Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Learning to Localize Reference Trajectories in Image-Space for Visual Navigation

Finn Lukas Busch, Matti Vahs, Quantao Yang, Jesús Gerardo Ortega Peimbert,
Yixi Cai, Jana Tumova, Olov Andersson - Paper ID 732

***Abstract*—We present LoTIS, a model for visual navigation that provides robot-agnostic image-space guidance by localizing a reference RGB trajectory in the robot's current view, without requiring camera calibration, poses, or robot-specific training. Instead of predicting actions tied to specific robots, we predict the image-space coordinates of the reference trajectory as they would appear in the robot's current view. This creates robot-agnostic visual guidance that easily integrates with local planning. Consequently, our model's predictions provide guidance zero-shot across diverse embodiments. By decoupling perception from action and learning to localize trajectory points rather than imitate behavioral priors, we enable a cross-trajectory training strategy that learns robust invariance to viewpoint and camera changes. We outperform state-of-the-art methods by 20-50 percentage points in success rate on forward navigation, and paired with a local planner we achieve 94-98% success rate across diverse sim and real environments. Furthermore, we achieve over 5× improvements on challenging tasks where baselines fail, such as backward traversal. The system is lightweight ($\sim$ 22 Hz on Jetson Orin AGX) and straightforward to use: we show how even a video from a handheld phone camera directly enables different robots to navigate to any point on the trajectory. Videos, demo, and code are available at https://finnbusch.com/lotis.**

I. Introduction

Visual navigation enables robots to navigate environments using only camera observations. Early work focused on navigating to a single goal image. To enable long range navigation, subsequent work has considered visual reference trajectories, i.e. sequences of unposed RGB images recorded along a path that capture the full route to be followed. Such trajectories can be recorded with any camera, from a handheld smartphone to a robot-mounted sensor, allowing routes to be defined through demonstration without requiring metric maps.

State-of-the-art approaches typically address this task via end-to-end learning, training policies that map current observations and a goal image directly to robot actions [26, 29, 31]. To follow trajectories, these methods extract a single subgoal image from the trajectory and use this as the goal image for the learned policy. We identify three limitations. First, outputting actions directly constrains models to the training action space (typically planar differential drive), limiting generalization to platforms like aerial robots. Second, to learn a policy mapping from observations to actions, these methods require training examples where the robot traverses from the current view to the goal image. This restricts training to start and goal images from the same trajectory, and thus implicitly assumes that the deployment camera shares the characteristics of the recording device. We show that deviations in camera intrinsics or robot embodiment therefore lead to degraded performance. Third, relying on a single subgoal image makes the system sensitive to subgoal selection errors. If an incorrect target is chosen, the navigation policy may fail, as the robot cannot visually match its current view to the incorrect subgoal.

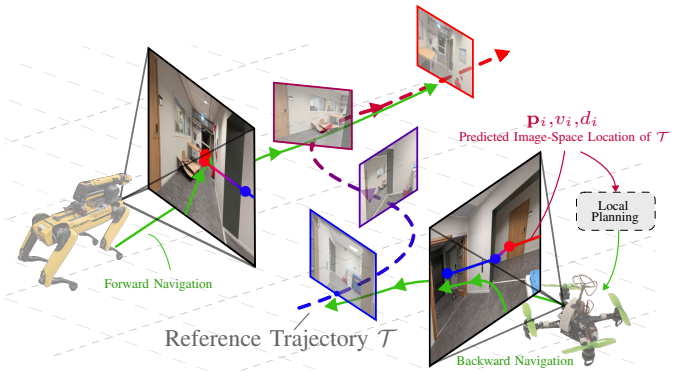

Fig. 1. Given only a reference trajectory of (unposed) RGB images $\mathcal{T}$, our model localizes the trajectory within the robot's current view. The predicted image-space coordinates, distances and visibility of the reference trajectory poses $(\mathbf{p}_i, d_i, v_i)$ act as an interface for local planning, allowing any robot such as quadrupeds or drones to go to any point on the trajectory, from any view of the trajectory.

We approach the problem of visual navigation by decoupling perception from action, and propose to learn a model that predicts the reference trajectory directly in image space, such that it can be tracked by conventional local planners.

To this end, we present LoTIS, a model for visual navigation that given a reference trajectory (sequence of RGB images) predicts the following representation: 1) the image-space coordinates of where each trajectory pose would appear in the robot's current view, 2) whether these poses are visible, and 3) a normalized distance to each visible pose, see Fig. 1. This overcomes the aforementioned challenges: First, our model predictions can be easily paired with downstream controllers or planners for diverse robot embodiments, without further training. Second, because our model learns to localize trajectory poses rather than imitate actions, we can construct training

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The development was partly enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

The authors are with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Sweden, and also affiliated with Digital Futures. Contact: {flbusch, vahs, quantao, jgop, yixica, tumova, olovand}@kth.se

pairs by sampling reference and query images from different trajectories. As a result, we can explicitly train on data from mismatched cameras, varied mounting heights, and off-trajectory viewpoints, enabling robustness to such variations. Finally, processing the full trajectory sequence rather than selecting a single subgoal image overcomes the sensitivity of accurate subgoal selection in existing works, leading to substantial performance gains in navigation success rate across all evaluations.

In addition, our model enables new capabilities, most notably backward traversal, where related approaches largely fail while our method maintains robust performance. This enables, for the first time, using an RGB reference trajectory in the general setting where the robot can navigate from anywhere in the vicinity (with visual overlap) to any point on the trajectory, both forward and backward.

In summary, we decouple perception from action and provide a learned perception model that interfaces with classical planners, with the following contributions:

1) We propose an image-space representation, defined by 2D coordinates, visibility, and distance of each reference trajectory pose in the robot's current view, that easily pairs with local planners to guide diverse robots in image space. In real-world experiments, we show that this representation is suitable to guide both a quadrotor and a quadruped from a single phone-recorded trajectory.
2) We introduce a cross-trajectory training strategy tailored for this representation, where reference and query images are sampled from different trajectories, exposing the model to camera mismatches and challenging viewpoints. This enables robust backward traversal and consequently supports navigation to any point on the trajectory.
3) We propose a model architecture that processes the full reference trajectory jointly rather than relying on single subgoal-selection, while enabling real-time deployment on embedded hardware. We show that for existing methods, reliance on a single subgoal-selection leads to decreased localization accuracy with distance from the trajectory, whereas our joint processing enables LoTIS paired with a local planner to maintain robust success rates even when initialized far from the trajectory.

II. Related Work

A. Visual Navigation

The core challenge of visual navigation is translating image observations into physical motion, a task traditionally addressed by visual servoing. Visual servoing stands as the classical foundation for visual navigation, providing a framework for translating sensor visual feedback into control actions, with Image-Based Visual Servoing (IBVS) minimizing error directly in feature space and Position-Based Visual Servoing (PBVS) operating through explicit pose estimation [8, 5, 6]. However, these methods suffer from well-documented limitations: local minima, sensitivity to calibration errors, and reliance on continuous feature tracking that breaks under occlusions or appearance changes [4]. These challenges motivated learning-based approaches that can extract features robust to appearance variation and operate without explicit calibration.

Modern learning-based visual navigation has evolved from image-goal navigation benchmarks [13, 14] toward topological methods designed for real-world deployment. GNM [25] introduced a cross-embodiment navigation policy trained on heterogeneous robot data, predicting both temporal distance for subgoal selection and normalized waypoint actions. ViNT [26] scaled this approach using a Transformer architecture trained on over one hundred hours of diverse navigation data. NoMaD [29] unified goal-directed navigation and exploration within a diffusion policy, using goal masking to enable flexible inference.

Since performance largely depends on accurate subgoal selection, PlaceNav [31] aims to improve robustness by reframing subgoal selection as visual place recognition using CosPlace [1], applying Bayesian filtering for temporal consistency while relying on GNM for low-level waypoint control. FAINT [32] combines EigenPlaces [2] for place recognition with a navigation policy trained on frozen Theia [27] representations, demonstrating that simulation-trained policies can outperform real-world-trained counterparts when sufficient synthetic data is available.

These methods share fundamental limitations that our work addresses. First, they couple perception with learned behavioral priors, outputting actions tied to specific robot kinematics. While these approaches train on data from multiple robots, they remain constrained to platforms with similar action spaces (e.g., ground-based differential drive), limiting generalization to platforms with different motion capabilities such as aerial robots. Second, they are primarily designed for forward trajectory following and struggle with backward traversal, where the robot encounters viewpoints substantially different from those in the recorded trajectory. Third, they are sensitive to camera mismatch between recording and deployment, as training on on-trajectory data does not expose the models to such variations.

Concurrent image-space local navigation methods such as VAMOS [3] and VENTURA [42] also aim to improve cross-embodiment generalization by predicting local traversability or local paths from the current image. However, these methods address a complementary problem: local navigation toward a nearby language or image goal from the current observation. In contrast, LoTIS addresses reference-trajectory navigation, where the robot must localize itself relative to a pre-recorded image sequence and navigate to arbitrary trajectory indices, including distant goals, multi-turn routes, and backward traversal. Thus, local image-space planners could in principle be combined with LoTIS as downstream local controllers, while LoTIS provides the long-range trajectory localization signal.

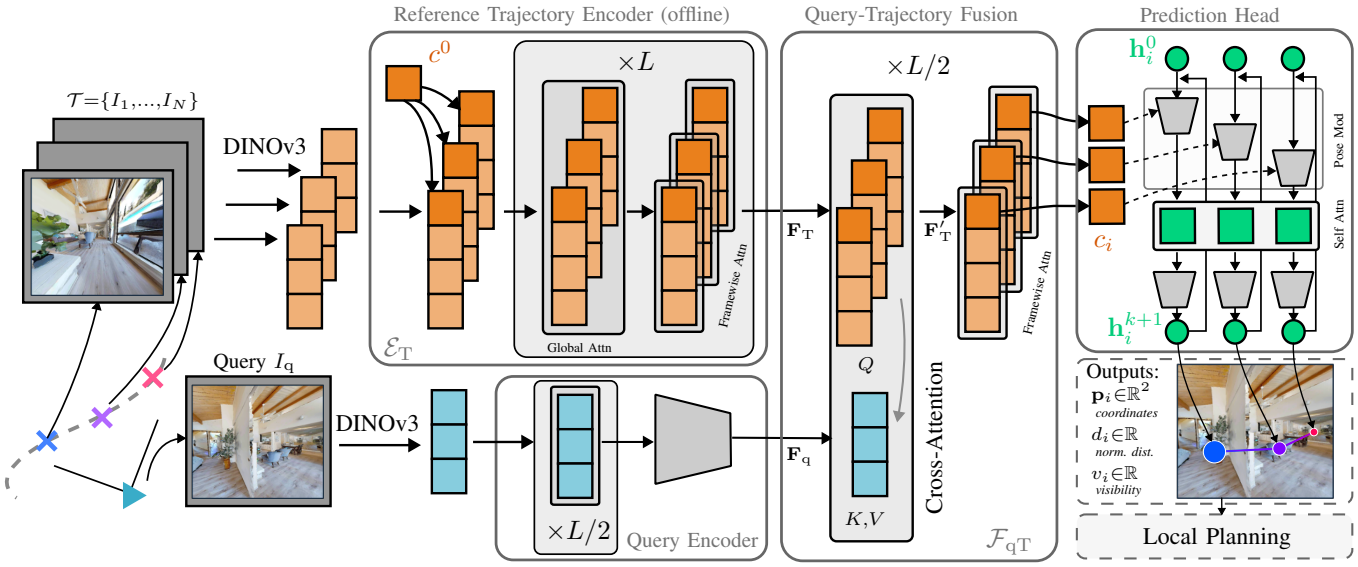

Fig. 2. **LoTIS Architecture.** Reference trajectory $\mathcal{T}$ and query I_q are processed by frozen DINOv3 backbones. A trajectory encoder ($\mathcal{E}_T$) captures spatio-temporal context once (offline), while a query encoder ($\mathcal{E}_q$) and query-trajectory fusion ($\mathcal{F}_{qT}$) perform online feature extraction and fusion, respectively. Finally, a recurrent transformer iteratively regresses image-space coordinates ($\mathbf{p}_i$), visibility (v_i), and distances (d_i).

B. Learned Visual Geometry

The field of learned visual geometry is related to visual navigation through a shared need for spatial understanding from images. In this domain, recent models achieve remarkable geometric understanding from images. DUSt3R [36] recovers dense pointmaps from image pairs without calibration, MASt3R [15] augments this with dense local features and a matching loss for robust correspondences under extreme viewpoint changes, VGGT [35] extends to joint pose and geometry estimation, and Depth Anything 3 [16] unifies multi-view depth and pose estimation through a depth-ray representation with a plain DINOv2 backbone.

These methods are primarily designed for 3D reconstruction and mapping rather than online robot control, and have not yet been adapted into real-time navigation policies. We found that applying these state-of-the-art reconstruction models to our evaluation trajectories (see Fig. 4) often yielded inconsistent poses or failed reconstructions. We suspect that solving the full 3D reconstruction problem, which prioritizes global metric accuracy, is computationally heavier and likely harder than the task of relative trajectory localization required for navigation. We provide examples covering all of our real-world evaluation trajectories in the appendix in the supplementary material, S7. LoTIS addresses this by learning a representation explicitly tailored for navigation. Instead of recovering general scene geometry, our model is designed to provide the specific information required for navigation: the location of the reference trajectory relative to the robot's current view.

III. Problem Statement

We consider the task of navigating to any point along a recorded RGB reference trajectory using only RGB images. Formally, given a reference trajectory $\mathcal{T} = \{I_1, \ldots, I_N\}$ consisting of N RGB images, and a query image I_q from the robot's current viewpoint, the goal is to navigate to any point on the trajectory, indexed by $g \in \{1, \ldots, N\}$, starting from any position in the trajectory's vicinity, as long as there is visual overlap with some portion of $\mathcal{T}$.

We make no explicit assumptions about camera calibration or robot embodiment. As such the reference trajectory $\mathcal{T}$ may be recorded with a different camera or platform than the one used for navigation.

IV. Method

We approach this problem by decoupling perception from action and learn a model that provides robot-agnostic guidance that can be easily consumed by classical motion planning and control stacks designed for specific embodiments. To this end, we propose LoTIS, a model that predicts where the poses of a reference trajectory appear in the robot's current view in image-space. As this output is formulated in the robot's current view, it can easily be used by downstream controllers, e.g. by simply steering towards the predicted points.

A. Guidance Representation

Our model's output provides guidance to be used by classical motion planning stacks. Formally, in the image-space of the robot's current view, we predict a triplet $(\mathbf{p}_i, v_i, d_i)$ for each reference frame $I_i \in \mathcal{T}$:

- A 2D point in image-space $\mathbf{p}_i \in \mathbb{R}^2$: where the camera pose corresponding to trajectory image I_i would appear in the query view, normalized to $[-1, 1] \times [-1, 1]$
- A visibility logit $v_i \in \mathbb{R}$: whether the pose is visible or occluded/out of view
- A normalized distance $d_i \in [0, 1]$: relative distance to the pose from the current query viewpoint. For scale-independency, we normalize the distances such that the farthest visible point is at distance 1.

B. Model Architecture

We design our model to process the reference trajectory $\mathcal{T}$ and match this with the current view of the robot I_{q}. Unlike most existing methods [26, 29, 32, 31] that match the current view pairwise to each image on the trajectory, we propose processing the full trajectory jointly and matching the robot's view against that. To ensure real-time deployment, we propose an asymmetric model architecture: a large *trajectory encoder* $\mathcal{E}_{\mathrm{T}}$ runs once at deployment to process the full trajectory, while a lightweight *query encoder* $\mathcal{E}_{\mathrm{q}}$ and *decoder* run efficiently online to produce the final predictions. The decoder consists of query-trajectory fusion $\mathcal{F}_{\mathrm{qT}}$, which matches I_{q} against the processed trajectory to find visual correspondences, and aggregates those into a global context within one summary token $\mathbf{c}_i$ per frame i on the reference trajectory; and a prediction head predicts the one final prediction $\mathbf{p}_i, v_i, d_i$ per frame i from the summary tokens. An overview is provided in Fig. 2

We use DINOv3 [28] as a frozen backbone to extract initial features for all images, given the strong performance of the DINO family on geometric vision tasks [35]. All images are resized to 224×224 before DINOv3.

1) Trajectory Encoder $\mathcal{E}_{\mathrm{T}}$: Given the reference trajectory $\mathcal{T} = \{I_1, \ldots, I_N\}$, we extract per-frame features using frozen DINOv3, project to dimension D, and prepend a learnable *summary token* $\mathbf{c}_0$ per frame. The resulting tokens are processed by L transformer blocks, each alternating between global attention (across all frames and patches) and frame-wise attention (within each frame), following [35]. We incorporate temporal structure via rotary position encodings (RoPE [30]). The encoder outputs $\mathbf{F}_{\mathrm{T}} \in \mathbb{R}^{N\times(P+1)\times D}$, where P is the number of patches per image.

2) Query Encoder $\mathcal{E}_{\mathrm{q}}$: The query image I_q is processed by the same frozen DINOv3 backbone, projected to dimension D, and refined through $L/2$ self-attention layers with spatial RoPE, producing tokens $\mathbf{F}_{\mathrm{q}} \in \mathbb{R}^{P\times D}$. A linear adapter aligns features with the trajectory encoder's representation space.

3) Query-Trajectory Fusion $\mathcal{F}_{\mathrm{qT}}$: This module fuses query and trajectory features $\mathbf{F}_{\mathrm{q}}, \mathbf{F}_{\mathrm{T}}$ through $L/2$ blocks, each alternating cross-attention and frame-wise self-attention. For cross-attention, trajectory features $\mathbf{F}_{\mathrm{T}}$ (patches and summary tokens) serve as queries, while query image tokens $\mathbf{F}_{\mathrm{q}}$ serve as keys and values:

$$\mathbf{F}'_{\mathrm{T}} = \mathrm{CrossAttn}(Q{=}\mathbf{F}_{\mathrm{T}}, K{=}\mathbf{F}_{\mathrm{q}}, V{=}\mathbf{F}_{\mathrm{q}}). \tag{1}$$

This allows each trajectory frame to attend to the query view and find visual correspondences. The subsequent frame-wise self-attention aggregates these local correspondences into global context within each frame's summary token $\mathbf{c}_i$.

4) Prediction Head $\mathcal{P}$: We employ a recurrent transformer regressor, adapting the iterative refinement architecture from [35]. The head maintains a latent estimate $\mathbf{h}_i \in \mathbb{R}^D$ for each trajectory frame $i \in \{1, \ldots, N\}$, initialized by a learnable query. Over K iterations, the current estimate $\mathbf{h}^{(k)}$ is projected and used to condition the encoded summary tokens $\mathbf{c}_i$ via Adaptive Layer Normalization (AdaLN [21]). The modulated tokens are processed by self-attention across all N frames, enabling the model to enforce geometric consistency along the trajectory. A linear layer predicts residual updates, and the estimate is refined as $\mathbf{h}^{(k+1)} \leftarrow \mathbf{h}^{(k)} + \Delta\mathbf{h}$.

After K iterations, the final estimates are projected to the output space: image coordinates $\mathbf{p}_i$ via $\tanh$ (normalized to $[-1, 1]^2$), visibility logits v_i, and normalized distances d_i via scaled $\tanh$ (bounded to $[0, 1]$).

5) Implementation: We use $L{=}12$ layers, hidden dimension $D{=}256$, and $K{=}4$ refinement iterations. This yields $\sim$50 M parameters, with $\sim$32 M in the trajectory encoder. Further details on the model and scaling are provided in the supplementary material, S6.

C. Training

1) Cross-Trajectory Training: We train on a combination of real-world navigation datasets and synthetic data generated in simulation [24].

A key advantage of our representation is that it enables *cross-trajectory* sampling: For real-world datasets, we sample the reference trajectory $\mathcal{T}$ and query image I_{q} from *different* trajectories within the same environment. We do this to encourage generalization, as this exposes the model to camera mismatches, off-trajectory viewpoints and environmental variations between traversals (lighting changes, dynamic objects, seasonal differences). Importantly, cross-trajectory sampling is only possible because we do not require action annotations between reference trajectory and query image.

In simulation, we generate reference trajectories between random start and goal points, sampling query images from random poses in the vicinity of these trajectories. To encourage generalization, we independently randomize camera parameters (field-of-view, aspect ratio, mounting height) for reference and query views, and apply rotational perturbations to query and trajectory poses before recording the images. Trajectory frames are sampled at stochastic intervals to vary sequence density for both simulation and real-world datasets.

Ground-truth image coordinates and distances are obtained by projecting trajectory camera centers into the query camera using camera poses. Visibility labels are generated by checking whether the projected point lies inside the image and is not occluded; depth maps are used only for this occlusion check. For real-world data, we preprocess depth with PriorDepthAnything [37].

2) Losses: We optimize the model using a weighted sum of three objectives:

$$\mathcal{L} = \lambda_{\mathrm{pos}}\mathcal{L}_{\mathrm{pos}} + \lambda_{\mathrm{vis}}\mathcal{L}_{\mathrm{vis}} + \lambda_{\mathrm{dist}}\mathcal{L}_{\mathrm{dist}}, \tag{2}$$

where $\mathcal{L}_{\mathrm{pos}}$ is an L_1 loss applied to the predicted coordinates $\mathbf{p}_i$, masked to only penalize points where the ground truth is visible ($v_i^* = 1$). $\mathcal{L}_{\mathrm{vis}}$ is a Binary Cross-Entropy loss applied to the visibility logits. $\mathcal{L}_{\mathrm{dist}}$ is an L_1 loss applied to the normalized distance predictions. We compute this loss on the output of each iteration of the prediction head and apply temporal weighing [33].

3) Datasets: Our training data spans diverse environments, including simulation (HM3D [23], HSSD [11], AI2-THOR [12]) and real-world trajectories (CODa [41], LILocBench [34], BotanicGarden [17], TartanGround [20]). This mixture exposes the model to cluttered indoor spaces, unstructured outdoor paths, and urban settings with dynamic objects, as well as seasonal and day-night variations.

In total, we use approximately 25,000 reference trajectories (up to 40 frames each) and 850,000 query images across 500 unique environments. We train on a single NVIDIA RTX 5090 for approximately 4 days.

D. Navigation

Our model outputs a set of points $\mathbf{p}_i$ in the robot's current image frame representing the reference trajectory. To navigate, a downstream controller uses these predictions alongside a user-specified goal index $g \in \{1, \dots, N\}$ to determine the appropriate actions. We demonstrate this flexibility on two distinct controllers:

1) Yaw Controller with constant forward velocity: To demonstrate robust performance with minimal complexity, we employ a simple controller that controls only yaw velocity while commanding constant forward velocity. The controller identifies visible points $\mathcal{I}_{\text{vis}} = \{i \mid \sigma(v_i) > 0.5\}$, selects the closest one ($k = \text{argmin}_{i \in \mathcal{I}_{\text{vis}}} d_i$), and applies an offset in the desired direction of travel. We apply a proportional controller to derive yaw velocity commands that steer toward this target while commanding constant forward velocity.

2) Model Predictive Path Integral Control: To demonstrate that our model predictions can be easily combined with more sophisticated control strategies, we employ a perception-aware Model Predictive Path Integral (MPPI) controller [38] for a drone. We formulate a cost function that ensures that our predictions remain in the robot's FOV, aligning with similar approaches in perception-aware MPC [9, 18], and further incorporate proactive collision-avoidance. To this end, we use depth maps predicted by UniDepthV2 [22] to ground the model's predictions in 3D and to formulate collision avoidance. We refer the reader to the supplementary material for details on both controllers.

V. Simulation Experiments

We evaluate our method in photo-realistic simulation to assess robustness to environmental variations and to enable reproducible experiments, addressing the following questions:

- **Q1:** How well does our method perform on forward trajectory following compared to baselines that were specifically designed for this task?
- **Q2:** How does our model handle off-trajectory starts compared to methods that rely on discrete subgoal retrieval?
- **Q3**: How robust is our method to mismatched camera intrinsics and mounting heights between reference and query trajectories?
- **Q4:** To what extent does a model trained only on forward trajectories generalize to backward traversal without explicit backward traversal demonstrations?

A. Experimental Setup

We utilize the Gibson Habitat split [40] (5 environments) and HM3D [23] (100 environments) datasets within the Habitat simulator. All evaluation environments are held out from training. Hyperparameters were tuned during initial development on training environments and then fixed across all reported evaluations, baseline parameters were tuned for best performance. Overall, we evaluate on 100 reference trajectories for each dataset using two randomized initial poses for the robot per trajectory for each evaluation setup. To test robustness against camera mismatch, we define two evaluation setups: *1) Matched Camera:* The query agent is equipped with the exact camera with same mounting height as the reference trajectory, and *2) Cross-Camera:* The query agent has mismatched FOV (avg. $20°$, max $60°$ difference), aspect ratio (avg. 0.5, max 1.5), and mounting height (avg. $0.5\,\text{m}$, max $1.2\,\text{m}$) compared to the recorder. We provide details about the evaluation dataset statistics in the supplementary material. Furthermore, we apply rotational noise to the reference trajectory poses before capturing the frames to simulate imperfect recording. We investigate the following three navigation tasks:

1) **To End (Forward Navigation):** The goal is the final frame I_N of the reference trajectory. This is the standard task for the baselines we compare against.
2) **To Start (Backward Navigation):** The goal is the first frame I_1 of the reference trajectory. The agent must retrace the trajectory in reverse order, i.e. while opposing the views the reference trajectory was recorded from.
3) **Any Point (Random Goal):** The goal is a randomly selected frame I_k along the trajectory, requiring the agent to navigate to arbitrary targets within the trajectory.

For the **To End** and **To Start** tasks, we evaluate two initialization conditions: (1) **On-Trajectory:** The agent starts at a pose exactly belonging to the reference trajectory. (2) **Off-Trajectory:** The agent starts at a random pose in the vicinity of the trajectory with visual overlap. The **Any Point** task is evaluated only in the off-trajectory setting, as it is intended to assess the most versatile setting, where the agent starts from arbitrary starting positions and is tasked to navigate to any point on the reference trajectory.

Metrics. We report Success Rate (SR) and Success weighted by Path Length (SPL). A run is considered successful if the agent arrives within $0.5\,\text{m}$ of the goal. We terminate a run after 1000 simulation steps.

Baselines. We compare against four state-of-the-art navigation methods using official pre-trained weights. All baselines operate on a topological graph constructed from the reference trajectory: **ViNT** [26] and **NoMaD** [29] select subgoals via learned temporal distance, with NoMaD employing a diffusion policy for multimodal action distributions. **PlaceNav** [31] utilizes visual place recognition (CosPlace [1]) for retrieval and GNM [25] for waypoint control. Finally, **FAINT** [32] uses EigenPlaces [2] for retrieval and learns a navigation policy over frozen Theia [27] representations to enhance sim-to-real transfer.

TABLE I

NAVIGATION PERFORMANCE: WE REPORT SUCCESS RATE (SR) AND SPL (IN PARENTHESES). **CROSS**: QUERY CAMERA DIFFERS FROM REFERENCE CAMERA. **MATCHED**: SAME CAMERA PARAMETERS. **OFF-TRAJECTORY**: ROBOT INITIALIZED AWAY FROM THE TRAJECTORY. **ON-TRAJECTORY**: ROBOT INITIALIZED ON THE TRAJECTORY. FOR DETAILS, SEE SEC. V-A.

		To End (Forward)				**To Start (Backward)**				**Any Point**	
		On-Trajectory		Off-Trajectory		On-Trajectory		Off-Trajectory		Off-Trajectory	
	Method	Matched	Cross	Matched	Cross	Matched	Cross	Matched	Cross	Matched	Cross
Gibson	ViNT [26]	70.4 (67.8)	21.1 (17.8)	40.1 (30.7)	21.1 (17.5)	1.3 (0.9)	3.9 (3.4)	9.2 (8.1)	9.9 (7.8)	27.6 (22.8)	21.7 (19.3)
	PlaceNav [31]	53.9 (52.0)	5.3 (4.9)	15.1 (12.8)	11.2 (10.5)	3.9 (3.7)	4.6 (3.9)	9.2 (9.0)	10.5 (9.9)	22.4 (19.5)	13.8 (12.2)
	NoMaD [29]	23.0 (20.2)	8.6 (7.4)	24.3 (17.0)	17.1 (11.8)	8.6 (7.0)	7.2 (5.8)	11.2 (8.7)	11.2 (9.0)	31.6 (22.4)	27.0 (19.8)
	FAINT [32]	50.7 (47.8)	34.2 (31.0)	50.0 (42.5)	41.4 (33.6)	11.8 (9.3)	9.2 (7.5)	11.2 (10.1)	13.2 (10.4)	52.0 (42.6)	40.8 (35.2)
	LoTIS (Ours)	**94.7** (94.7)	**83.6** (83.1)	**88.2** (85.0)	**82.2** (79.5)	**88.2** (84.2)	**80.9** (78.1)	**77.6** (72.4)	**71.7** (67.7)	**85.5** (82.7)	**81.6** (77.8)
	+ Obstcl. Avoidance	**100.0** (99.9)	**98.0** (96.1)	**98.7** (93.9)	**94.7** (87.8)	**97.4** (89.9)	**89.5** (83.6)	**96.7** (87.6)	**90.8** (81.9)	**98.0** (92.4)	**95.4** (87.9)
HM3D	ViNT [26]	65.7 (64.4)	12.7 (12.1)	24.0 (20.8)	12.3 (10.3)	3.4 (3.1)	2.5 (2.3)	4.4 (3.7)	5.4 (5.0)	20.1 (17.9)	10.3 (9.3)
	PlaceNav [31]	55.4 (54.5)	7.8 (7.3)	11.3 (10.0)	6.4 (5.0)	3.4 (3.2)	2.5 (2.1)	4.9 (3.4)	4.9 (4.0)	13.2 (11.4)	8.3 (7.5)
	NoMaD [29]	25.0 (22.2)	9.3 (8.2)	14.2 (11.2)	12.7 (10.0)	4.4 (4.0)	4.4 (3.9)	10.3 (8.9)	8.3 (7.1)	23.5 (16.3)	14.7 (12.3)
	FAINT [32]	60.3 (59.0)	34.8 (31.6)	46.1 (40.6)	28.9 (25.3)	7.8 (7.1)	11.3 (10.3)	10.8 (9.6)	10.8 (9.3)	36.3 (32.4)	26.5 (23.5)
	LoTIS (Ours)	**98.5** (98.4)	**90.2** (90.1)	**74.0** (72.3)	**74.5** (73.0)	**81.9** (79.4)	**69.6** (67.6)	**69.6** (65.6)	**65.2** (61.6)	**71.1** (69.5)	**71.6** (70.4)
	+ Obstcl. Avoidance	**100.0** (99.8)	**97.5** (96.7)	**95.1** (89.6)	**92.2** (85.1)	**96.1** (90.7)	**84.8** (80.2)	**92.6** (83.2)	**86.8** (76.2)	**94.1** (90.0)	**94.1** (88.2)

We use the yaw controller with constant forward velocity, see Sec. IV-D, paired with LoTIS for our simulation experiments. We also report LoTIS and the controller paired with simple reactive obstacle avoidance to correct movements to ensure collision-free movement. Baselines integrate collision handling into their learned policies and cannot easily be augmented with external modules, illustrating a practical benefit of decoupling perception and control.

B. Results

We summarize our simulation results in Table I.

1) Forward On-Trajectory Navigation: We first evaluate the standard navigation task: following a reference trajectory forward from an on-trajectory start. LoTIS achieves a 94.7% success rate (SR) on Gibson and 98.5% on HM3D, outperforming the strongest baseline (ViNT) by 24.3 and 32.8 percentage points, respectively. Notably, when paired with simple obstacle avoidance, our method achieves 100% SR on both datasets. We hypothesize that this performance gap is largely due to how the reference trajectory is processed: while baselines extract a single subgoal, LoTIS localizes the entire sequence in image-space. This likely provides a much richer and more consistent guidance signal, which even a basic downstream controller (controlling only yaw angle) can exploit to achieve superior performance.

2) Off-Trajectory Initialization: A common failure mode for visual navigation is the inability to recover when the robot begins far from the reference trajectory. As shown in the results, baseline performance drops sharply in the "Off-Trajectory" setting. For example, ViNT's success rate on HM3D falls from 65.7% to 24.0%.

Our model remains robust in these settings, maintaining 88.2% SR on Gibson. We observe lower off-trajectory performance on HM3D compared to Gibson (74.0% vs. 88.2%), which we attribute to HM3D's more cluttered environments. We note that the naive yaw controller at times causes the robot to get stuck behind small obstacles while attempting to return to the trajectory. By integrating basic obstacle avoidance, our success rate increases to 98.7% (Gibson) and 95.1% (HM3D).

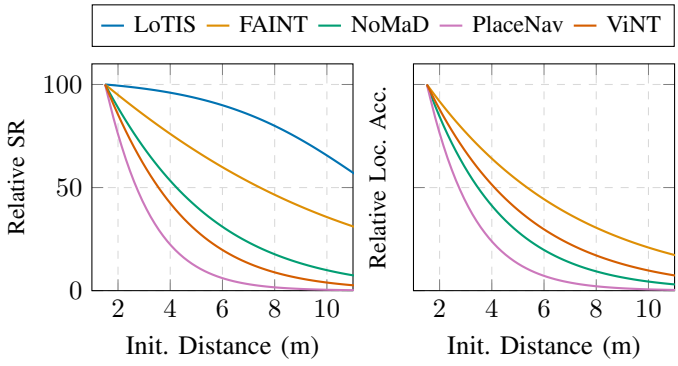

Fig. 3. Relative success rate (SR) for all methods on off-trajectory initialization over initialization distance (left), compared to subgoal localization accuracy for baseline methods (right). LoTIS does not perform discrete subgoal selection and is therefore omitted from the right panel.

This performance suggests that our model's predictions remain accurate even from distant viewpoints where baselines get lost. We further explore this in Fig. 3, which shows the localization accuracy of the baselines (right), measured by correctly identifying the closest image of the reference trajectory, compared to their respective localization accuracy for on-trajectory following. We observe that this accuracy drops rapidly as the initialization distance increases, which leads to substantial degradation of performance in SR (left). By contrast, LoTIS's ability to localize the full trajectory appears to be substantially more robust, resulting in less sensitivity with respect to the initialization distance.

3) Camera Mismatch Robustness: The "Cross" columns in Table I evaluate each method's ability to handle mismatches in camera FOV, aspect ratio and camera mounting height. We observe a significant performance drop for all baselines here, e.g. ViNT's performance on Gibson drops from 70.4% to 21.1% in SR when the camera changes.

In contrast, LoTIS maintains high performance across all evaluations (e.g. 83.6% SR on Gibson, and 98.0% when

paired with obstacle avoidance). We attribute this to our cross-trajectory training strategy, which likely helps increase robustness for the model against camera mismatch between query and reference trajectory. We note that our method is more sensitive to large height mismatches than FOV or AR mismatch, as this can lead to the trajectory being outside the field of view of the robot which may result in the robot getting lost. We provide a more detailed analysis of each method's sensitivity to different parameter mismatches in the supplementary material.

4) Backward Navigation: We evaluate the methods on the "To Start" task, where the robot must navigate to the first image of the reference trajectory. To successfully do this, the robot must be able to understand views that oppose the images of the reference trajectory. The results for this task are summarized in the "To Start" columns of Table I.

We observe a significant performance gap between LoTIS and the baseline methods in this setting. On the Gibson dataset (On-Trajectory/Matched),LoTIS achieves a 88.2% success rate (SR) or 97.4% when paired with obstacle avoidance. In contrast, the baselines largely fail: ViNT achieves 1.3%, NoMaD 8.6%, and FAINT 11.8%. This trend remains consistent across the HM3D dataset, where LoTIS maintains an 96.1% SR while all baselines remain below 8%. We note that here, even if subgoal selection is accurate, the baselines still largely fail due to the policy not being able to predict appropriate actions due to the robot's view opposing the chosen subgoal image.

When moving to the most challenging Off-Trajectory+Cross-Camera setting for backward navigation, LoTIS performance experiences a moderate decrease but remains mostly successful, with success rates between 65.2% and 86.8% (86.8% to 90.8% when paired with obstacle avoidance). Meanwhile, the success rates of the baseline methods stay within the 2% to 13% range.

We note that LoTIS achieves this without being trained on explicit backward traversal demonstrations. Backward traversal is difficult for subgoal-based policies because the current view and selected reference image may correspond to opposing travel directions, yielding limited visual overlap. Moreover, their action policies are trained on same-trajectory current-goal-action tuples and therefore do not observe such opposing-view pairs during training. In contrast, our cross-trajectory training does not require action annotations and naturally includes query views from other trajectories that observe the reference trajectory from different directions. Combined with joint trajectory processing, this allows LoTIS to provide meaningful guidance even when individual frame-to-frame matches are ambiguous.

5) Navigation to Arbitrary Goals (Any Point): The Any Point setting represents a more general usage of a reference trajectory where the robot is initialized off-trajectory and tasked to reach a certain point on the reference trajectory. This task serves as a summary for each method's performance as it requires strong capabilities in each of the aforementioned tasks. As shown in the rightmost columns of Table I, LoTIS maintains high performance in this setting, achieving an 85.5% SR on Gibson and 71.1% on HM3D (increasing to >94% with obstacle avoidance).

In summary, the results demonstrate that LoTIS paired with only a simple yaw controller achieves substantially higher success rates than the baselines across all evaluations, while enabling backward traversal where baselines largely fail. When further paired with reactive collision avoidance, this leads to robust overall performance with over 95% success rates across most experiments.

VI. Real-World Experiments

We evaluate our method in real-world scenarios to answer:

Q1: How does the performance transfer from simulation to real-world deployment?

Q2: How well does our method allow transfer to diverse embodiments (quadrotor, quadruped) from phone-recorded trajectories?

Q3: How do environmental variations, such as dynamic occlusions and day-night lighting changes, impact navigation performance?

A. Evaluation Setup

We collect four reference trajectories using a handheld Google Pixel 6 smartphone: two indoors and two outdoors, see Fig 4. We evaluate on a Crazyflie quadrotor with an analog FPV camera with MPPI control (Sec. IV-D) for indoors trajectories, and on a Boston Dynamics Spot equipped with a Realsense D455 with our yaw controller (Sec. IV-D), but leave its on-board collision avoidance enabled. All trials initialize off-trajectory. We compare against FAINT [32], the strongest baseline in the off-trajectory cross-camera simulation setting. Each trajectory is evaluated with 6 trials (indoors) or 3 trials (outdoors) per direction, totaling 36 runs per method.

For the Spot, we run our method on a Jetson Orin AGX, and for the Crazyflie, we run computation off-board on a desktop with an RTX 5090. We provide videos of all real-world evaluations in the supplementary material.

B. Runtime

LoTIS uses an asymmetric deployment pipeline. When a reference trajectory is provided, all reference images are processed once by the frozen DINOv3 backbone and trajectory encoder $\mathcal{E}_T$. The resulting trajectory features F_T are kept in memory and reused throughout navigation. This offline step is performed before online inference begins. During online deployment, each incoming camera image is processed only by the query encoder, query-trajectory fusion module, and prediction head, producing image-space trajectory predictions for the controller.

We report timings for both real-world compute platforms. On the Jetson Orin AGX used onboard Spot, offline trajectory encoding takes $120\,\mathrm{ms}$ for a trajectory of up to 40 frames, while online query encoding and decoding takes $45\,\mathrm{ms}$ per frame, yielding approximately $22\,\mathrm{Hz}$ control-rate inference. On the RTX 5090 desktop used for the Crazyflie experiments, offline trajectory encoding takes $15\,\mathrm{ms}$ and online inference

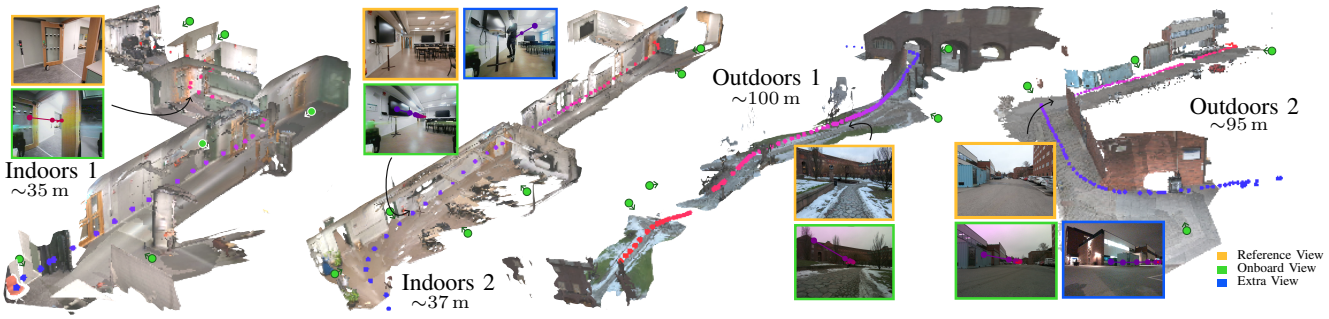

Fig. 4. Four trajectories used for real-world evaluation. Each reference trajectory starts at ● and ends at ●, with initial experiment positions shown as ●. We present an offline-computed reconstruction of the environments [19] alongside representative views from: reference trajectory camera, on-board camera (analog FPV for indoors, RealSense D455 for outdoors), and additional robustness study viewpoints. Our model's trajectory predictions for the on-board cameras are overlaid in the corresponding views.

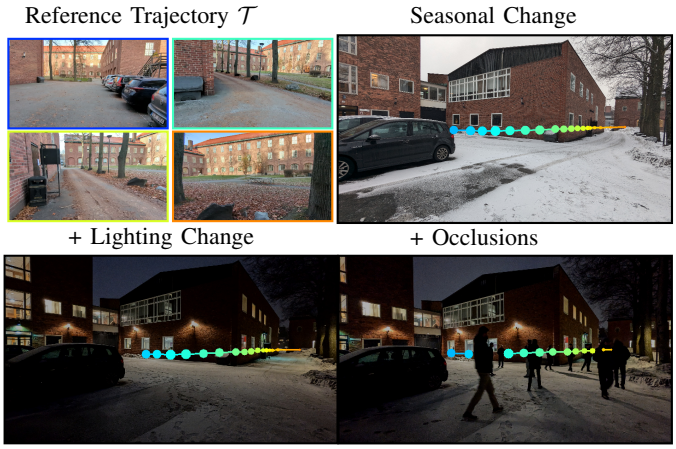

Fig. 5. Impact of environment changes on the predictions of our model with respect to a reference trajectory recorded on a sunny autumn day. Top Right: Seasonal Change, Bottom Left: Seasonal and day-night change, Bottom Right: Seasonal, day-night change and people occluding the view.

takes 6 ms per frame, yielding approximately 160 Hz. The offline timing includes DINOv3 reference feature extraction and the trajectory encoder forward pass, but is not part of the per-frame control loop. All reported timings use trajectories subsampled to 40 frames.

C. Main Results

TABLE II
REAL-WORLD NAVIGATION RESULTS. ALL TRIALS USE PHONE-RECORDED TRAJECTORIES WITH OFF-TRAJECTORY INITIALIZATION. INDOORS: CRAZYFLIE WITH MPPI. OUTDOORS: SPOT WITH YAW CONTROLLER.

	Indoors 1		Indoors 2		Outdoors 1		Outdoors 2	
	Fwd	Bwd	Fwd	Bwd	Fwd	Bwd	Fwd	Bwd
FAINT [32]	3/6	0/6	1/6	1/6	1/3	0/3	**3/3**	0/3
LoTIS (Ours)	**6/6**	**5/6**	**6/6**	**6/6**	**3/3**	**3/3**	**3/3**	**3/3**

As shown in Table II, our method achieves 100% success on forward navigation across all environments, while FAINT succeeds in only 27.8% of trials. This matches, or outperforms the results obtained in simulation. For indoors, we attribute the improved real-world performance compared to sim to the MPPI's ability to keep the predicted trajectory centered in view by actively matching the reference height. We note that for fairness, we manually adjust the drone's flight height to match the recording height for FAINT since FAINT solely provides actions in the horizontal plane. We show that the yaw controller is sufficient for the evaluated outdoors settings due to larger open spaces where Spot's internal reactive collision avoidance is sufficient, though it was not required to intervene during our evaluations. For both indoors and outdoors, we observe that FAINT being unable to reliably localize from off-trajectory initializations, is a main cause of failure.

The gap widens further on backward traversal: our method maintains 94.4% success (17/18), whereas FAINT largely fails (5.6%). These results match our simulation findings and suggest that our cross-trajectory training strategy and joint processing of the full trajectory enables the challenging task of backward traversal. Though performance remains consistent overall, we occasionally encounter views during backward traversal where our model can no longer match the view, and thus predicts no visible points. For the indoors setting, the MPPI is warm-started by the previous solution and will thus continue following the previous solutions, usually leading to better views and recovery within a few steps. For outdoors, we hypothesize that larger open spaces are less prone to produce views with no or little overlap with the trajectory.

TABLE III
ROBUSTNESS TO ENVIRONMENTAL CHANGES. **CROWDED ENV**: INDOORS 2 WITH AND WITHOUT PEOPLE OCCLUDING THE VIEW. **DAY→NIGHT**: OUTDOORS 2 EVALUATED AT NIGHT WITH SCENE CHANGES.

	Crowded Env		**Day→Night**	
	Clean	+People	Day	Night
Forward	6/6	5/6	3/3	3/3
Backward	6/6	5/6	3/3	2/3

D. Robustness Experiments

Table III evaluates robustness under challenging conditions, and we provide an example of our model queried for representative challenging conditions in Fig. 5. We study two changes for the navigating agent: 1.) We repeat the Indoors 2 evaluation with people present in the environment, at times occluding the robot's view and blocking its path, and 2.) We repeat the Outdoors 2 evaluation but with changes in scene (crowded parking lot is now empty) and at night time. Note that the reference trajectory remains the same as in our original

evaluation, i.e. no people for indoors and a crowded parking lot at daytime for outdoors. When people walk through the scene and temporarily occlude the camera view, our method maintains high success (5/6 forward, 5/6 backward). The model's predictions remain accurate, even if a large proportion of the view is occluded and correctly predicts points that are still visible. The one additional failure case can be attributed to people walking in the drone's path in a way that the drone needs to steer away from the reference trajectory, ultimately losing view of the trajectory and being unable to recover (see videos in supplementary). For day-to-night transfer with simultaneous scene changes (trajectory recorded in a daytime parking lot with cars, evaluated at night with the lot empty), we achieve 6/6 forward and 5/6 backward. The one failure case for day-night can be attributed to one initial condition whose view was largely impacted by the scene changes, which leads to the system not being able to recognize any points of the trajectory. While we observe some degradation in the quality of the predictions for the day-night transfer, the results demonstrate that our learned representation is robust to appearance and geometric variation.

VII. Conclusion

We presented LoTIS, a model for visual navigation that predicts where the reference trajectory would appear in the robot's current view. Our approach provides zero-shot guidance for different embodiments, and enables backward traversal and robustness to camera mismatch—capabilities difficult for prior end-to-end methods. Experiments demonstrate 94-98% success on forward navigation across diverse embodiments both in simulation and real-world, and $5\times$ improvement on backward traversal, with real-world deployment confirming transfer to both quadrotor and quadruped platforms using phone-recorded trajectories. For the first time, we used a single RGB reference trajectory in the general setting where the robot can navigate from anywhere in the vicinity to any point on the trajectory, both forward and backward.

Limitations. Our method requires visual overlap between the current view and some portion of the reference trajectory; failures occur when this overlap is lost, particularly during backward traversal around sharp corners or under extreme viewpoint differences. Performance degrades with large camera height mismatches (see supplementary material). Our real-world evaluation is limited to four environments, and does not exhaustively cover the diversity of challenging scenarios (e.g., unstructured environments, or highly repetitive environments) where failure modes may differ. We evaluated trajectories up to $\sim 100\,\mathrm{m}$ subsampled to 40 frames. Longer trajectories require chunking to ensure coverage, which we demonstrate at kilometer-scale in supplementary video but do not systematically evaluate. LoTIS assumes a pre-recorded trajectory with visual overlap to the current view, and navigation to entirely unseen goals remains outside our setting.

Future work. Incorporating temporal memory could help recover when the trajectory temporarily leaves the field of view. Learning to provide guidance even without a clear view of the trajectory could address the limitation of height mismatch. Finally, combining image-space guidance with semantic understanding could enable navigation to functional goals (e.g., "the kitchen") rather than trajectory indices.

References

[1] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4878–4888, 2022.

[2] Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11080–11090, 2023.

[3] Mateo Guaman Castro, Sidharth Rajagopal, Daniel Gorbatov, Matt Schmittle, Rohan Baijal, Octi Zhang, Rosario Scalise, Sidharth Talia, Emma Romig, Celso de Melo, et al. Vamos: A hierarchical vision-language-action model for capability-modulated and steerable navigation. *arXiv preprint arXiv:2510.20818*, 2025.

[4] Francois Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78. Springer, 2007.

[5] Francois Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006. doi: 10.1109/MRA.2006.250573.

[6] Francois Chaumette and Seth Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007. doi: 10.1109/MRA.2007.339609.

[7] Lizhang Chen, Jonathan Li, Kaizhao Liang, Baiyu Su, Cong Xie, Nuo Wang Pierse, Chen Liang, Ni Lao, and Qiang Liu. Cautious weight decay. *arXiv preprint arXiv:2510.12402*, 2025.

[8] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. In *Workshop on Geometric Reasoning for Perception and Action*, pages 106–136. Springer, 1991.

[9] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[10] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

[11] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for

objectgoal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16384–16393, 2024.

[12] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[13] Jacob Krantz, Stefan Lee, Jitendra Malik, Dhruv Batra, and Devendra Singh Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances. *arXiv preprint arXiv:2211.15876*, 2022.

[14] Jacob Krantz, Theophile Gervet, Karmesh Yadav, Austin Wang, Chris Paxton, Roozbeh Mottaghi, Dhruv Batra, Jitendra Malik, Stefan Lee, and Devendra Singh Chaplot. Navigating to objects specified by images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10916–10925, 2023.

[15] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024.

[16] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[17] Yuanzhi Liu, Yujia Fu, Minghui Qin, Yufeng Xu, Baoxin Xu, Fengdong Chen, Bart Goossens, Poly ZH Sun, Hongwei Yu, Chun Liu, et al. Botanicgarden: A high-quality dataset for robot navigation in unstructured natural environments. *IEEE Robotics and Automation Letters*, 9(3):2798–2805, 2024.

[18] Ihab S Mohamed, Guillaume Allibert, and Philippe Martinet. Sampling-based mpc for constrained vision based control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3753–3758. IEEE, 2021.

[19] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[20] Manthan Patel, Fan Yang, Yuheng Qiu, Cesar Cadena, Sebastian Scherer, Marco Hutter, and Wenshan Wang. Tartanground: A large-scale dataset for ground robot perception and navigation. *arXiv preprint arXiv:2505.10696*, 2025.

[21] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[22] Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. Unidepthv2: Universal monocular metric depth estimation made simpler. *arXiv preprint arXiv:2502.20110*, 2025.

[23] Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

[24] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[25] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[26] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[27] Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. In *Conference on Robot Learning*, pages 724–748. PMLR, 2025.

[28] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[29] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[30] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[31] Lauri Suomela, Jussi Kalliola, Harry Edelman, and Joni-Kristian Kämäräinen. Placenav: Topological navigation through place recognition. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5205–5213. IEEE, 2024.

[32] Lauri Suomela, Sasanka Kuruppu Arachchige, German F. Torres, Harry Edelman, and Joni-Kristian Kämäräinen. Synthetic vs. real training data for visual navigation. In *arXiv preprint arXiv:2509.11791*, 2025.

[33] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[34] Niklas Trekel, Tiziano Guadagnino, Thomas Läbe, Louis Wiesmann, Perrine Aguiar, Jens Behley, and Cyrill Stachniss. Benchmark for evaluating long-term localization in indoor environments under substantial static and dynamic scene changes. In *2025 IEEE/RSJ International Confer-*

ence on Intelligent Robots and Systems (IROS), pages 10770–10777. IEEE, 2025.

[35] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[36] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.

[37] Zehan Wang, Siyu Chen, Lihe Yang, Jialei Wang, Ziang Zhang, Hengshuang Zhao, and Zhou Zhao. Depth anything with any prior. *arXiv preprint arXiv:2505.10565*, 2025.

[38] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[39] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[40] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[41] Arthur Zhang, Chaitanya Eranki, Christina Zhang, Ji-Hwan Park, Raymond Hong, Pranav Kalyani, Lochana Kalyanaraman, Arsh Gamare, Arnav Bagad, Maria Esteva, et al. Toward robust robot 3-d perception in urban environments: The ut campus object dataset. *IEEE Transactions on Robotics*, 40:3322–3340, 2024.

[42] Arthur Zhang, Xiangyun Meng, Luca Calliari, Dong-Ki Kim, Shayegan Omidshafiei, Joydeep Biswas, Ali Agha, and Amirreza Shaban. Ventura: Adapting image diffusion models for unified task conditioned navigation. *arXiv preprint arXiv:2510.01388*, 2025.