

Plann3r: Predicting Planning Costs Grounded in 3D

Aditya Vadali^{1,†} Krish Pandya^{1,†} Vansh Garg¹ Nishant Chandna² Dustin Craggs³
Feras Dayoub³ Madhava Krishna¹ Sourav Garg¹

¹IIT Hyderabad, India

²DTU, India

³Adelaide Uni., Australia

[†]Equal contribution

Abstract: Planning paths or trajectories for robot navigation requires understanding scene geometry and traversability. Classic approaches rely on accurate 3D maps to define occupancy based planning costs. Learning-based alternatives predict planning costs either in terms of distance to goal or temporal distance between images. However, the former overfits to the scene layout and the latter lacks geometric understanding. Moreover, most of these methods estimate *image-level* scalar costs, which are not sufficiently informative to effectively guide the robot. In this paper, we propose **Plann3r** as a 3D grounded method to predict *pixel-level* planning costs in terms of geodesic distances for any given set of images with an arbitrary subgoal pixel. We integrate Plann3r within a navigation pipeline, dubbed **VGGT-Nav**, in two unique ways: the offline mapping cum global planning phase iteratively uses Plann3r to generate subgoals and reference image global costmaps; and in the execution phase Plann3r simultaneously performs localization and planning to generate local costmaps that directly condition a learnt control policy. We show that the proposed Plann3r and VGGT-Nav largely outperform baselines respectively on the planning and navigation tasks on an existing prior-traversal navigation benchmark. We further show real world demonstrations that highlight sim-to-real deployment. The code, models, and demos are publicly available at <https://plann3r.github.io/>.

Keywords: Visual Navigation, Path Planning, Action Prediction

1 Introduction

Visual navigation using only a single RGB camera is a challenging task. It requires an effective representation of the world that can subsequently enable robust localization, planning and control. Even the recent advances in 3D mapping based on learnt feedforward techniques [1, 2, 3, 4, 5, 6] do not directly inform about navigability or “plannability”. This generally requires reasoning about traversable regions in a geodesic space as opposed to Euclidean space. In this paper, for the first time, we fill this gap through a geometry-grounded feedforward planning technique, dubbed “Plann3r”, which predicts per-pixel planning costs in the form of geodesic distances. These distances are predicted from query image pixels to a randomly sampled subgoal pixel within the given submap.

Plann3r differs from prior works on cost prediction in the following ways. Predicting *temporal distance* between image pairs [7, 8, 9] only provides a scalar cost which is severely limited to image-level topology, thus ineffective beyond teach-and-repeat tasks [10]. On the other hand, predicting *distance to goal* [11, 12] generally requires overfitting to environment layout types. In contrast, we predict dense pixel-level costmap images as a form of interpretable geodesic navigation affordances, which can then be relayed to the learnt controller to predict cost-informed waypoints. Furthermore, Plann3r uniquely enables *simultaneous mapping/localization and planning* as the query costmap is implicitly based on query-to-submap cross-attention; this forgoes the need for explicit pairwise feature matching, which is more prone to failures.

We make the following key contributions:

- a method, dubbed **Plann3r**, to predict planning costs in the form of pixel-level geodesic distances for a query image to an arbitrary subgoal pixel given within a submap; and
- a single-camera visual navigation pipeline, dubbed **VGGT-Nav**, that leverages 3D grounded backend for mapping, localization and planning, which enables improved control prediction and navigation success rates across multiple tasks of a navigation benchmark based on prior traversals (i.e., seen environment).

2 Related Works

Topological versus Metric Mapping: Visual navigation systems commonly represent prior experience as either metric or topological maps of previously seen environments. Metric approaches build explicit geometric representations, which are particularly *globally* geometrically consistent, such as occupancy grids, point clouds, or 3D graphs [13, 14, 15, 16, 17, 18, 19, 20, 21], and often use classical planners to compute traversable paths. While geometrically grounded, these methods depend on accurate pose estimation, map fusion, and free-space inference, so reconstruction or localization errors can directly degrade planning.

Topological approaches instead represent the environment as a graph of images, objects, or landmarks collected during prior experience [7, 22, 8, 10]. While these methods avoid the need for globally consistent metric reconstruction, resorting to image-or object-level nodes discards dense spatial structure resulting in sparse planning cues. Plann3r bridges these paradigms by using geometry-aware multi-view features from VGGT while predicting dense pixel-level geodesic costmaps over a local submap, rather than planning only over discrete image or object nodes.

Learning to Estimate Planning Costs: Augmenting conventional planning with *learned* cost prediction has become a common technique in visual navigation, particularly in topology-based methods that avoid estimating global metric poses. Approaches based on prior traversals commonly train a predictor to estimate the temporal distance between two images [7, 22, 8, 23]. This predictor is applied to the agent’s prior experience to establish connectivity and edge weights in an image topological map, which is then used for planning. In exploration tasks, costs must instead be predicted for an *unvisited* goal. A number of methods use pre-trained models like CLIP [24] as zero-shot heuristics to assign scores to images [25] or spatial regions [26, 27, 28, 29, 30, 31, 18, 32, 33] that guide planning; however, these scores typically emphasise visual or semantic similarity that does not necessarily align with actual navigation planning cost. NRNS [11] and OVG-Nav [12] instead construct an image topology online during exploration, and predict costs conditioned jointly on the entire map to an unvisited image [11] or object [12] goal. The agent then plans a path through the topology to the image with the lowest predicted cost. Several of these approaches can systematically overestimate planning costs by training on sub-optimal trajectories [7, 22, 8] or failing to identify connections between nodes [11]. Plann3r overcomes these limitations by using true geodesic cost targets, which becomes especially important when predicting costs over longer trajectories. Further, prior methods lack multi-frame context [7, 22, 8, 23] and suffer from a representational bottleneck at the image level [11, 12], limiting their ability to use detailed geometric context to estimate navigation costs. In contrast, Plann3r’s multi-frame context and VGGT backbone preserve the granular 3D representations necessary to produce more informative, fine-grained predicted costmaps.

3 Approach

3.1 Problem Formulation and Solution Overview

Given a prior traversal represented as reference images $\mathcal{M} = \{I_i\}_{i=1}^N$, a current query image I_q , and a goal pixel p_{goal} selected in one of the reference images, our objective is to reach the goal by iteratively predicting planning costs and control waypoints for I_q as the robot moves through the

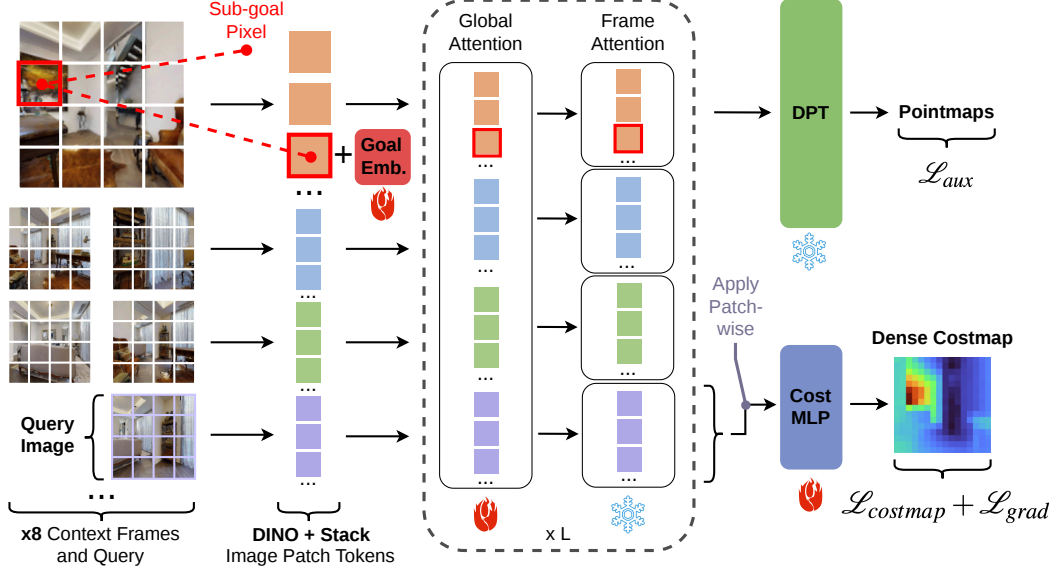


Figure 1: **Plann3r Architecture.** We build Plann3r on top of a pre-trained VGGT model, fine-tuning it to predict geometrically consistent planning costs in terms of geodesic distances from query image pixels to a (sub)goal pixel belonging to one of the submap images.

environment. We achieve this through a three-phased approach [10]: *Preprocessing/Mapping* - we use the given prior information, i.e., the reference images and goal pixel, to convert them into a *functional* map representation in the form of pixel-level planning costs. *Training Phase* - we train two modules: one for planning that can predict geodesic costs from any pixel to the (sub)goal pixel in a given set of images, and the other for control that predicts a trajectory rollout conditioned on the planning costmaps. *Execution Phase* - during live operation, as the robot moves through the environment, we iteratively process its current RGB image along with the submap images and the subgoal to predict both the planning costs and the egocentric waypoints, until it reaches the goal or exceeds its steps limit.

Similar to the prior works, RoboHop [34] and ObjectReact [10], we represent the planning costs through an egocentric costmap $C \in \mathbb{R}^{H/P \times W/P}$ with a scalar cost per patch, where H and W are the input image height and width, and P is the edge length of a square patch. However, unlike ObjectReact’s *object-level* 3D *Euclidean* costs, we use Plann3r to predict *geodesic distances*.

3.2 Plann3r: Architecture and Training

We propose **Plann3r** to predict geodesic distances from any arbitrary pixel in an image to a given (sub)goal pixel within a set of images, as illustrated in Figure 1. The model architecture of Plann3r is based on VGGT [2] which employs a large feed-forward architecture to directly infer 3D scene attributes from multiple views. The VGGT backbone first processes input images by patchifying them into tokens via a pretrained DINOv2 model [35]. It then augments each frame’s tokens with one dedicated camera token and four learnable register tokens. Its training continuously alternates between frame-wise self-attention and global self-attention to effectively associate multi-view scene information. We adapt the VGGT architecture in two key ways: a learnable token, referred to as **goal token**, is used to identify the subgoal, and a custom **MLP decoder** is used to convert output patch embeddings into scalar planning costs.

The goal token is initialized as a learnable vector of the same size as the other input tokens obtained from DINOv2. This goal token is added to one of the patch tokens to designate it as the local subgoal to which the geodesic costs will be predicted.

Our custom decoder is a dedicated, lightweight Multi-Layer Perceptron (MLP) head for costmap prediction. This MLP is comprised of two sequential blocks of fully connected layers with GELU activations in between. Unlike the DPT head of VGGT, which is aimed at dense high-resolution predictions, we are only interested in a coarse signal to direct the robot. Our MLP head directly projects each of the patch embeddings independently (although batch processed) to a scalar cost value, resulting in a costmap of $H/P \times W/P$ resolution. We use VGGT’s patch size of $P = 14$, resulting in a 16×16 costmap for 224×224 input images.

We exclusively train the MLP costmap head and the global self-attention layers of the VGGT backbone. We use a joint objective function that balances direct cost regression, spatial structural consistency, and auxiliary geometric grounding (Section 5.3 ablates loss terms). The total loss $\mathcal{L}$ is:

$$\mathcal{L} = \lambda_{\text{costmap}} \mathcal{L}_{\text{costmap}} + \lambda_{\text{grad}} \mathcal{L}_{\text{grad}} + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}} \quad (1)$$

Costmap Loss ($\mathcal{L}_{\text{costmap}}$): We apply a Huber loss ($\delta = 1.0$) between the valid pixels of the predicted and ground-truth costmaps. To mitigate the outsized influence of extreme cost values, both the predictions and ground truth costmaps are transformed using a logarithmic scale ($\log(1+x)$) prior to calculating the residual.

Gradient Matching Loss ($\mathcal{L}_{\text{grad}}$): To ensure the predicted costmaps preserve critical structural boundaries and obstacles, we enforce an L1 Sobel gradient matching loss. This is computed in a multi-scale fashion across downsampled resolutions (scales of 1, 2, and 4) to capture both fine-grained local obstacles and coarse spatial layouts.

Auxiliary Pointmap Loss ($\mathcal{L}_{\text{aux}}$): To maintain geometric consistency within the network’s representations, we utilize the frozen DPT head to predict query frame world-points. We then apply a masked L1 loss between these predictions and the ground truth, denoted as $\|P - \hat{P}\|_1$. This auxiliary signal preserves the geometric priors in the model’s representations.

3.3 Preprocess Phase: Simultaneous Mapping and Planning

As illustrated in Figure 2, given the reference images $\mathcal{M} = \{I_i\}_{i=1}^N$ from a prior traversal of the environment, we aim to directly represent this prior information in terms of costmap images $C_i \in \mathbb{R}^{H/P \times W/P}$, where each image pixel represents a global path planning cost to a given goal pixel p_{goal} . Unlike all prior work that uses either 3D occupancy grids or topological graphs to represent the map of the environment, we directly convert the previously seen (map) pixels into their functional value, that is, their cost towards the goal. We achieve this through our learnable Plann3r module that operates locally on an image sequence to predict local planning costs, which are then iteratively back propagated from the goal to all the images in the reference traverse using a sliding window with overlapping image frames.

Concretely, we begin from the goal image I_g containing the goal pixel p_g . Around this image, we retrieve a local submap consisting of K temporally adjacent reference images. For every image in this submap, we use Plann3r to predict a patch-level costmap conditioned on the goal pixel p_g . Let the current submap span images $\{I_s, \dots, I_e\}$. After obtaining the predicted costmaps for all images in the window, we select the common image that will connect the current submap to the next propagation step. Specifically, when propagating backward through the traversal, the first image I_s of the current submap becomes the common image for the next submap. Within this common image, we identify the patch with the minimum predicted cost:

$$(p_y, p_x) = \arg \min_{(p_y, p_x)} C_s(p_y, p_x); \quad \alpha_{\text{init}} = \min C_s(p_y, p_x) \quad (2)$$

where C_s denotes the predicted costmap of the common image, where the center of the selected patch location is used as the subgoal pixel for the next submap. The corresponding minimum cost value, denoted with α_{init} , is accumulated as a propagation offset and added to all predicted costs in the subsequent window.

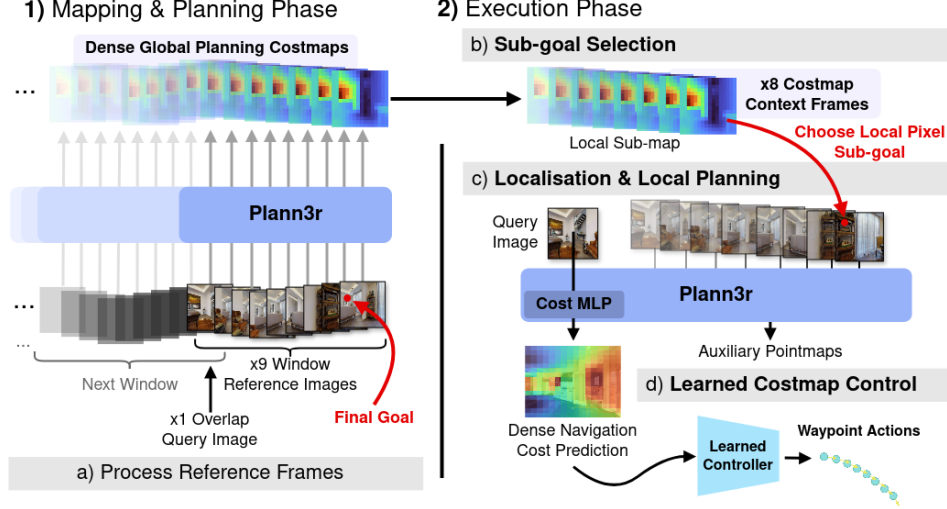


Figure 2: **The VGGT-Nav Pipeline.** We use the proposed Plann3r, based on VGGT, to predict planning costs in terms of geodesic distances from query image pixels to an arbitrary (sub)goal pixel given within a submap. Plann3r is used iteratively both during the offline mapping cum global planning phase and the online execution phase of localization cum local planning, where the output costmaps from the latter are used to predict egocentric trajectory.

The next submap is then formed by shifting the temporal window backward while retaining the common image. This creates a chain of overlapping local planning problems where each window inherits its subgoal from the minimum-cost prediction of the previous window. We repeat this process iteratively until the full traversal sequence has been processed. Formally, if C_i^{local} denotes the locally predicted costmap for image I_i within the current window and α denotes the accumulated propagation offset from previous windows, the propagated global costmap is computed as:

$$C_i^{global} = C_i^{local} + \alpha; \quad \alpha \leftarrow \alpha + \min(C_s) \quad (3)$$

where α is updated after every propagation step using the minimum cost obtained from the common image. The final output of this mapping-cum-planning phase is a set of **precomputed reference costmaps**, each having pixel-wise global planning costs for every reference image in the traversal. Each patch stores the accumulation of predicted geodesic cost to the original goal pixel. Since propagation is performed entirely through overlapping local windows, the procedure only requires local geometric consistency and visual overlap between nearby frames.

3.4 Execution Phase: Subgoal Selection and Cost Prediction

During online execution, the agent must plan its local trajectory based on its current query image, I_{query} , and the prior map. Instead of explicitly matching query pixels to map pixels, we formulate this as a learnt cost prediction task.

Subgoal Selection: We first retrieve a local submap consisting of 8 reference images around the agent’s current location. We use oracle retrieval based on query and map image poses from the simulator for benchmarking and analyses, and separately ablate it against inferred retrieval using MegaLoc [36]. Given the set of submap images, we select the submap image with the lowest median cost in its *precomputed costmap* and use its minimum-cost pixel as the local subgoal.

Localization and Local Planning With the query image, the submap images, and the designated subgoal pixel identified, we pass this triplet into Plann3r. As both the query and submap tokens are passed together, the network’s global attention layers inherently localize the query with respect to the submap based on the visual overlap across the frames, thus forgoing the need for explicit feature

matching. These localized patch tokens of the query are then passed through the custom MLP head to predict the planning costmap for the query image. The predicted costmap is then used to predict a trajectory in the form of local waypoints, as explained next.

Control and Navigation Similar to ObjectReact [10], we define a costmap-conditioned learnt controller, where the key difference lies in the information density of the costmaps. As shown in Figure 3, our predicted costmaps provide dense geometric costs unlike object-level costs in ObjectReact. We inherit the model architecture of the controller and the imitation learning strategy directly from ObjectReact [10]. Briefly, it uses a convolutional encoder to process an image comprising pixel-level costs, which is then decoded by an MLP to output a trajectory, i.e., 5 future 2D waypoints (relative position and yaw) in the local bird’s eye view (BEV) space. It is trained on HM3Dv0.2 trajectories using L2 regression loss that minimizes the error between predicted waypoints $\hat{\mathbf{y}}$ and the optimal ground-truth waypoints $\mathbf{y}$ obtained from the simulator. We validate our approach by comparing it directly to the baseline ObjectReact controller, both of which are trained identically but conditioned on their respective costmap types.

4 Experimental Setup

We provide the implementation details for training the planner and controller in Appendix B. Below, we provide details about navigation tasks and evaluation.

Evaluation Dataset: Following ObjectReact [10], we evaluate on the IIN-val navigation benchmark constructed from HM3D scenes [37]. Each episode provides a prior traversal that is used as the map, and the agent must navigate using only monocular RGB observations at test time. Note that this benchmark is aimed at RGB-only navigation in previously seen environments, unlike the original unexplored-environment setting of Instance ImageNav (IIN) [37]. All navigation episodes are run in Habitat with 320×240 RGB observations, a 120° horizontal field of view, a success threshold of $1m$ with oracle stop condition, and a maximum horizon of 300 steps. We use an oracle retriever to obtain the local submap window from the prior map using query and map image poses; this matches ObjectReact’s controlled evaluation protocol. Given this local submap, Plann3r predicts a dense query costmap conditioned on the selected subgoal pixel, and the learned controller converts this costmap into executable waypoints. We evaluate non-oracle retrieval and stopping in Sec. 5.3.

Navigation Tasks and Evaluation Metrics: We evaluate the same four task variants as ObjectReact [10]. In *Imitate*, the agent follows the demonstrated route from the prior traversal, corresponding to a teach-and-repeat setting. In *Alt Goal*, the goal is visible in the prior map but was not the endpoint of the demonstrated trajectory, requiring the agent to navigate to a previously seen but unvisited target. In *Shortcut*, the map trajectory includes an additional detour through the alternate goal, so the agent should exploit spatial understanding to take a shorter route to the final goal. In *Reverse*, the agent must traverse the route in the opposite direction, testing robustness to viewpoint reversal. We report Success weighted by Path Length (SPL) and Soft-SPL (SSPL), following ObjectReact. We average both metrics over all episodes in each task.

5 Results

5.1 Benchmarking Planning Cost Predictions

We evaluate the ability of our method to estimate geodesic distances by comparing it against the Ground Truth (simulator’s) geodesic distances. We perform this evaluation on pixel-level costmaps (normalized per image) by computing the Mean Absolute Rank Difference (MARD) between the binary masks of the predicted and the GT costmap. The rank refers to the pixel-level cost ordering within a single costmap image, where minimum cost pixel has a rank of 1; these ranks are normalized per image between $[0, 10]$. The binary masks are obtained by selecting either the $k\%$ least-cost pixels or by slabs over k , where $k \in \{5, 15, 30, 50\}$. The former cumulatively evaluates the cost

Table 1: **Quantitative Comparison of Planning Costmaps.** Mean Absolute Rank Difference (MARD) between ground truth and predicted costs (sweeping over min- k % cost pixels or k -based slabs). *Lower is better.* *Left:* cumulative MARD over all pixels below the given cost threshold. *Right:* MARD restricted to pixels within each cost slab.

Costmap Source	ALL PIXELS BELOW THRESHOLD					WITHIN COST RANGE				
	5%	15%	30%	50%	100%	0-5%	5-15%	15-30%	30-50%	50-100%
Euclidean Distance	2.284	2.533	2.617	2.561	2.232	2.236	2.403	2.412	2.296	2.645
ObjectReact [10]	3.650	3.600	3.483	3.260	2.837	3.731	3.312	2.975	2.619	3.304
Plann3r (Ours)	2.398	2.424	2.369	2.299	1.986	2.387	2.262	2.091	2.043	2.318

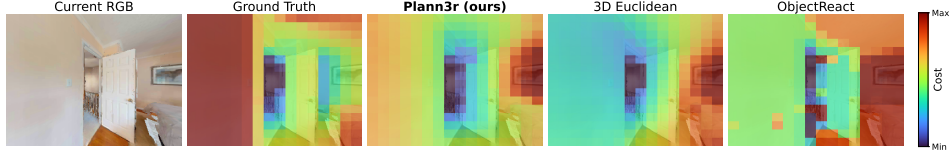


Figure 3: **Qualitative Comparison of Planning Costmaps.** The goal is outside the room and we observe that the side of the wall inside the room gets a high cost (shades of red) in both the Ground Truth and Plann3r generated costmaps, whereas the same wall either gets a low cost (shades of blue) when directly using 3D Euclidean distances or misses a cost gradient in ObjectReact.

predictions spanning from low-cost regions to the entire cost image, whereas the latter evaluates different cost slabs. We report MARD error in Table 1. We compare against ObjectReact which uses monocular depth based 3D Euclidean distances to estimate object-level costs. We also include a simple baseline where the planning costs are represented by Euclidean distance between the VGGT’s 3D projections of the query pixels and the subgoal pixel. This baseline represents a lower bound against geodesic distances rather than a 3D reconstruction-based planner; Plann3r takes a middle ground for path planning which does not explicitly require a *globally-consistent* geometric map, BEV grids, traversability understanding, and 3D pose estimation. Table 1 shows that our proposed method consistently achieves lower error than its Euclidean counterpart as well as ObjectReact. As expected, in the low-cost region surrounding the subgoal (corresponding to lower values of k), Euclidean distances are more accurate, but its utility for planning and navigation becomes worse when reasoning about the rest of the scene (beyond closeby regions) in terms of traversability.

Figure 3 presents a qualitative comparison: the goal is outside the room and it can be observed that the side of the wall visible from the agent’s current location inside the room correctly gets a high cost (shades of red) in both the Ground Truth and Plann3r generated costmaps. On the other hand, the same wall either gets a low cost (shades of blue) when directly using 3D Euclidean distances or misses the cost gradient when using ObjectReact. Additional examples are provided in Appendix A.3.

5.2 Benchmarking Navigation Performance

We evaluate navigation performance on four diverse tasks, as defined in Sec. 4. Table 2 compares our proposed navigation pipeline based on Plann3r against different baselines. GNM [8] and Pix-Nav [38] represent image- and pixel-goal conditioned control policies, while RoboHop [34] and ObjectReact [10] use higher-level object or segment abstractions. In contrast, Plann3r conditions control on dense predicted geodesic costmaps.

It can be observed that our proposed method outperforms prior art in almost all the cases. On the Imitate task, while we achieve the best SSPL, our SPL is slightly below that of the GNM variants, which shows that the agent only missed the last mile of its trajectory for some cases. On the more challenging AltGoal and Shortcut task, we outperform all the baselines with large margins - this shows that geometric grounding of our method enables better spatial understanding than those based

Table 2: **State-of-the-art** comparison on four diverse navigation tasks. Best scores in **bold**.

Method	Imitate		Alt Goal		Shortcut		Reverse		Average	
	SPL	SSPL	SPL	SSPL	SPL	SSPL	SPL	SSPL	SPL	SSPL
GNM [8]	78.79	82.95	8.70	15.44	15.38	31.74	3.33	6.11	26.55	34.56
GNM (HM3D) [10]	81.82	86.38	0.00	10.91	15.38	24.57	13.28	20.77	27.62	35.66
PixNav [38]	42.42	46.75	26.09	31.66	7.69	22.29	16.16	25.56	23.09	31.57
RoboHop [34]	57.56	64.99	30.43	38.23	30.77	40.87	9.98	16.92	32.19	40.25
ObjectReact [10]	60.60	68.51	21.74	26.68	23.08	39.64	30.00	42.01	33.36	44.71
VGGT-Nav (Ours):										
Plann3r + GT-trained Control	61.30	75.53	58.88	69.42	45.36	61.10	0.00	9.80	41.39	53.96
Plann3r + Control	78.43	87.69	65.92	77.78	55.68	70.82	0.00	11.99	50.01	62.07

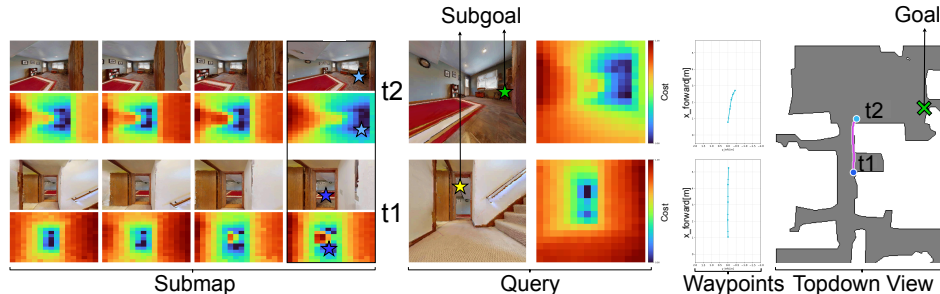


Figure 4: **Qualitative Overview of Navigation Pipeline.** The query costmaps remain geodesically consistent with the submap costmaps and correctly direct the robot at both the timesteps, regardless of the visibility of the final goal in the current image at timestep t1.

on image- [8, 38] or object-level [34, 10] spatial relations. On the Reverse task, our method achieves low performance: this is attributed to incorrect cost prediction due to opposing viewpoints between query image and the submap; this is perhaps expected as our model training only considers similar viewpoints for cost prediction.

Controller Trained on GT Costs: Table 2 also includes a variant (penultimate row) where the controller is trained on the simulator’s navmesh-based ground truth (GT) costmaps, but it still receives Plann3r-predicted costmaps at deployment, i.e., GT costs are used only during controller training, *not inference*. Thus, it only indicates the effect of the controller’s input distribution, and performance expectedly improves when training on Plann3r-predicted costmaps (last row).

Figure 4 presents a qualitative visualization of our navigation pipeline, depicting the submap images, query image, their respective costmaps, subgoal/goal pixels, predicted waypoints and a topdown overview. It can be observed how the query costmap remains consistent with the submap costmaps and correctly directs the robot at both the timesteps, regardless of the visibility of the final goal in the given image at timestep t1. We show **real-world** results on the project page.

5.3 Ablations and Non-Oracle Evaluation

Ablations: Table 3 ablates Plann3r’s training by switching across loss functions (gradient loss $\mathcal{L}_{grad}$ and pointmap loss $\mathcal{L}_{aux}$) and freezing of the VGGT backbone. We find that combining $\mathcal{L}_{grad}$ and $\mathcal{L}_{aux}$ improves SPL (our default) versus applying them in isolation, though ablating these only has a small effect overall. However, freezing the backbone and only training the MLP head dramatically reduces performance, indicating that geodesic affordance gets learnt within the backbone and that too from the middle layers onward (see probing analysis in Appendix A.2).

Non-Oracle Localization and Stopping: Table 3 (penultimate row) drop-in replaces oracle retrieval with MegaLoc [36]. Performance reduces as expected, but the reduction in average SPL is marginal. We further evaluate a non-oracle stopping mechanism leveraging Plann3r’s geometric understanding to identify the final goal: when the localized submap contains the final goal, we iden-

Table 3: **Ablating loss functions, oracle retrieval and oracle stopping condition (SPL).** All variants use $\mathcal{L}_{costmap}$. The first row of the lower block is our default configuration

Backbone	$\mathcal{L}_{grad}$	$\mathcal{L}_{aux}$	Retrieval	Stop	Imitate	Alt Goal	Shortcut	Reverse	Avg
Unfrozen			Oracle	Oracle	73.22	55.65	57.36	0.00	46.56
Unfrozen		✓	Oracle	Oracle	73.72	59.58	56.64	0.00	47.49
Unfrozen	✓		Oracle	Oracle	75.41	67.56	56.98	0.00	49.99
Frozen	✓	✓	Oracle	Oracle	31.58	16.56	11.00	6.32	16.37
Unfrozen	✓	✓	Oracle	Oracle	78.43	65.92	55.68	0.00	50.01
Unfrozen	✓		MegaLoc	Oracle	77.36	66.60	45.61	6.38	48.99
Unfrozen	✓	✓	MegaLoc	Inferred	60.08	2.60	31.24	3.15	24.27

tify pixels with low cost estimates and a distance to the robot of less than $1m$ (using depth from VGGT) to inform the stop condition. Paired with MegaLoc localization, this approach maintains a reasonable SPL of 60.08 on the Imitate task (vs. 77.36 with oracle stopping). However, overall performance reduction indicates the need for a better method for stop condition especially under the influence of viewpoint variations in other three navigation tasks.

6 Limitations

Key limitations of our method are: a) *lack of large-scale training*: we currently only use an indoor sim environment as a proof of concept, however, we show real-world results that demonstrate sim-to-real transfer in environments that differ significantly from the training data; b) *limited pose variations* within the input image set for predicting costmaps: this limitation comes up in the task of reverse traverse but could be addressed by training on large viewpoint variations; c) *oracle retrieval and stop condition* in our main benchmark: although this limits our understanding of absolute performance, we used this to decouple the effect of somewhat complementary issues, and our initial non-oracle variants (Sec. 5.3) leave room for improvement, particularly for stopping; d) *lack of loop closures*: we assume only a single non-overlapping traverse of the environment, even though it provides 3D spatial understanding, it misses the shortcut connections based on loop closures unless such intersections are given (Appendix A.4); e) *goal specification*: we assume a pixel-level goal in a previously traversed reference image; image or segment goals can be converted to pixels (e.g., text-to-pixel cosine similarity using GroundingDINO [39] selects a pixel on the correct object for 26 of 32 instance image goals), but navigating to a target missing from the prior traversal, which lies in between object/image-goal navigation in unexplored scenes and RGB-only navigation in seen scenes, warrants a deeper analysis and a corresponding evaluation protocol; and f) *trajectory-centric nature*: we trained and tested our method in a trajectory-centric manner and not for an unordered collection of images; although planning cost propagation inherently leverages this prior, our study on integrating loop closures (Appendix A.4) addresses this limitation. We provide a detailed account of these limitations in Appendix C.

7 Conclusion and Future Work

We presented Plann3r, a 3D-grounded method for predicting dense pixel-level geodesic planning costs from a query image, local submap, and arbitrary subgoal pixel. We integrated the proposed Plann3r in both the offline phase of mapping cum global planning to produce subgoals, and in the execution phase of localization cum local planning to produce query costmaps. This demonstrates how learning-based spatial understanding can directly provide functional information for the navigation task. Future work can consider large-scale training for a more general solution, potentially also addressing the extremely challenging scenarios of reverse traversing.

References

- [1] V. Leroy, Y. Cabon, and J. Revaud. Grounding image matching in 3D with MAST3R. In *European Conference on Computer Vision (ECCV)*, 2024.
- [2] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [3] R. Murai, E. Dexheimer, and A. 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.
- [4] B. Duisterhof, L. Zust, P. Weinzaepfel, V. Leroy, Y. Cabon, and J. Revaud. MAST3R-SfM: a fully-integrated solution for unconstrained structure-from-motion. In *International Conference on 3D Vision (3DV)*, 2025.
- [5] D. Maggio, H. Lim, and L. Carlone. VGGT-SLAM: Dense rgb slam optimized on the sl (4) manifold. *Advances in Neural Information Processing Systems*, 38, 2025.
- [6] N. Keetha, N. Müller, J. Schönberger, L. Porzi, Y. Zhang, T. Fischer, A. Knapitsch, D. Zauss, E. Weber, N. Antunes, J. Luiten, M. Lopez-Antequera, S. R. Bulò, C. Richardt, D. Ramanan, S. Scherer, and P. Kotschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*. IEEE, 2026.
- [7] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.
- [8] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. 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.
- [9] D. Shah and S. Levine. ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. In *Proceedings of Robotics: Science and Systems*, 2022. URL <http://www.roboticsproceedings.org/rss18/p019.html>.
- [10] S. Garg, D. Craggs, V. Bhat, L. Mares, S. Podgorski, M. Krishna, F. Dayoub, and I. Reid. Objectreact: Learning object-relative control for visual navigation. In *Conference on Robot Learning*. PMLR, 2025.
- [11] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta. No rl, no simulation: Learning to navigate without navigating. *Advances in Neural Information Processing Systems*, 34:26661–26673, 2021.
- [12] H. Yoo, Y. Choi, J. Park, and S. Oh. Commonsense-aware object value graph for object goal navigation. *IEEE Robotics and Automation Letters*, 9(5):4423–4430, 2024.
- [13] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020.
- [14] Q. Gu, A. Kuwajerwala, S. Morin, K. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. de Melo, J. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [15] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013.

- [16] G. Georgakis, B. Bucher, K. Schmeckpeper, S. Singh, and K. Daniilidis. Learning to map for active semantic goal navigation. In *The Tenth International Conference on Learning Representations (ICLR 2022)*, 2022.
- [17] P. Paul, A. Garg, T. Choudhary, A. K. Singh, and K. M. Krishna. Lego-drive: Language-enhanced goal-oriented closed-loop end-to-end autonomous driving. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10020–10026. IEEE, 2024.
- [18] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.
- [19] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.
- [20] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5664–5673, 2019.
- [21] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.
- [22] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.
- [23] J. Krantz, A. Gokaslan, D. Batra, S. Lee, and O. Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [24] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [25] D. Shah, B. Osinski, B. Ichter, and S. Levine. LM-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *6th Annual Conference on Robot Learning*, 2022. URL <https://openreview.net/forum?id=UW5A3SweAH>.
- [26] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher. VLFM: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [27] H. Huang, Y. Hao, C. Wen, A. Tzes, Y. Fang, et al. Gamap: Zero-shot object goal navigation with multi-scale geometric-affordance guidance. *Advances in Neural Information Processing Systems*, 37:39386–39408, 2024.
- [28] F. L. Busch, T. Homberger, J. Ortega-Peimbert, Q. Yang, and O. Andersson. One map to find them all: Real-time open-vocabulary mapping for zero-shot multi-object navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14835–14842. IEEE, 2025.
- [29] J. Ortega-Peimbert, F. L. Busch, T. Homberger, Q. Yang, and O. Andersson. Div-nav: Open-vocabulary spatial relationships for multi-object navigation. *arXiv preprint arXiv:2510.16518*, 2025.

- [30] D. Schwartz, K. Kondo, and J. P. How. Efficient navigation in unknown indoor environments with vision-language models, 2025. URL <https://arxiv.org/abs/2510.04991>.
- [31] Z. Zhou, Y. Hu, L. Zhang, Z. Li, and S. Chen. Beliefmapnav: 3d voxel-based belief map for zero-shot object navigation. *arXiv preprint arXiv:2506.06487*, 2025.
- [32] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.
- [33] B. Sun, H. Chen, S. Leutenegger, C. Cadena, M. Pollefeys, and H. Blum. Frontiernet: Learning visual cues to explore. *IEEE Robotics and Automation Letters*, 10(7):6576–6583, 2025.
- [34] S. Garg, K. Rana, M. Hosseinzadeh, L. Mares, N. Suenderhauf, F. Dayoub, and I. Reid. Robo-hop: Segment-based topological map representation for open-world visual navigation. In *2024 International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [35] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.
- [36] G. Berton and C. Masone. Megaloc: One retrieval to place them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2886–2892, June 2025.
- [37] J. Krantz, S. Lee, J. Malik, D. Batra, and D. S. Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.
- [38] W. Cai, S. Huang, G. Cheng, Y. Long, P. Gao, C. Sun, and H. Dong. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE, 2024.
- [39] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision (ECCV)*, 2024.
- [40] J. Bratulić, S. Mittal, T. Brox, and C. Rupprecht. On geometric understanding and learned priors in feed-forward 3d reconstruction models, 2026. URL <https://arxiv.org/abs/2512.11508>.
- [41] S. Chittawar, V. Garg, A. Vadali, K. Pandya, R. Jayanti, S. Garg, and K. M. Krishna. Pixel-loop: Shortcut topological navigation with pixel-level loops. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Appendix

Here, we present additional results and analyses to support our claims (Appendix A); implementation details for training and evaluation (Appendix B); and detailed discussion on the limitations (Appendix C). An overview video, including real-world deployments, is available on the project website: <https://plann3r.github.io/>.

A Additional Results and Analyses

A.1 VGGT based Global 3D Mapping

Our work is aimed at prior map based navigation tasks, which inherently includes underlying capabilities of localization and path planning. Our prior map is simply an image sequence obtained from a single RGB camera. Although we use an incremental mapping approach for scalability purposes, the alternative is to use the same underlying mapper (VGGT in our case) over the entire mapping sequence to create a globally registered 3D map, dubbed **VGGT-Global** from here on. We compare our proposed method with VGGT-Global in terms of path planning based on *connectivity across the occupied space*. That is, for VGGT-Global, we compute planning costmaps for a map image by directly computing 3D Euclidean distances between the 3D position of the goal and the 3D points corresponding to the image pixels. This approach is in the same vein as ours, but primarily studies the effect of VGGT-Global’s large-scale global mapping ability and lack of any geodesic estimation. As can be observed in Table 4 and qualitatively in Figure 5, the costmaps estimated using VGGT-Global are much inferior to those obtained by Plann3r. Figure 5 shows a topdown view of planning costs, indicating how Euclidean distance based cost estimation in VGGT-Global incorrectly finds a shortcut *through the wall*, whereas the proposed Plann3r predicts a cost gradient along the trajectory.

Table 4: **Quantitative Comparison of Planning Costmaps, including VGGT-Global.** Mean Absolute Rank Difference (MARD) between ground truth and predicted costs (sweeping over min- k % cost pixels or k -based slabs). *Lower is better. Left:* cumulative MARD over all pixels below the given cost threshold. *Right:* MARD restricted to pixels within each cost slab.

Costmap Source	ALL PIXELS BELOW THRESHOLD					WITHIN COST RANGE				
	5%	15%	30%	50%	100%	0–5%	5–15%	15–30%	30–50%	50–100%
Euclidean Distance	2.284	2.533	2.617	2.561	2.232	2.236	2.403	2.412	2.296	2.645
ObjectReact [10]	3.650	3.600	3.483	3.260	2.837	3.731	3.312	2.975	2.619	3.304
VGGT-Global	3.667	3.561	3.391	3.144	2.744	3.675	3.321	2.891	2.513	3.144
Plann3r (Ours)	2.398	2.424	2.369	2.299	1.986	2.387	2.262	2.091	2.043	2.318

A.2 Plann3r Probing

To understand how Plann3r’s finetuning affects the base VGGT model in order to learn geodesic distance predictions, we ran a lightweight layer-wise probing study inspired by recent geometric interpretability work on feed-forward 3D reconstruction models. Bratulić et al. [40] study intermediate VGGT representations by probing for epipolar geometry through fundamental-matrix recovery and Sampson-error evaluation. We use the same broad probing philosophy but change the target: rather than asking whether epipolar geometry is linearly accessible, we ask whether navigation affordance is linearly accessible, i.e., whether geodesic planning cost from a query patch to the subgoal can be decoded from frozen patch tokens from each aggregation layer. We convert the ground-truth geodesic costmap into a patch-level target by log transforming the cost (i.e., $\log(1 + cost)$), masking invalid patches, and converting valid patch values into normalized cost ranks.

For each layer in the VGGT architecture, we fit a small ridge-regression probe [40] from frozen query patch tokens to geodesic cost rank. The probe is weak on purpose: tokens are random-projected to 256 dimensions, and only the linear probe weights are trained. Thus, high probe performance indicates that geodesic planning cost is already accessible in the representation, rather than

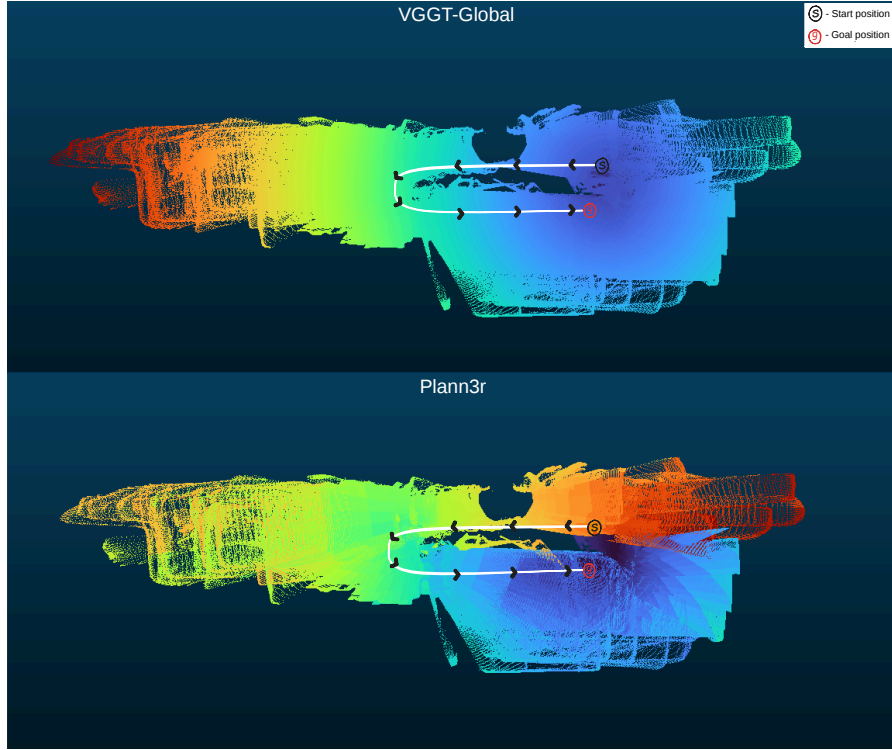


Figure 5: Comparing global planning costs from the start to the goal using VGGT-Global (top) and Plann3r (bottom). It can be observed that simply using Euclidean distance based costs in a globally-registered map (top) finds shortcuts *through the wall*, whereas predicting geodesic costs (bottom) correctly predicts the costs along the actual trajectory.

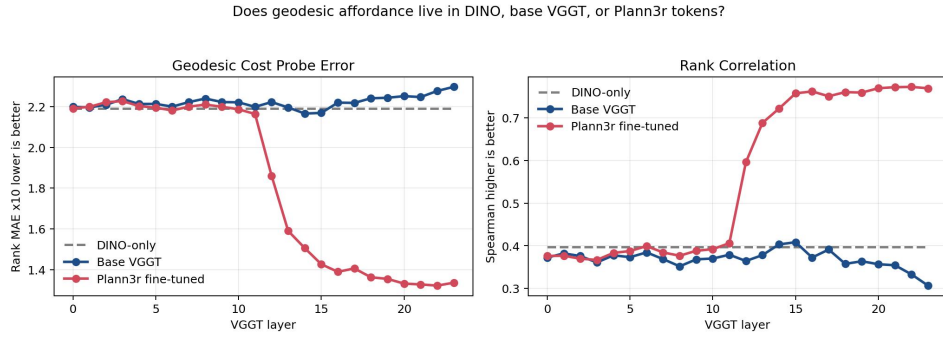


Figure 6: Layer-wise geodesic cost probing for DINO, VGGT, and our finetuned Plann3r.

being synthesized only by a powerful decoder. We evaluate on held-out sampled patches from 128 subtrajectories of the held-out validation scene, giving 16,256 sampled query patches in total. The left plot in Figure 6 reports Mean Absolute Rank Error (Rank MAE) multiplied by 10: this is the absolute error between the probe-predicted normalized geodesic rank and the ground-truth normalized geodesic rank, so lower means the frozen tokens allow a linear probe to predict the ordering of GT geodesic cost more accurately.

The right plot reports Spearman rank correlation between predicted and GT geodesic ranks, so higher means the probe better preserves the ordering of GT geodesic cost across query patches.

We compare three representations using the same method. DINO-only patch tokens test how much navigation cost can be inferred from local appearance and layout cues before multi-view reasoning. Base VGGT tokens test whether pretrained 3D geometric features make planning cost more accessible without task-specific fine-tuning. Plann3r tokens test the effect of goal-conditioned geodesic-cost training with the subgoal anchor token. DINO-only and base VGGT remain close to weak baselines, while Plann3r fine-tuning produces a sharp representational transition around layers 12-14, after which geodesic cost becomes strongly linearly decodable and remains high through the late layers.

Geodesic cost becomes linearly accessible in Plann3r at layers 12 to 15 (Figure 6), while base VGGT does not encode it at any layer. These are the same middle layers where Bratulić et al. [40] find that base VGGT can recover the fundamental matrix (layers 12 to 14). This suggests that finetuning with geodesic costs induces this understanding on top of the geometric understanding that VGGT already forms in these layers.

A.3 Qualitative Comparison with Euclidean Costmaps

Figure 7 qualitatively compares Plann3r costmaps against the 3D Euclidean baseline (Table 1) on the same query image for three goal placements. When the goal lies in either room, Plann3r assigns the highest costs to the dividing wall, whereas Euclidean distances assign it intermediate costs since it is spatially close to the goal, despite not being traversable. When the goal is placed on the wall itself, Plann3r’s minimum-cost patch lies at the base of the wall, i.e., on the navigable floor, consistent with how ground-truth geodesic costs are computed by snapping each pixel to its closest navigable floor point (Appendix B), whereas the Euclidean minimum remains at the goal patch.

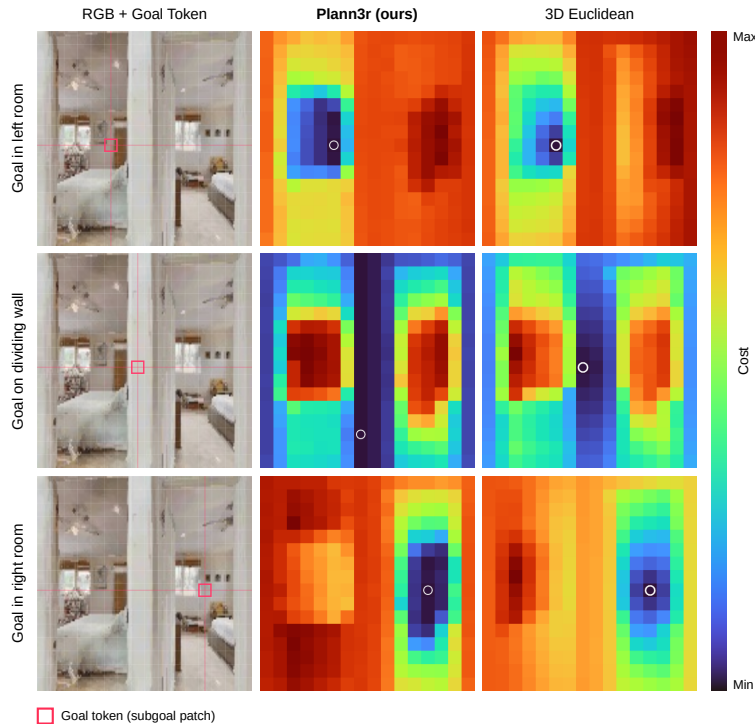


Figure 7: **Plann3r (Geodesic) vs. 3D Euclidean Costmaps.** For the same query image, the goal token (red square) is placed in the left room (top), on the dividing wall (middle) and in the right room (bottom). White circles mark the minimum-cost patch. Plann3r correctly predicts geodesic traversability whereas simply using 3D Euclidean distance provides a misleading cost signal.

A.4 Self-intersecting Traversals

We extend our cost propagation (Sec. 3.3) to self-intersecting traversals when loop closures are given [41]. The reference sequence is generalised to an unweighted image-level graph, with *temporal edges* between consecutive frames and *loop edges* between frames that observe the same place (e.g., using place recognition, retained above a confidence threshold). Propagation then becomes a frontier expansion: from each frontier frame, every simple path of K frames is passed to Plann3r as a window, so loop-spanning windows arise automatically, while subgoal handover and offset accumulation remain unchanged. A frame reached by multiple windows retains the candidate costmap with the lowest median cost, which enables shortcut paths. Figure 8 shows such a resulting shortcut.

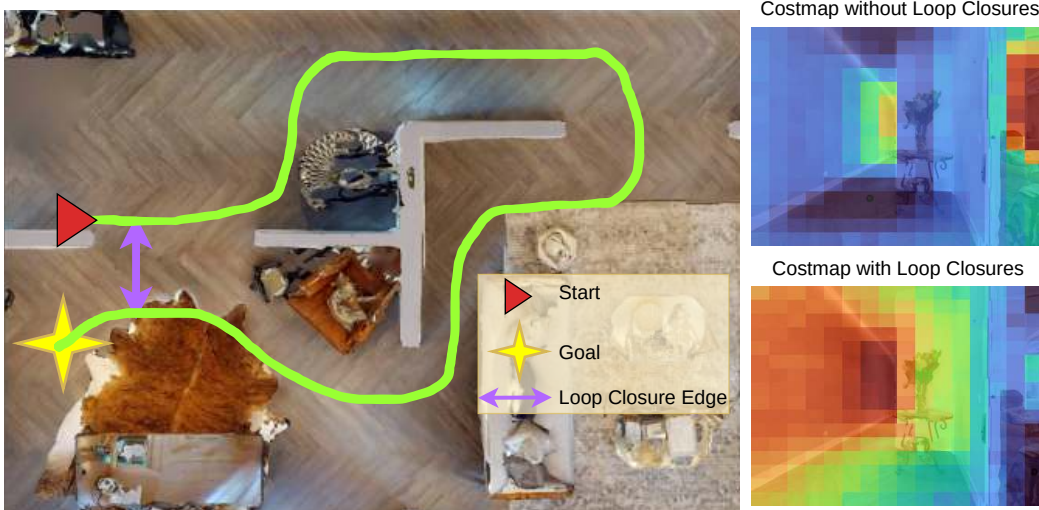


Figure 8: Ego-overlaid costs without (top) and with (bottom) loop closure for a shortcut to the right (top-down view).

B Implementation Details

B.1 Training Data and Parameters

Planning cost prediction. We train Plann3r on subtrajectory windows generated from training trajectories [41] sampled from the HM3D Instance ImageNav dataset [37]. We generated a dataset of 23,508 such samples from the training split. Each sample contains a query RGB image, a local submap of eight reference images, a designated subgoal pixel in one reference image, and a ground-truth geodesic costmap for the query image. We utilize the Habitat simulator along with the navmesh files for computing these ground-truth costmaps. Specifically, the ground truth is computed by snapping the 3D point corresponding to each pixel in the sample’s images to its closest navigable point on the floor (navmesh point). We then assign that pixel a cost equal to the geodesic distance from its corresponding floor point to the floor point of the subgoal pixel. Note that we refer to one of the input images as the “query” mainly to clarify the intent of its usage during inference; while using Plann3r for the preprocessing/mapping stage, we use the “query” as a placeholder for any image for which we want to predict the costmaps.

During training, the network receives one query image and eight submap images, resized to 224×224 . We unfreeze all 24 global attention layers of the VGGT backbone, while the frame-attention blocks, DINOv2 patch embedder, camera/register tokens, and point head remain frozen. To convert the (16, 16, 2048) VGGT output of the query image into a single (16, 16) costmap, we attach a 2-layer MLP head featuring GELU activation. This head takes an input dimension of 2048, projects to a hidden dimension of 8192, and maps to a final output dimension of 1. With the VGGT patch

size $P = 14$, the prediction target is a 16×16 patch-level costmap. This configuration results in approximately 335.9M trainable parameters and 638.3M frozen parameters.

We apply train-time color jitter augmentations (brightness, contrast and saturation at 0.2, hue at 0.05) with a probability of 0.8. We train the model for 18 epochs, which takes approximately 30 hours and consumes 15GB of VRAM on a single NVIDIA A6000 GPU. We optimize with AdamW using a learning rate of 5×10^{-5} , a weight decay of 0.05, bfloat16 mixed precision, and gradient clipping with a maximum global norm of 1.0. The learning rate follows a linear warmup from 10^{-8} for the first 5% of training, followed by cosine decay to 10^{-8} . We utilize a Huber loss with $\delta = 1.0$. The loss weights are set to $\lambda_{costmap} = 1.0$, $\lambda_{grad} = 0.5$, and $\lambda_{aux} = 0.02$.

Controller training. For closed-loop navigation, we train a costmap-conditioned controller following the imitation-learning setup of ObjectReact [10]. The controller observes RGB context frames and a dense planning costmap, and predicts future waypoints in the agent-centric BEV frame. In our current setup, the controller consumes 16×16 per-image normalized costmaps, and predicts 5 future waypoints. Similar to ObjectReact, the controller is trained on HM3D training trajectories with Adam using learning rate 7×10^{-4} , batch size 256, and 30 epochs.

B.2 Evaluation Details

Planning Costmap Evaluation. For the planning cost evaluation, we construct local evaluation windows from the mapping traversals [10] of the HM3D-IIN validation split [37]. Each window contains nine images, matching the training-time input structure of one query image and eight submap images. For each window, we compute simulator geodesic costmaps using the Habitat navigation mesh and use the minimum-cost valid patch across the local window as the subgoal anchor. This gives a controlled setting where all compared cost sources are evaluated with the same image window, subgoal, valid-pixel mask, and patch resolution.

We compare Plann3r against two geometry-based cost sources. The first is a local 3D Euclidean baseline, where VGGT pointmaps are used to project query pixels and the selected subgoal pixel into 3D and each query patch is assigned its Euclidean distance to subgoal. The second is VGGT-Global, which uses the global VGGT reconstruction described in Appendix A.1 and similarly assigns planning costs using 3D Euclidean distance to the goal. These isolate whether metric 3D reconstruction alone is enough for planning cost prediction, without learning geodesic traversability.

All the evaluated costmaps are converted to the common 16×16 patch grid before computing metrics. For each image, valid patches are ranked by cost and the ranks are normalized to $[0, 10]$, where lower rank corresponds to lower planning cost. We then compute Mean Absolute Rank Difference (MARD) between each predicted/baseline costmap and the simulator geodesic costmap. We report MARD over both cumulative low-cost regions, obtained by selecting the lowest $k\%$ patches, and over disjoint cost-rank slabs. The cumulative metric measures whether the method identifies increasingly larger low-cost navigable regions, while the slab metric evaluates errors at different cost ranges.

Navigation Tasks Evaluation. For evaluating navigation tasks, we use the preprocessed prior map costmaps produced by the mapping-cum-planning stage and run the learned controller in closed loop. At each timestep, the current observation is joined with the retrieved local submap and the subgoal pixel selected from the precomputed reference costmaps. Plann3r then predicts a query frame costmap, which is further normalized as per controller training before being passed to the waypoint controller.

The controller outputs five future waypoints in the agent-centric BEV frame. These waypoints are converted to low-level simulator actions using the same execution interface and action constraints as the ObjectReact [10] controller, so differences in navigation performance are attributable to the planning cost representation rather than to a different controller architecture or action execution setup. For the GT-cost ablation (“Plann3r + GT-trained Control” in Table 2), the controller is trained on the

corresponding simulator geodesic costmaps instead of Plann3r-predicted costmaps, while at inference it receives Plann3r-predicted query costmaps; the subgoal selection procedure and execution stack are kept fixed.

B.3 Compute

On an NVIDIA RTX 6000 Ada, mapping-cum-planning stage for a non-intersecting image sequence takes 3.1 s on average across the 32 episodes of the Imitate task, ranging from 1.4 s for the shortest traversal (26 images, spanning 2.34 m) to 6.1 s for the longest (131 images, spanning 20.76 m). During navigation, each step runs a single Plann3r forward pass on the query and eight submap images, which takes 260 ms with 4.06 GiB of peak GPU memory (fp32).

C Detailed Limitations

C.1 Training Data Diversity

Currently, we train Plann3r and the costmap controller on indoor data from simulator for a single embodiment. Despite this, our approach demonstrates strong sim-to-real transfer capabilities to environments and embodiments that differ significantly from the training scenario. This can be attributed to the diverse pre-training of Plann3r’s VGGT backbone, as well as the use of a costmap representation for the controller that provides rich planning costs while being largely invariant to irrelevant visual details of the environment. However, it is likely that performance could be further improved in future work by incorporating training data from a wider set of embodiments and scenes in simulation, as well as real-world demonstration data for training the controller (e.g. datasets used in GNM [8]).

The training data for Plann3r also exhibits limited *trajectory diversity*, because we train exclusively on data from forward-facing traversals. We hypothesise that this restricts the performance of Plann3r on the *reverse* task in particular, where the viewpoint of the robot’s query images differs significantly from the mapping perspective. It may be possible to improve Plann3r’s predictions by collecting training data with more diverse perspectives, for example by randomising the rotation of the agent during data collection.

C.2 Oracle Retrieval and Stopping Condition

In our main experiments, we use odometry data to retrieve a local sub-map for the agent’s context window, as well as an oracle stopping condition that activates when the agent is within 1m of the goal. This allows us to isolate the impact of the learned planning costs from Plann3r. However, future work could instead use Plann3r to create a *local point cloud* based on its context window to address both of these requirements. This point cloud would be used to infer whether the points from the query frame are near those of the current context frames, informing whether the current context window should be shifted forwards or backwards. It could also be used to find geometric correspondences between the query points and goal points, and identify how close the agent is to these points to inform a non-oracle stopping condition. As a first step in this direction, Sec. 5.3 replaces oracle retrieval with MegaLoc [36] and oracle stopping with a condition based on Plann3r’s cost estimates and depth from VGGT.

C.3 Self-intersecting Prior Traversals

VGGT-Nav assumes access to a simple prior traversal of the environment that is not self-intersecting. In cases where the prior traversal’s path intersects with itself, however, obtaining correct geodesic costs requires identifying these intersection points when propagating costs from the goal to identify “shortcut” connections. Shorter time-horizon loops could be identified by expanding or dilating the context window provided to Plann3r. Future approaches could develop ways of identifying loop closures during mapping and planning to propagate minimum costs across each window provided

to Plann3r. Alternatively, rather than pre-computing loop closures, the approach proposed in Appendix C.2 could be adapted to instead *relocalise* during deployment within another, lower-cost window that is still geometrically consistent with the query image. This would allow the agent to “switch” to a more efficient path to the goal at a point where the prior traversal intersects with itself, but would require regular, costly tests across different context windows.