S \in \{\text{front}, \text{back}, \text{left}, \text{right}\}$ is the approach side, $d \in [0.0\text{ m}, 1.0\text{ m}]$ is the approach distance, and $\theta \in \{0°, \pm 15°, \pm 30°\}$ is the approach angle. See Fig. 2(c) for an illustration. Hence, $G = \{I_R, M, \mathbf{C}\}$.

Discussion: Existing object instance navigation systems require taking a close-up view for each object or mapping object locations [8], [16], [17]. In comparison, our formulation uses *mask* to specify the object instance in an image. It does not require a close-up view or building a map. Moreover, it can be interfaced with a high-level task planner (e.g. SAM [18], TAMP [19] and LLM [20]) that outputs mask and pose parameters whereby AMR guarantees *precise* base and camera positioning.

III. Aim My Robot

In this section, we detail our technical approach. We first describe our data pipeline that generates large-scale, high-quality demonstrations entirely in simulation. Then, we introduce AMR, a multi-modal architecture for robust and precise local navigation.

A. Data Generation

Achieving strong generalization and high precision requires training data containing diverse scenes, objects, and precise trajectories. Since humans are poor at estimating distances and angles from images, we forgo teleoperation and leverage simulation and model-based planning for data collection.

Assets and simulator: We import the Habitat Synthetic Scenes Dataset (HSSD) [21] which contains diverse room layouts (100+) and objects (10,000+) into Isaac Sim. Since HSSD contains Physics-Based Rendering (PBR) textures, Isaac Sim is able to render photorealistic images using ray tracing (Fig. 3). The diversity and realism of the perception data enable strong visual sim2real transfer.

Sampling goal condition: We model the robot as a cylindrical rigid body with a radius R (i.e. its footprint) sampled from $[0.1\text{ m}, 0.5\text{ m}]$. The robot is randomly placed in the traversable area of a room. A reference image I_R is rendered either from the robot's initial camera view or from a random camera view in the room. Then, the target object mask M (obtained from the simulator) is randomly chosen from I_R. Non-objects such as walls and floors are excluded. We sample a goal specification $\mathbf{C}$ by randomly choosing a side $S \in \{\text{front}, \text{back}, \text{left}, \text{right}\}$, distance $d \in [0.1\text{ m}, 0.5\text{ m}]$ and angle $\theta \in \{0°, \pm 15°, \pm 30°\}$. Finally, we place the robot at the goal and check for collisions.

Trajectory generation: The robot kinematic model is assumed to be a differential-drive robot. We run AIT* [22] in OMPL [23]

Fig. 3. Example rendered images of HSSD scenes in Isaac Sim.

Fig. 4. Sample objects in the scenes. We consider all objects, including those that are not semantically labeled.

using the ReedsShepp state space with a turning radius 0 for planning the base motion. It is straightforward to change the state space to model robots with other kinematics. The cost function encourages the camera to look toward the goal while penalizing excessive backward motion (small backward motion is allowed if the robot can reach the goal faster). For each feasible path, we render the camera observations along the path with a distance gap of 0.2 m or an angular gap of 5°. The camera tilt angle is set such that the lowest vertex of the object mesh appears at $1/4$ above the bottom of the image (even when the object is out of view).

B. Model

Transformers have seen wide adoption in vision and robotics [2], [14], [18], [24] due to their excellent scalability and flexibility. To this end, we design AMR to be a transformer-based model (Fig. 5) based on two design principles: 1) a unified approach to representing multi-modal sensory data, and 2) an action decoding scheme that generates *precise* and *collision-free* actions. The system has three stages: multi-modal sensor encoding, goal-aware sensor fusion, and multi-modal motion generation. We detail each stage as follows.

1) Encoding Multi-Modal Sensor Data: We use RGB, depth, and 2D LiDAR observations to achieve high precision and robustness. RGB is used for visual reasoning, depth provides precise 3D geometric information, and LiDAR ensures a 360° coverage for robust collision avoidance.

Encoding RGB-D: The current RGB image I_t ($224 \times 224 \times 3$) is passed into a frozen (i.e. with locked weights) Masked Autoencoder (MAE-Base) [25] pretrained on ImageNet1K to obtain a $14 \times 14 \times 512$ feature map. Each depth image is resized to 14×14 and we compute the spatial location of each depth pixel in the robot's egocentric coordinate frame via $[x', y', z'] = \mathbf{R}\mathbf{K}^{-1}[x, y, d]^T + \mathbf{t}$, where $\mathbf{K}$ is the camera intrinsics and $[\mathbf{R}|\mathbf{t}]$ is the camera extrinsics. We apply a sinusoidal position embedding f for x', y', z', respectively, and concatenate them to obtain the position embedding for each depth patch as $[f(x'); f(y'); f(z')]$. We add depth position embedding to each corresponding RGB patch. To incorporate past observations I_{t-i}, we set the camera extrinsics $[\mathbf{R}|\mathbf{t}]$ to be the transform between the camera pose at $t - i$ and the robot base pose at time t, which can be obtained from the robot's odometry. Compared to concatenating RGB and depth tokens, which require twice the number of tokens, using depth as the positional encoding is more efficient since it keeps the number of visual tokens unchanged.

Encoding LiDAR: We resample the LiDAR points into 256 points. The points are grouped into 32 directional bins. The points in each bin (8 of (x, y) coordinates) are passed into a Multi-layer Perceptron (MLP) to obtain one LiDAR token. In total, there are 32 LiDAR tokens.

2) Goal-Aware, Robot-Aware Sensor Fusion: We tokenize the reference image with the same frozen MAE. The mask M is encoded by a shallow convolutional network, similar to [18]. All the visual tokens are flattened and passed to the Vision Context Encoder $\mathcal{F}$. The output tokens corresponding to the robot's observations are decoded into target masks using the mask decoder $\mathcal{G}$. This helps $\mathcal{F}$ to learn to track targets. We use separate MLPs to tokenize the goal d, θ, S and the robot footprint (i.e. radius) R. Finally, all the tokens are input to the Multi-modal Context Encoder $\mathcal{H}$. The output serves as the context for motion generation.

3) Motion Generation: The output of $\mathcal{H}$ is cross-attended to separate transformer decoders for the base movement and the camera tilt angle, respectively. AMR predicts base trajectory and camera tilt angle at different frequencies. For base trajectory, the robot follows the predicted trajectory up to T steps and then predicts a new trajectory given the updated robot observations. Multi-step prediction improves trajectory consistency and reduces unwanted oscillations. For the camera tilt angle, it predicts a new value at every time step.

Base trajectory decoding: The base trajectory is parameterized by a sequence of waypoints in egocentric polar coordinates. To capture the multi-modal nature of robot trajectories, we use an autoregressive transformer decoder. Since autoregressive classification requires discretizing the actions, to preserve the precision, we adopt multi-token classification with residual predictions. Specifically, we represent each waypoint by a sub-action tuple (direction $\psi_i \in [0, 2\pi]$, distance $r_i \in [0, 0.2\text{ m}]$, heading $\phi_i \in [0, 2\pi]$), $i = 0, \ldots, T-1$ and sample ψ, r, ϕ conditionally in sequence. This representation reduces the number of bins required for classification, as classifying ψ, r, ϕ at once would require a combinatorial number of bins. In practice, we discretize ψ, r, ϕ into 30, 32, 12 bins, respectively. For each output token z, we recover the continuous value $z' = C(z) + R(z, C(z))$ where $C(\cdot)$ is the output from the classifier, and $R(\cdot, \cdot)$ is an MLP that predicts the residual.

Camera tilt angle decoding: Since camera tilt control is uni-modal, we continuously predict the camera tilt angle α via regression at each time step t.

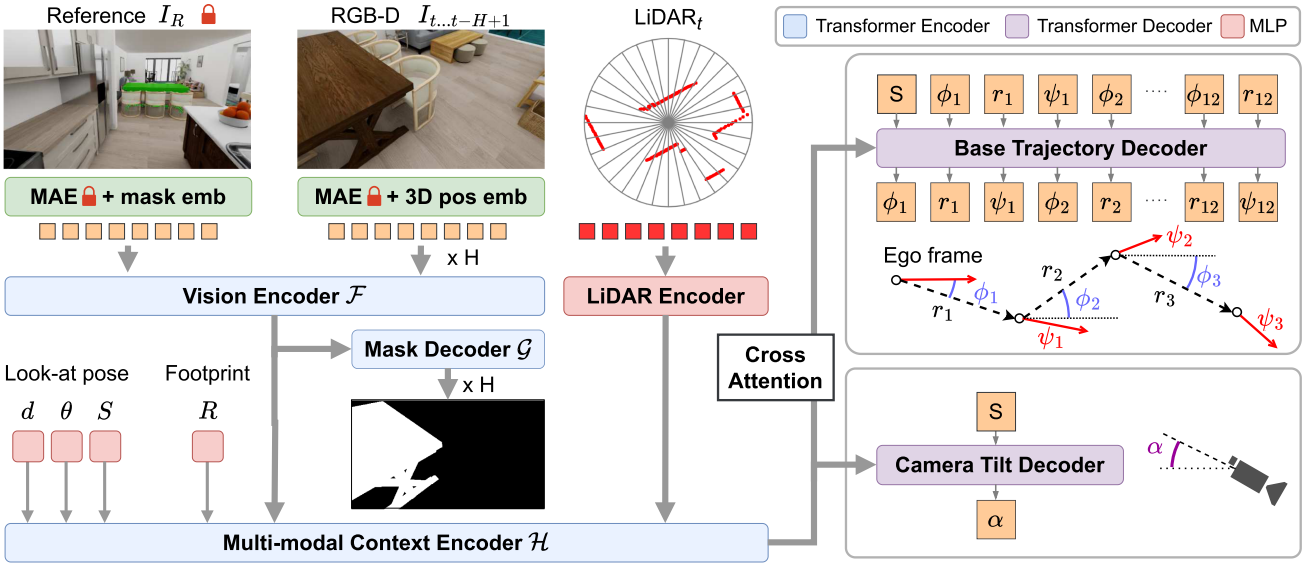

Fig. 5. Network architecture. The reference image I_R and the robot's RGB-D observations I_t are tokenized with MAE. The current LiDAR scan is tokenized by grouping points into directional bins. Image and LiDAR tokens are input into the multi-modal context encoder jointly with the look-at pose and footprint tokens. Finally, the output tokens of the context encoder are cross-attended to the base trajectory decoder and the camera tilt decoder. S are the learned start tokens for the decoders.

IV. Implementation Details

Dataset: We converted 54 HSSD [21] environments into Universal Scene Description (USD) format and generated 500 k trajectories (7.5 M frames) in Isaac Sim [26]. We use 49 scenes for training and 5 scenes for evaluation. There are 3,119 target objects in the training scenes and 275 target objects in the test scenes. Among the 275 test objects, 160 are unseen during training. We randomly sampled robot starting locations and target objects, and created 2000 navigation tasks in unseen environments for evaluation.

Training: We train the model with a supervised loss function:

$$L = L_{\text{mask}} + L_{\text{base}} + L_{\text{tilt}}$$

where L_{mask} is pixel-wise L2 loss to regress the object mask in all history frames, L_{base} is a sum of classification loss and regression loss for each waypoint, akin to [27], and L_{tilt} is an L2 loss that regresses the camera tilt angle. We train AMR for 150 k steps with a batch size of 128 on 8 A100 GPUs. After the initial Behavior Cloning phase, we ran the model in simulation and identified failures (collision, tracking loss, imprecise reaching) and used DAgger [28] to augment the dataset. For experiments with HSSD, we trained all models without history (*hist* $= 0$), as we found that history did not show noticeable benefits in simulation. In our real robot experiment, we trained a model with 4 past frames and compare it against *hist* $= 0$ model.

Deployment: We deploy AMR on a real mobile manipulator with an omnidirectional mobile base and a simulated forklift with Ackermann steering in Isaac Sim. Each trajectory contains $T = 12$ waypoints, separated by $dt = 0.2$ s. To track the predicted trajectories, we use a Pure Pursuit controller [29] for the omnidirectional robot and a Model Predictive Control (MPC) controller for the forklift. The camera tilt angle is updated at every time step, whereas the trajectory is updated with a new prediction at every 8 steps. The models run at 12 Hz on an RTX 3090.

V. Experiments

We perform **quantitative experiments** in simulation and the real world. Each run performs closed-loop inference until the robot comes to a stop, collides with obstacles, or exceeds the time limit. We consider a run *complete* if the robot is within 1.0 m to the goal. Otherwise, we consider the run incomplete. The error distribution contains the final pose error for only the completed runs, and the *completion rate* is reported as a separate metric. In real-world experiments, we measure the ground truth goal pose against an occupancy map with 2 cm resolution. The robot's ground truth pose is obtained by classical Monte Carlo Localization [30].

We conduct additional **qualitative experiments** to highlight generalization to novel tasks using AMR: We show generalization to new tasks and scenes in the real world, closing a fridge drawer using the robot body and a forklift picking up a pallet.

A. Simulation Results With HSSD

Baseline: We compare our method against a classical pipeline based on object pose estimation. We use FoundationPose [13] to estimate the pose of the target object in the initial observation given the CAD model. The posed bounding box is used to compute the goal pose as described in Section II. Then, we run the same planner to find a path using a groundtruth occupancy map and navigate the robot to the goal. Note that such a system uses extra information not required by AMR: 1) a CAD model of the target object; 2) the object is visible in the robot's initial observation (required by pose estimation); and 3) perfect mapping.

Quantitative Results: Fig. 6 compares the error distribution between AMR and the classical approach. We consider two cases: *visible*, where the initial observation is used as the reference image, and *invisible*, where the object is initially out-of-view. The classical approach can only be used in the *visible* case. In the *visible* cases, AMR performs better than the baseline

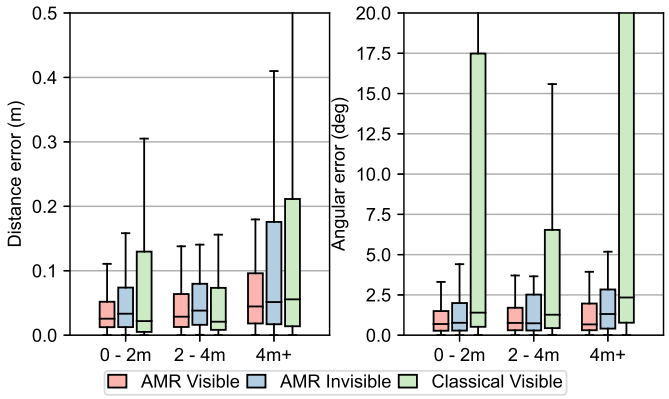

Fig. 6. Navigation error distribution in the test scenes for various object distances for completed runs. AMR outperforms the classical baseline when objects are initially visible or out-of-view, with a higher completion rate. Note that classical pose estimation requires the object to be initially visible.

TABLE I
COMPARISON OF MEDIAN ANGULAR ERRORS (MAE) AND MEDIAN DISTANCE ERRORS (MDE) FOR SEEN AND UNSEEN INSTANCES ACROSS SHARED CATEGORIES IN THE TEST SCENES

Category	**Seen** (136 unique objects) MAE (°)	MDE (m)	**Unseen** (166 unique objects) MAE (°)	MDE (m)
Chair	1.45	0.05	1.08	0.04
Drawers	0.77	0.02	0.55	0.03
Couch	0.82	0.02	0.57	0.04
Picture	0.84	0.03	0.64	0.04
Shelves	0.65	0.07	0.68	0.02
Tables	7.14	0.04	1.20	0.04
Unlabeled	0.78	0.03	0.74	0.04

approach even when the object is far. In the *invisible* cases, our approach outperforms the baseline approach with similar accuracy and significantly lower variance. One interesting observation is that the angular error for the baseline increases when objects are too close ($0-2$ m). This is usually caused by large objects (e.g. furniture) being partially out of view. In contrast, AMR actively moves the robot, making it more robust to state estimation errors.

In Table I we compare the median pose errors between the seen objects and unseen objects in the test scenes. We do not see a clear gap, showing that AMR generalizes well to unseen objects.

Qualitative results: Fig. 7(a) and (b) present two successful runs with different goal poses and robot sizes. AMR plans safe trajectories by considering the robot's size and can track and reach the target object precisely, even when the target is far or out of view. Fig. 7(c) shows typical failure cases for both the baseline and AMR. For the baseline, the failure is mainly caused by pose estimation in challenging scenarios, such as occlusion and object being too far. AMR sometimes fails to track the correct object when there are repetitive objects and may go to the wrong side when the viewpoint is ambiguous.

B. Ablation Study

To verify our design choices, we ablate our models by disabling some of the components and analyze their impacts on the precision (Fig. 8) and robustness (Table II). In particular, we find **Sensor Modality** has the biggest impact. Without LiDAR, the robot is more prone to colliding with surrounding objects. Likewise, the 3D information from depth helps the robot avoid obstacles and approach targets accurately. Without the **Footprint** token, the model suffers from a high collision rate because the robot cannot consider its size when planning. Our **Autoregressive Trajectory Decoding** scheme outperforms Action Chunking Transformer (ACT) since it better captures the multi-modal nature of navigation trajectories. While **Mask Tracking** only moderately improves the precision, it improves the model interpretability, as we find a strong correlation between incorrect mask tracking and robot navigating to the wrong object.

Large-scale evaluation matters: In Fig. 8, all ablated models have comparable median errors, indicating that they perform almost equally well for more than half of the test cases, but their long-tail failure cases vary significantly. This implies that our method is more *robust* across large datasets with environment variations.

TABLE II
COLLISION RATE OF ABLATED MODELS

Full	No mask dec.	ACT decoder	No DAgger	No depth	No footprint	No LiDAR
3.8	7.1	5.8	5.5	17.7	26.2	25.6

TABLE III
MEAN NAVIGATION ERRORS IN A REAL KITCHEN

	Fridge	Sink	Stove
Hist=0	1 / 2.4 cm / 2.3°	3 / 14.8 cm / 0.1°	3 / 2.0 cm / 0.8°
Hist=4	3 / 3.1 cm / 0.4°	3 / 7.9 cm / 0.2°	3 / 2.4 cm / 0.8°
	Cabinet	**Spam**	**Cup**
Hist=0	2 / 3.1 cm / 1.7°	1 / 1.8 cm / 0.6°	3 / 9.6 cm / 0.8°
Hist=4	3 / 2.1 cm / 0.6°	2 / 2.0 cm / 0.1°	3 / 8.7 cm / 0.9°

Each object we report (#completed / distance error / orientation error). 3 runs were performed per object. Cells are colored green according to the number of completed runs.

C. Closed-Loop Hardware Evaluation

We evaluate our system in a real kitchen on a mobile manipulator with an omnidirectional base and an RGB-D head camera with tilt support. We use one reference image covering the whole kitchen to specify 4 large objects (*fridge, sink, stove* and *cabinet*) and two zoomed-in images for small targets (*spam* and *cup*). The robot is initialized at three distinct starting locations. For each run, we continuously run AMR to navigate the robot to all 6 objects, with the relative goal condition describing the robot facing the object of interest, i.e., $\mathbf{C} = (d = 1\,\text{m}, \theta = 0°, S = \text{front})$. For each object, we perform 3 runs.

In Table III, we show the number of completed trials and compute the errors against hand-measured ground truths. *hist=4* succeeded in all tasks except for *spam* starting at location #3. Unlike in simulation, history is critical for the robot to track the target, as *hist=0* fails more often on large (*fridge*) and small (*spam*) objects. Among the objects, *sink* has the largest distance error due to unstable tracking and the lack of a large surface area, as the sink is an object that is concave to the tabletop surface and can easily be obscured. Except for *sink* and *cup*, all other objects achieve distance errors on the order of $1.8 \sim 3.1$ cm to the goal specification.

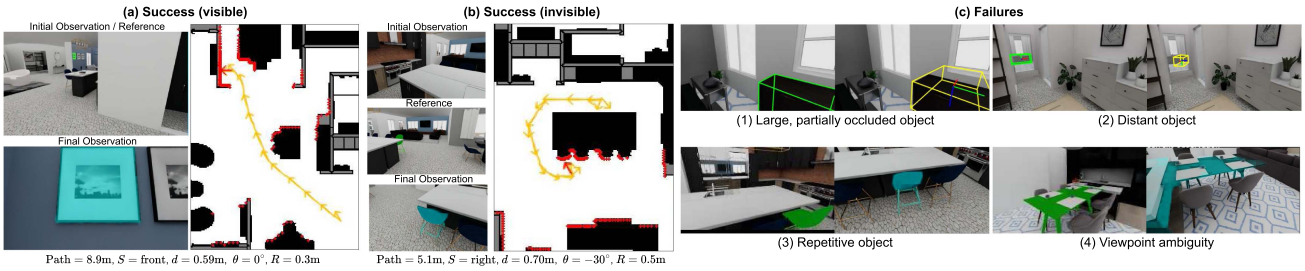

Fig. 7. Qualitative examples: (a) object initially visible (b) object initially out of view. The initial masks (green) are specified in the reference images. Cyan masks are predicted by the model. Trajectory accommodates the robot radius R for obstacle avoidance. (c) Typical failures: (1,2) Baseline: Object pose estimation may fail when the object is partially occluded or too far (green box: groundtruth, yellow box: prediction) (3) AMR may go to the wrong object when the object is repetitive. (4) AMR and Baseline: a robot may go to the wrong side with no dominantly visible side (i.e. looking at an object from 45° angle). This can be addressed by using a less ambiguous reference image.

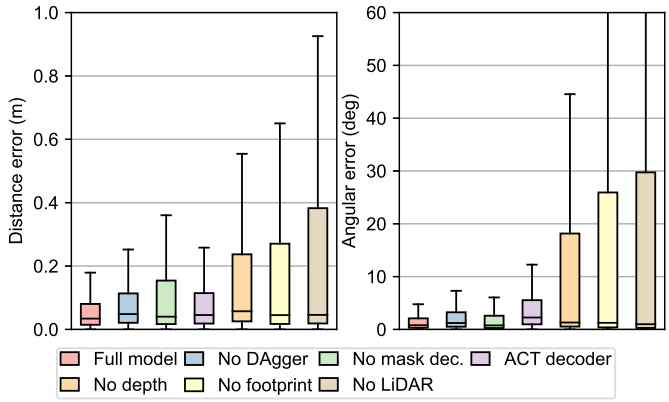

Fig. 8. Ablation study demonstrating effects of various choices on navigation errors in the test scenes. We consider both complete and incomplete runs.

D. Qualitative Experiments

Generalization in new real-world scenes: In Fig. 10, we show additional results of AMR navigating to diverse objects in another real-world environment. AMR can handle objects being out of view and large viewpoint changes (i.e., going to the back of a cabinet). We refer the reader to the website for the videos.

Closing the fridge drawer: In Fig. 11, we show that by accurately aiming the robot at a target object, a robot can perform downstream tasks using its body as the end effector. The goal is specified by $M = I_{\text{drawer}}$ and $\mathbf{C} = (d = 1\,\text{m}, \theta = 0^\circ, S = \text{front})$. Once the robot reaches the goal, the robot moves forward to close the drawer in an open loop.

Loading a pallet in a warehouse: We test AMR on a simulated forklift to load a randomly placed pallet in a warehouse. Existing work on controlling forklifts requires sophisticated modeling and motion planning [31], [32] with privileged state information. Here, we show that a learning system with onboard sensors can achieve the same precision. We empirically generate ~ 500 demonstrations with forklift kinematics to finetune AMR. In Fig. 11, we show AMR successfully navigates a forklift to face a pallet directly $\mathbf{C} = (d = 2\,\text{m}, \theta = 0^\circ, S = \text{front})$. The precision is sufficient such that the fork can be completely inserted into a pallet by driving the forklift forward in an open loop fashion. More information is provided on the website.

VI. Related Work

Object-goal and instance-goal navigation: Object-goal navigation typically refers to a robot navigating to any object of the specified category (e.g., couch) [15], [33], [34]. Instance-goal navigation enables a robot to reach a specific object by using a close-up view of that object [8], [35]. These systems typically consider the goal reached when the robot is within 1.0 m radius to the goal. While [36] improves the last-mile goal approaching through image matching, it cannot reach a precise pose relative to an object. Unlike existing approaches, AMR is designed to reach any object with centimeter-level precision. Such generality and precision are enabled by a novel goal parametrization (reference image with mask) and a model trained on diverse and precise demonstrations.

Object pose estimation: Object pose estimation [13], [37] provides an alternative solution to high-precision object-centric navigation. State-of-the-art object pose estimation typically requires prior knowledge of the object (CAD model, template images, or object category), and the object must be visible in the camera. These assumptions are difficult to satisfy since a robot often needs to go to arbitrary objects, and the object can be temporarily out of view. In contrast, AMR is trained on diverse and photorealistic simulation data to learn general knowledge of object instances and geometries. Furthermore, unlike object pose estimation, which is "passive" and struggles when the object is too small or too far, AMR is an active approach akin to Image-based Visual Servoing (IBVS) [38] that refines its predictions as the robot gets closer to the goal. AMR is more sophisticated than IBVS since AMR reasons about objects, adheres to robot kinematics, and handles collision avoidance.

Mobile manipulation systems: There is a surge of interest in developing learning-based mobile manipulation systems [2], [39], [40], [41], [42], [43]. Some works assume additional information such as object pose or robot pose to be available ([39], [40], [42], [43]). Other systems show that it is possible to directly map sensor observations to actions [2], [41], but they only consider a limited set of scenarios (e.g. no need to avoid unseen obstacles or handle unseen objects), so that high-precision base positioning is not a major concern. As of today, the most general and capable mobile manipulation systems are still modular [16], [17], and they require mapping the environment and objects beforehand. In particular, [16] shows that accurate base positioning is crucial for successful manipulation. Our work can be integrated into both learning and modular mobile

Fig. 9. Real kitchen experiments. Left: reference images and contours of target object masks. While the *fridge* is partially out of view, the model can still reach the fridge. Middle: Initial view of the scene through the tilt camera. Right: robot observations after reaching each object overlaid with predicted masks (in cyan).

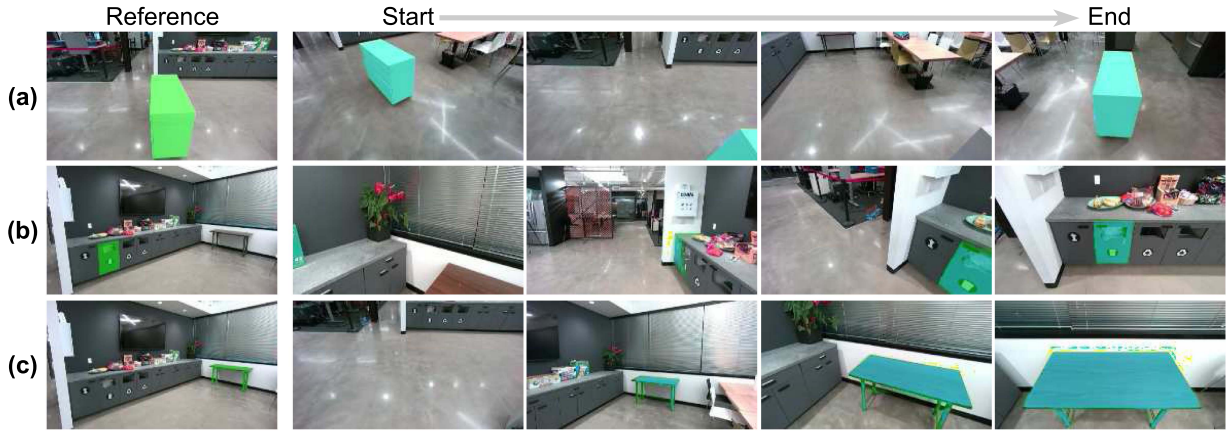

Fig. 10. Real-world qualitative experiments in new scenes. Left: reference image with target object highlighted by the green mask. Remaining columns: robot's camera view while moving. The tracked object is highlighted by the cyan mask. The tasks are: (a) Go to the back of the cabinet and stand 0.8 m away. (b) Go to the front of the landfill trashcan and stand 1 m away. (c) Go to the front of the table and stand 0.5 m away.

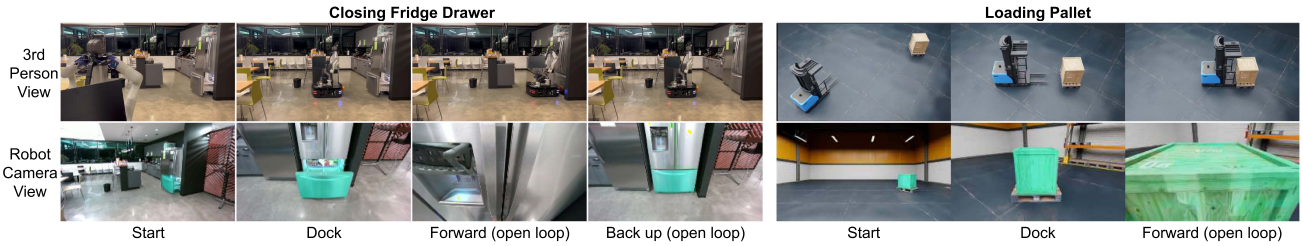

Fig. 11. Using AMR for mobile manipulation tasks. Left: closing a fridge drawer by pushing the robot body towards the drawer. Right: forklift loading a pallet. AMR accurately tracks the target objects throughout.

manipulation systems so that a robot can reach an accurate base pose for manipulation without a prior map or object model.

VII. Conclusion and Limitations

We present AMR, a vision-based navigation model that navigates to any object with centimeter precision. While trained entirely in simulation, it transfers to real-world and unseen objects with little degradation in precision. AMR does not require object 3D models or a map to operate, runs in real-time at 10 Hz, and easily adapts to other robots with fine-tuning.

Limitations: AMR is designed for local navigation as it has a short-term memory. Since it is only trained in household environments, its generalization in other scenarios, such as factories and

unstructured environments, is potentially limited. Additionally, real robots have complex geometric shapes and sensor placements, and our assumption of a robot being cylindrical with a centered camera can hinder its applicability to diverse mobile robots such as legged robots. However, we anticipate that these limitations can be addressed by increasing the robot's memory window, improving the diversity of the training environments, and more comprehensive modeling of robot shape and sensor placements.

REFERENCES

[1] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 672–678.

[2] K. Ehsani et al., "Imitating shortest paths in simulation enables effective navigation and manipulation in the real world," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 16238–16250.

[3] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal masked diffusion policies for navigation and exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 63–70.

[4] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez, "SayNav: Grounding large language models for dynamic planning to navigation in new environments," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2024, pp. 464–474.

[5] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A general navigation model to drive any robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7226–7233.

[6] R.-Z. Qiu et al., "Learning generalizable feature fields for mobile manipulation," 2024, *arXiv:2403.07563*.

[7] S. Raychaudhuri, T. Campari, U. Jain, M. Savva, and A. X. Chang, "MOPA: Modular object navigation with pointgoal agents," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2024, pp. 5763–5773.

[8] M. Chang et al., "GOAT: Go to any thing," 2023, *arXiv:2311.06430*.

[9] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine, "Navigation with large language models: Semantic guesswork as a heuristic for planning," in *Proc. 7th Conf. Robot Learn.*, 2023, pp. 2683–2699.

[10] Z. Zhang, A. Lin, C. W. Wong, X. Chu, Q. Dou, and K. Au, "Interactive navigation in environments with traversable obstacles using large language and vision-language models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 7867–7873.

[11] N. Hirose, C. Glossop, A. Sridhar, D. Shah, O. Mees, and S. Levine, "LeLaN: Learning a language-conditioned navigation policy from in-the-wild videos," in *Proc. 7th Conf. Robot Learn.*, 2024, pp. 666–688.

[12] K. Yadav et al., "Habitat challenge 2023," 2023. [Online]. Available: https://aihabitat.org/challenge/2023/

[13] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6D pose estimation and tracking of novel objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 17868–17879.

[14] D. Shah et al., "ViNT: A foundation model for visual navigation," in *Proc. 7th Conf. Robot Learn.*, 2023, pp. 711–733.

[15] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "ZSON: Zero-shot object-goal navigation using multimodal goal embeddings," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 32340–32352.

[16] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "OK-Robot: What really matters in integrating open-knowledge models for robotics," 2024, *arXiv:2401.12202*.

[17] M. Bajracharya et al., "Demonstrating mobile manipulation in the wild: A metrics-driven approach," 2024, *arXiv:2401.01474*.

[18] A. Kirillov et al., "Segment anything," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 3992–4003.

[19] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2020, pp. 440–448.

[20] J. Liang et al., "Code as policies: Language model programs for embodied control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9493–9500.

[21] M. Khanna et al., "Habitat synthetic scenes dataset (HSSD-200): An analysis of 3D scene scale and realism tradeoffs for objectgoal navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2024, pp. 16384–16393.

[22] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3191–3198.

[23] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, [Online]. Available: https://ompl.kavrakilab.org

[24] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Proc. Robot., Sci. Syst.*, 2023.

[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16000–16009.

[26] NVIDIA, "NVIDIA Isaac Sim," 2021. [Online]. Available: https://developer.nvidia.com/isaac-sim

[27] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 22955–22968.

[28] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[29] R. C. Conlter, "Implementation of the pure pursuit path tracking algorithm," Camegie Mellon Univ, Pittsburgh, PA, USA, 1992.

[30] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," in *Proc. Innov. Appl. Artif. Intell. Conf.*, 1999, vol. 1999, no. 343–349, pp. 2–2.

[31] T. A. Tamba, B. Hong, and K.-S. Hong, "A path following control of an unmanned autonomous forklift," *Int. J. Control, Automat. Syst.*, vol. 7, no. 1, pp. 113–122, 2009.

[32] Y. Sun, J. Yang, Z. Zhang, and Y. Shu, "An optimization-based high-precision flexible online trajectory planner for forklifts," *Actuators*, vol. 12, 2023, Art. no. 162.

[33] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Sci. Robot.*, vol. 8, no. 79, 2023, Art. no. eadf6991.

[34] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4247–4258.

[35] J. Krantz et al., "Navigating to objects specified by images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 10882–10891.

[36] J. Wasserman, K. Yadav, G. Chowdhary, A. Gupta, and U. Jain, "Last-mile embodied visual navigation," in *Proc. 7th Conf. Robot Learn.*, 2023, pp. 666–678.

[37] J. Lin, L. Liu, D. Lu, and K. Jia, "SAM-6D: Segment anything model meets zero-shot 6D object pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 27906–27916.

[38] Y. Wang et al., "NeRF-IBVS: Visual servo based on NeRF for visual localization and navigation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 8292–8304.

[39] S. Uppal, A. Agarwal, H. Xiong, K. Shaw, and D. Pathak, "SPIN: Simultaneous perception, interaction and navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 18133–18142.

[40] N. Yokoyama et al., "ASC: Adaptive skill coordination for robotic mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 9, no. 1, pp. 779–786, Jan. 2024.

[41] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *Proc. Conf. Robot Learn.*, 2024.

[42] X. Huang, D. Batra, A. Rai, and A. Szot, "Skill transformer: A monolithic policy for mobile manipulation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2023, pp. 10852–10862.

[43] J. Hu, P. Stone, and R. Martín-Martín, "Causal policy gradient for whole-body mobile manipulation," in *Proc. Robot.: Sci. Syst.*, 2023.