2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

MASt3R-Nav: WayPixel Navigation in Relative 3D Maps

Project Page: `https://mast3r-nav.github.io/`

Vansh Garg[1†], Rohit Jayanti[1*], Krish Pandya[1*], Sarthak Chittawar[1*]
Siddharth Tourani[2,3], Muhammad Haris Khan[3], Sourav Garg[1‡], and Madhava Krishna[1‡]

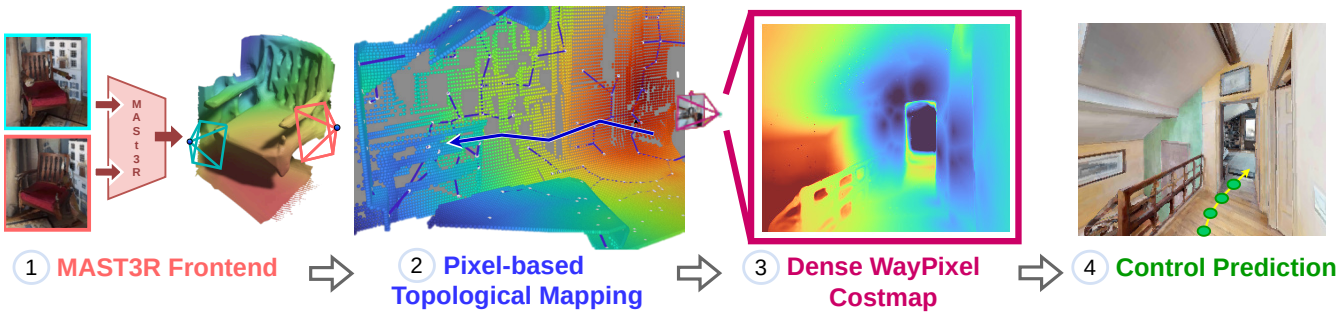

Fig. 1: **Overview of MASt3R-Nav.** ① Consecutive RGB frames are matched using MASt3R to obtain dense pixel correspondences and relative 3D point maps. ② Matched pixels are composed into a pixel-level topological graph, where inter-image correspondences form zero-cost edges and intra-image edges are weighted by relative 3D Euclidean distance. ③ Shortest-path planning over this graph yields dense pixel-wise costs for a query view, forming a geometrically informed WayPixel costmap with fine-grained gradients toward the goal. ④ A learned controller conditioned on this dense costmap predicts a trajectory rollout that guides the robot toward the target.

***Abstract*— Visual navigation ability is strongly tied to its underlying representation of the world. Unlike classical 3D maps that require globally-consistent geometry, image- or object-relative topological graphs almost entirely do away with geometric understanding. But, this comes at the cost of navigation capability, often limiting it to merely teach-and-repeat. In this work, we propose a novel map representation in the form of *pixel-relative connectivity*, which is geometrically accurate but does not require global geometric consistency. Inspired by recent progress in 3D grounded image matching, we construct a map from an image sequence through inter-image connectivity based on pixel correspondences in the relative 3D coordinate systems of individual image pairs. We then use this pixel-level graph to perform global path planning by approximating and sparsifying intra-image pixel connectivity. Through this, we derive a "*WayPixel Costmap*" representation and train a controller conditioned on it to predict a trajectory rollout. We show that this dense pixel-level costmap based on relative geometry is a more accurate conditioning variable for control prediction than its image- and object-level counterparts. This enables a highly capable navigation system, as validated on four types of navigation tasks in the simulator and through real world demonstrations.**

I. INTRODUCTION

Visual navigation requires a structured representation of the surrounding environment that can support direct reasoning for core tasks such as localization, path planning, and control. Classical approaches rely heavily on geometric occupancy maps [1], where the world is represented in terms of free and occupied space. While such maps enable globally-consistent planning, they typically operate in a metric space and require highly-accurate global registration of 3D points and robot's poses. More recent alternatives propose image-relative topological graphs [2], [3], where nodes correspond to observed images, and edges encode traversability between them. Most of these metric and image-topological approaches plan paths in the form of 3D waypoints [4] and image goals [3] respectively. However, these waypoints decouple planning from control: the controller often only has access to a single specific subgoal in either geometric or image space, and that too without any notion of the quality of that subgoal.

In contrast, recent work has argued for object-relative representations [5], [6], [7], where the world is abstracted into semantically meaningful entities and the object-level path planning costs are directly communicated to the controller. As a result, the controller can learn to be robust to imperfections in planning. However, this benefit is achieved at the cost of reduced geometric understanding due to the object-level abstraction of the underlying map representation. In this paper, we bridge this gap by introducing a *geometrically precise yet relative* 3D map representation, formulated as dense **pixel-relative connectivity**, which enables relaying of *pixel-level* path planning costs to the controller. Unlike

The authors acknowledge the support provided by MeitY, Govt. of India, under the project "Capacity building for human resource development in Unmanned Aircraft System (Drone and related Technology)".

† Corresponding Author. * Equal Contribution. ‡ Equal Advising.
[1] Robotics Research Center, IIIT-Hyderabad, India
[2] University of Heidelberg [3] MBZUAI

conventional map abstractions that operate at the level of objects [6] or images [3], our representation retains pixel-level geometric structure while remaining anchored in relative rather than globally consistent coordinates. This is made possible by recent advances in 3D-grounded pairwise image matching [8], most notably MASt3R [9], whose predicted depth maps and point maps provide complementary sources of intra-image and inter-image pixel correspondences. Leveraging these predictions, we compose a dense connectivity structure across the relative 3D coordinate systems of consecutive image pairs in a traversal sequence, ultimately constructing a graph where each node corresponds to a pixel.

Within this pixel-level graph, edges encode two types of relationships: (1) inter-image correspondences, which form zero-cost edges that directly connect pixels across frames, and (2) intra-image edges, weighted by the 3D Euclidean distance between pixels derived from predicted depth. This formulation enables the use of classical graph search algorithms such as Dijkstra's algorithm to perform global path planning directly over dense pixel-level structure. To make this representation usable for downstream control, we transform the resulting pixel-level planning costs into a visual costmap that we term the *WayPixel Costmap*. This costmap serves as a fine-grained representation of navigation affordances, encoding subtle geometric cues (e.g., cost gradients on walls, ceilings, or sloped surfaces) that are typically lost in higher-level object- [6] or image-relative [3] abstractions.

We then couple this representation with a learning-based controller conditioned on the WayPixel Costmap and predict trajectory rollouts. A key property of this planning-control interface is that the controller can learn to be robust to inconsistencies in the planned path by exploiting local costmap gradients, which is more robust than its object-relative counterpart. This is illustrated in Figure 1: we compare pixel-and object-level costmaps, showing how our representation preserves fine-grained geometric structure that correctly leads the robot to the 'metal cabinet' goal, unlike the coarseness of object-level abstraction which results in an incorrect right turn.

While pixel-level representations offer fine-grained geometric fidelity, a key limitation of pixel-relative connectivity is its density. A naïve formulation would require a fully connected graph over millions of pixel nodes, which is computationally prohibitive. To make this tractable, we introduce three approximations: (a) intra-image nodes are limited to pixels with at least one inter-image correspondence, ensuring only geometrically grounded pixels are retained; (b) intra-image connectivity is enforced via a Euclidean Minimum Spanning Tree (MST) instead of full connectivity, reducing edge redundancy while preserving distances; and (c) unconnected pixels are selectively re-activated during live operation to bridge localized and connected regions. Together, these approximations reduce graph complexity while maintaining sufficient geometric information for navigation.

With this efficient graph, we construct dense pixel-level costmaps that encode *relative geometry* at high fidelity. These costmaps provide finer gradients and richer local structure than image- or object-level abstractions, yielding a more effective conditioning variable for control. By grounding planning in pixel-relative connectivity, our representation achieves a balance between computational feasibility and geometric precision, enabling more robust and accurate trajectory prediction.

Contributions. In summary, this paper makes the following key contributions:

- We propose **MASt3R-Nav**, a topological navigation pipeline based on a novel **pixel-relative representation** that leverages only *local* geometry through relative 3D connectivity of pixels;
- We propose a dense **WayPixel Costmap** representation as an interface between path planning and control, obtained through an efficient method for computing pixel-level path planning costs; and
- We propose a new learnt controller, **PixelReact**, conditioned on the WayPixel Costmaps, enabling it to exploit fine-grained cost gradients for robust trajectory rollout while mitigating the impact of planning errors.

We demonstrate that our pixel-relative navigation pipeline can offer both enhanced geometric understanding and improved robustness compared to image- or object-relative navigation paradigms, as validated in the simulator and through real world demonstrations.

II. RELATED WORK

Geometry-Rich 3D Maps: A longstanding line of research in visual navigation builds globally-consistent 3D maps that capture accurate geometry. Classical SLAM systems such as ORB-SLAM and LSD-SLAM [10], [11], [12] reconstruct metric maps, while extensions like SLAM++ [13], Quadric-SLAM [14], and Kimera [15] augment them with semantics. Scene graph formulations [16], [15] further encode spatial relations between objects and have been used for planning and navigation [17], [18]. More recently, methods like VGGT [19], MASt3R-SLAM [20] and MASt3R-Sfm [21] have demonstrated 3D reconstruction based on geometry-grounded learning for image matching [9], [8]. While these maps offer geometric precision, they are primarily designed for globally-registered 3D mapping.

Image-Relative Representations: To avoid the overhead of global 3D reconstruction, topological methods build maps where nodes correspond to images and edges encode visual or temporal similarity. Inspired by animal landmark navigation, SPTM [2] proposed such an image graph and trained a controller to reach subgoal *images*. Follow-up works demonstrated extensions to long-horizon tasks [22], language goals [23], and embodiment generalization [3]. Local control in this paradigm is obtained by comparing the current view with a chosen subgoal image, either through learning [24], [25], [26], or visual servoing [27], [28]. Despite their simplicity, image-relative approaches are tightly coupled to the viewpoints seen during mapping, and errors in subgoal selection can cascade through the controller.

Object-Relative Representations: An alternative line of work grounds navigation in semantically meaningful entities.

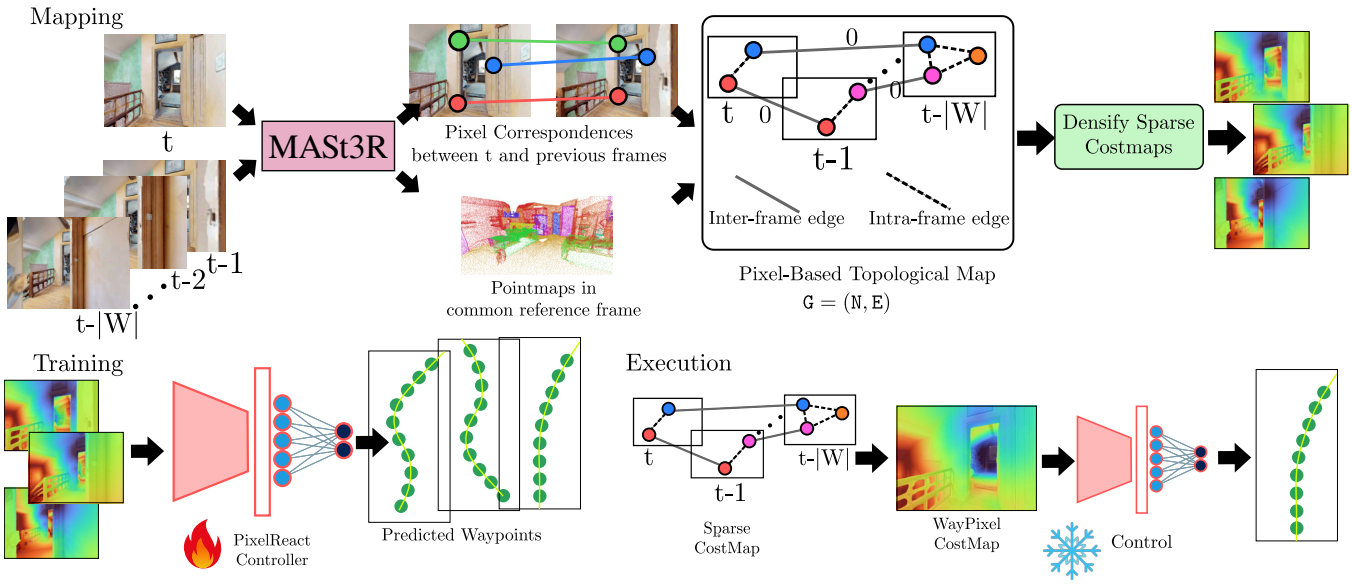

Fig. 2: **MASt3R-Nav Architecture.** *Mapping* involves constructing a pixel-level topological graph by linking correspondences across frames and encoding traversal costs using 3D geometry from MASt3R. *Execution* has the agent localize itself against the map and generate a fine-grained pixel costmap by matching current observations and propagating their costs to all pixels. *Planning* is performed by computing shortest paths through this costmap, yielding dense pixel-wise gradients towards the goal. Finally, a neural controller consumes the pixel costmap to predict waypoints, enabling a direct comparison with object-based baselines.

Recent methods define subgoals at the level of objects, pixels, or points visible in the current observation. RoboHop [5] and PixNav [29] respectively use objects and pixels as subgoals, while TANGO [7] uses 3D points with occupancy-based traversability estimation. ObjectReact [6] uses a *WayObject Costmap* representation based on image segments and their path lengths, and learns a controller conditioned on it. These approaches highlight the benefits of object-relative reasoning: subgoals are invariant to the robot's pose and can be specified via open-vocabulary queries [5], [4]. However, their conditioning variables for control remain coarse. RoboHop's controller is prone to collisions due to simplistic motion priors, PixNav's learned policies overfit to layout, and WayObject costmaps discard fine-grained geometric gradients, limiting control precision.

Pixel-Relative Connectivity: Our work takes a different route by moving from image- and object-relative abstractions to a pixel-level representation. Leveraging recent advances in 3D-grounded image matching [9], we construct a graph where nodes are pixels and edges capture relative 3D connectivity both within and across frames. This representation retains geometric fidelity locally without requiring global consistency. From this dense structure, we derive the *WayPixel Costmap*, a fine-grained cost representation that preserves subtle geometry (e.g., wall boundaries, slopes, occlusions) typically lost in image- or object-level abstractions. By conditioning a controller directly on WayPixel costmaps, our approach combines the task relevance of object-level reasoning with the geometric precision of metric maps. In contrast to WayObject costmaps [6], which rely on discrete entities, our pixel-relative formulation enables smoother gradients for control and greater robustness to planning errors.

III. APPROACH

Our pixel-relative navigation pipeline is designed to test the hypothesis that a controller conditioned on a fine-grained, pixel-level representation will outperform one conditioned on object- and image-level representations. The framework is structured into three distinct phases: mapping, execution, and training.

A. The MASt3R Backbone

MASt3R [9] is a 3D foundation model that takes as input a pair of images and outputs pixel-level correspondences between these images and dense point maps in the form of per-pixel (x, y, z) coordinates for both input images. We utilize both outputs of MASt3R in our pipeline. We denote $\mathtt{D}(p)$ as the point coordinate output by MASt3R corresponding to pixel p. We thus denote the 3D distance between the points corresponding to pixels p and q as $\mathtt{D}_{3D}(p,q) = ||\mathtt{D}(p) - \mathtt{D}(q)||$. Additionally, the important thing to note is that the point maps are output in a shared coordinate frame (that of the first image). Our motivation behind using MASt3R is that relative geometry (as provided by MASt3R) is effective for object-goal navigation and global geometry which can be hard to obtain accurately is not always required. In our pipeline, we use a frozen pre-trained MASt3R throughout the training and evaluation process.

B. Mapping Phase: Pixel-based Topological Map

Given a set of reference images from a prior traversal of the environment, we construct a pixel-level topological

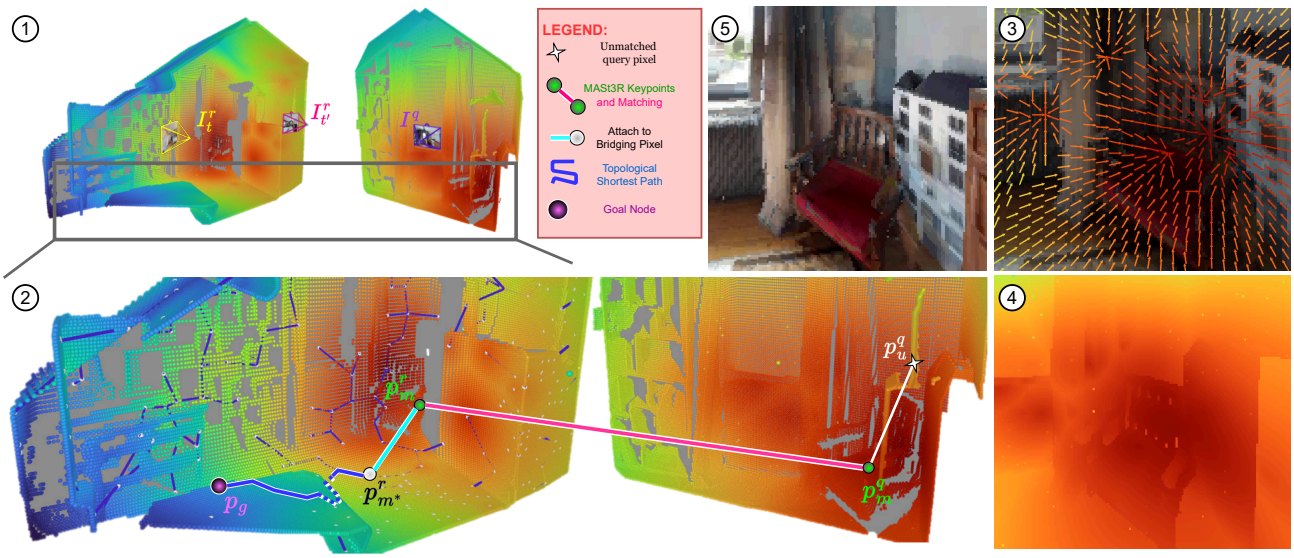

Fig. 3: **WayPixel Costmap generation.** Given the pixel-relative map representation and a query ①, we obtain pixel-level planning costs through a series of steps that form a path highlighted in white background ② from p_u^q to p_g through p_m^q, p_m^r and $p_{m^*}^r$. ③ We show the flow of cost gradients from each pixel to the its closest least-cost matched pixel and ④ the final dense WayPixel Costmap on which we condition our trained controller *PixelReact*. ⑤ shows the query RGB; the goal position is off-screen, toward the left of the scene.

map. Unlike prior approaches that rely on object segmentation [6], [5], our pipeline leverages MASt3R to establish correspondences between consecutive frames. Specifically, for the current frame I_t, we compute correspondences with a window $W = \{t-1, t-2, \ldots, t-|W|\}$ of previous frames. The resulting sets of pairwise pixel correspondences are then used to construct a pixel-based graph, denoted as the *Pixel-Based Topological Map*, formally defined as $\mathtt{G} = \{\mathtt{N}, \mathtt{E}\}$. $\mathtt{N}$ is the set of pixels that participate in correspondences, *i.e.* each matched pixel becomes a node $n \in \mathtt{N}$. The edge set $\mathtt{E}$ consists of two categories:

1) **Inter-frame edges:** $e = (p,q) \in \mathtt{E}$ connect a pixel p in the current frame $\mathtt{I}_t$ to its correspondence q in one of the previous frames. These edges are assigned cost 0 [30], as they denote the same 3D point observed across frames. Optionally, to prevent unbounded memory growth over long trajectories, we merge pixel node pairs corresponding to these edges such that the merged node is incident to any edge that was incident to the original two nodes.
2) **Intra-frame edges:** $e = (p_1, p_2) \in \mathtt{E}$ connect pairs of pixels within the same frame that each correspond to pixels in other frames. The cost of these edges is the 3D Euclidean distance $\mathtt{D}_{3D}(p_1, p_2)$ between the points corresponding to p_1 and p_2 in the MASt3R point cloud. These edges capture the cost of moving between two distinct 3D points within a frame.

More formally, if $\mathtt{C}(\cdot,\cdot)$ denotes the cost between two pixels, we have

$$\mathtt{C}(p,q) = 0 \qquad \text{if } (p,q) \text{ are matched, } p \in \mathtt{I}_t,\ q \notin \mathtt{I}_t, \tag{1}$$

$$\mathtt{C}(p_1,p_2) = \mathtt{D}_{3D}(p_1,p_2) \qquad \text{if } p_1, p_2 \in \mathtt{I}_k,\ k \in \mathtt{W} \cup \{t\} \tag{2}$$

Thus, $\mathtt{G}$ encodes costs between 3D points corresponding to a set of matched pixels. Different strategies can be adopted to determine the connectivity structure of the intra-frame edges (for example, fully connected graphs or minimum spanning trees). We evaluate some of these choices in the results section.

C. Execution Phase: Localization and Planning

Localization: During the execution phase, the agent must localize itself and plan a path to the goal. At each timestep, the agent captures an RGB image and uses the same dense matcher to find correspondences between its current query frame I_t^q and a submap $\mathtt{S}$. The submap for the current timestep is defined as a sequence of images centered at the localization index; this index is updated at every timestep as a global pointer to the submap image that yields the maximum number of matches with the query image.

Planning: The core of our planning approach is to define a cost for every pixel in the agent's current view. This is achieved through a multi-step process that leverages the pixel-level graph, dense feature matching, and relative 3D geometry. In the following, we explain the computation of pixel-level path planning costs for a given query image, as also illustrated in Figure 3.

a) Topological Path to the Goal: The foundation of our proposed costmap is the path from any pixel node p_j^r in the reference image r of the map graph $\mathtt{G}$ to the goal

pixel node p_g. This path is based on the node connectivity through both the intra-image and inter-image edges, and is calculated as a single-source shortest path in the weighted graph using Dijkstra's algorithm. The cost of this path is denoted as $\mathtt{C}(p^r_j, p_g)$. As the map graph and the goal is known apriori, we precompute these costs from every node to the goal node. An example of such a path is shown at the bottom left of Figure 3, depicted as the blue-colored path from $p^r_{m^*}$ (white pixel node from map image $I^r_{t'}$) to p_g (purple goal node from map image I^r_t).

b) Sparse Costmap via Dense Matching: As shown in Figure 3, a query pixel p^q_m (green node in the right point cloud) can get matched with a submap image pixel p^r_m (green node in the left point cloud), which is not a connected node in the graph $\mathtt{G}$. Thus, we first find a *bridging map pixel* $p^r_{m^*}$ (white node) in the submap image r that minimizes the sum of the local 3D distance and the node's known topological cost:

$$\mathtt{C}(p^r_{m^*}, p_g) = \min_{p^r_k \in \mathtt{N}^r} \left(\mathtt{D}_{\mathtt{3D}}(p^r_m, p^r_k) + \mathtt{C}(p^r_k, p_g)\right) \tag{3}$$

where $\mathtt{N}^r$ is the set of graph nodes belonging to the submap image r. We then use these costs to select a single best matching submap image r^* such that it leads the robot towards the goal. We achieve this by minimizing the median of the pixel-level costs per submap image:

$$r^* = \arg\min_{r \in \mathtt{S}} \operatorname*{median}_{m^*} \mathtt{C}(p^r_{m^*}, p_g) \tag{4}$$

Given the selected submap image (r^*) and the costs of its pixels that matched with the query image, we directly transfer the costs between the corresponding pixels, *i.e.*, $\mathtt{C}(p^q_m, p_g) = \mathtt{C}(p^{r^*}_{m^*}, p_g)$. By computing costs for all the matched query pixels, we obtain a *sparse costmap*, which is then converted into a dense representation as described below.

c) Dense Costmap via Cost Propagation: Similar to the object-level costmap representation of ObjectReact [6], we aim to obtain a *dense pixel-level* costmap as a more informative path planning signal for the learnt controller. We refer to this costmap as **WayPixel Costmap**, defined as an image array where each pixel represents its cost to the goal. Given the sparse costs of only the matched query pixels ($\mathtt{P}^q_m$), we propagate these costs to all the unmatched query pixels ($\mathtt{P}^q_u$). Leveraging the per-pixel 3D point coordinates provided by MASt3R, we can compute the local 3D Euclidean distance $\mathtt{D}_{3D}(p^q_u, p^q_m)$ between an unmatched pixel and all matched pixels in the current view. The cost for an unmatched pixel is then determined by the same cost function shown in Equation 3, this time using the matched pixels $\mathtt{P}^q_m$ as the *bridging* source of known costs. Concretely, the cost for an unmatched pixel is determined by finding a matched neighbor that minimizes the sum of the local 3D distance and the neighbor's known topological cost to the goal, as illustrated with the white colored path from p^q_u (white star) to p^q_m (green node) in Figure 3:

$$\mathtt{C}(p^q_u, p_g) = \min_{p^q_m \in \mathtt{P}^q_m} \left(\mathtt{D}_{\mathtt{3D}}(p^q_u, p^q_m) + \mathtt{C}(p^q_m, p_g)\right). \tag{5}$$

TABLE I: Comparison of object-level vs. pixel-level representations for image-goal navigation.

Mapper	Localizer	Controller	SPL	SSPL
Object-level Representation				
LGlue	LGlue	ObjectReact	51.51	58.59
MASt3R	LGlue	ObjectReact	45.45	53.21
LGlue	MASt3R	ObjectReact	51.50	60.85
MASt3R	MASt3R	ObjectReact	51.48	58.64
Pixel-level Representation				
MASt3R	MASt3R	ObjectReact	63.63	74.11
MASt3R	MASt3R	PixelReact	**81.77**	**84.36**

This process yields our dense WayPixel Costmap that provides a continuous gradient towards the goal for every pixel in the agent's field of view. Similar to ObjectReact [6], to create a representation suitable for a neural network, these scalar costs are then encoded into a higher-dimensional embedding using sinusoidal functions, analogous to positional encodings [31] in Transformers [32].

D. Training Phase: The PixelReact Controller

Finally, we propose a WayPixel Costmap conditioned learnt controller, **PixelReact**, which learns to predict a sequence of future waypoints from the dense pixel-level costs. To ensure a fair comparison and isolate the contribution of the novel representation, we adopt the neural network architecture and imitation learning procedure from the ObjectReact [6] baseline. The controller architecture consists of a convolutional encoder that processes the *Pixel Costmap* and a small MLP decoder that outputs local waypoints, that is, position and yaw relative to the current robot position. Specifically, this prediction takes the form of a trajectory rollout consisting of 10 future 2D waypoints in the robot's local bird's eye view (BEV) space. We train the model using the same data as ObjectReact [6] based on the trajectories from the Habitat simulator (HM3Dv0.2 dataset). The training objective is a regression loss, specifically the L2 distance, between the predicted waypoints $\hat{\mathbf{y}}$ and the ground-truth shortest-path waypoints $\mathbf{y}$. A rigorous, direct comparison is made against the ObjectReact controller, trained under identical conditions but with its original object-based costmap as input.

IV. EXPERIMENTAL SETUP

We evaluate navigation performance on the HM3D-IIN dataset, using its validation split [6], [7], which consists of 36 unique scenes, with one episode per scene. For every episode, a prior map is available in the form of an image sequence. In the offline mapping phase, we construct a pixel-relative 3D map using these RGB images and pointmaps obtained from Mast3R. During the online execution phase, the agent is initialized along the map trajectory in a way that it starts at least $5m$ (geodesic distance) or further away from the goal, ensuring that the path requires traversing multiple rooms and corridors on the same floor of the house. Throughout all our experiments, the agent is provided with ground-truth

TABLE II: **State-of-the-art** comparison of different control methods on four navigation tasks. Blue cells mark the best scores; Orange cells mark the second-best.

Method	Type	Train Data	Imitate		Alt Goal		Shortcut		Reverse		*Average*	
			SPL	SSPL	SPL	SSPL	SPL	SSPL	SPL	SSPL	SPL	SSPL
GNM [3]	Image-Relative	Real	78.79	82.95	8.70	15.44	15.38	31.74	3.33	6.11	26.55	34.56
GNM (HM3D) [6]	Image-Relative	HM3D	81.82	86.38	0.00	10.91	15.38	24.57	13.28	20.77	27.62	35.66
PixNav [29]	Object-Relative	HM3D	42.42	46.75	26.09	31.66	7.69	22.29	16.16	25.56	23.09	31.57
RoboHop [5]	Object-Relative	Zero Shot	57.56	64.99	30.43	38.23	30.77	40.87	9.98	16.92	32.19	40.25
ObjectReact [6]	Object-Relative	HM3D	60.60	68.51	21.74	26.68	23.08	39.64	30.00	42.01	33.36	44.71
MASt3R-Nav (**Ours**)	Pixel-Relative	HM3D	93.94	94.95	47.83	58.06	46.15	61.10	23.25	26.83	52.79	60.24

TABLE III: **Scalability study** comparing inter- and intra-image connectivity strategies.

Connectivity		Topomap Graph Stats			Computation Time (s)			Navigation	
EC	**NC**	**Num Nodes**	**Intra-Frame Edges**	**Disk (MB)**	**Intra**	**Edge-Weight**	**Dijkstra**	**SPL**	**SSPL**
Exhaustive	Sub10	24076	4660494	90.6	16.7	50.5	9.0	74.78	81.41
EMST	Sub10	24076	24011	**73.9**	8.7	**2.1**	**1.4**	**78.62**	**82.85**
Delaunay 3D	Sub10	24076	167970	75.3	**7.0**	3.7	1.7	78.56	**82.85**
EMST	None	191281	191215	78.4	98.2	4.3	2.3	62.06	73.16
Delaunay 3D	None	191281	1418359	85.9	25.7	24.2	8.5	66.28	74.92
ObjectReact [6]		1676	22633	4.76	–	–	0.011	60.60	68.51

localization in the form of the map image index that is closest to it's current 2D position.

Evaluation Metrics: We measure the controller's ability to reach the object goal within an episode. An episode is considered successful if the agent reaches within $1m$ of the goal in at most 300 steps, with an oracle stop condition. We report Success weighted by Path Length (SPL) [33] and Soft-SPL (SSPL) [34] metrics as in ObjectReact. SPL is a strict metric designed to measure both task completion and navigation efficiency, calculated by weighting a binary success indicator by the ratio of the shortest path distance to the maximum of the shortest and actually traversed path lengths. To account for near-misses, we also report SSPL, which replaces the binary success term with a soft value indicating the progress made by the agent toward the goal. This is particularly useful for episodes that are deemed failures (SPL$=0$) but have progressed toward the goal (SSPL>0). The results are averaged over 36 runs, using a fixed sensor height of $0.4m$ for both mapping and execution phases.

V. RESULTS

A. Object Relative vs. Pixel Relative

Table I ablates different modules of our navigation pipeline in comparison to ObjectReact [6] (first row). It can be observed that enhancing ObjectReact by simply replacing LightGlue (default for ObjectReact) with MASt3R as a matcher for mapping and localization (Rows 2-4) does not lead to any major performance gain. This can be attributed to a poor propagation of *dense* pixel-level matches to object-level matches, which subsequently affects the quality of Way-Object Costmap. On the other hand, our pixel-relative map representation generates fine-grained WayPixel Costmaps, which not only predict significantly better control using the original ObjectReact controller (penultimate row) but lead to a very high navigation performance with our PixelReact controller (last row).

B. State-of-the-art comparisons on multiple tasks

In Table II, we compare our proposed method against several existing methods on four challenging tasks as proposed in ObjectReact [6].

- **Imitate** requires the agent to follow it's prior trajectory, similar to a teach-and-repeat setup.
- **Alt Goal** involves navigating to a previously observed but unvisited goal, requiring the agent to take a new route.
- **Shortcut** modifies the mapping trajectory by including an additional stop at the Alt Goal, such that agent must find a shorter path to the final goal.
- **Reverse** tests the agent's ability to traverse the prior trajectory in the opposite direction.

Our method outperforms all the baselines on the Imitate, AltGoal and Shortcut tasks by large margins on both the metrics. On the Imitate task, we outperform the previous best method GNM [3] by absolute 10% on both SPL and SSPL. A similar trend holds against the GNM model trained on HM3D [6], using exactly the same training data as ours. On the AltGoal and Shortcut task, object-relative approaches substantially outperform image-relative baselines. On the other hand, our pixel-relative approach almost doubles the SPL and SSPL achieved by the object-relative methods. On the challenging reverse task, we achieve low performance, which is attributed to MASt3R's incorrect selection of a reference image for path planning. Although MASt3R like methods are demonstrate to work under large viewpoint shifts, we observed that it often found similar number of inliers for a true positive and a false positive match under the duress of low visual overlap. Nevertheless, our method

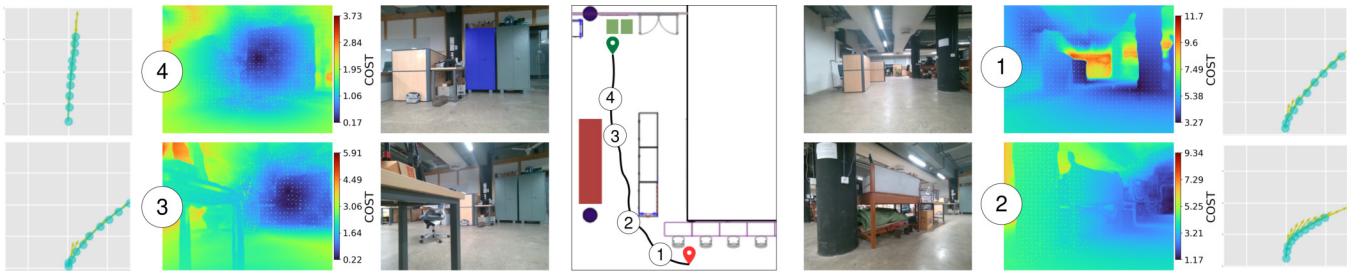

Fig. 4: **Real World Demonstration.** We show RGB observations, their WayPixel costmaps and the controller waypoints towards the goal object (shaded in blue in image 4) on four different locations in the robot trajectory.

achieves superior performance on average across all tasks, demonstrating its effectiveness in general.

C. Representation Efficiency for Pixel-Relative 3D Maps

Table III compares the extent of trade-off between representation efficiency and navigation success. We consider three types of intra-image connections: `Exhaustive`, *i.e.*, a complete intra-image graph with edges between all pixel nodes; `Delaunay 3D`, where edges are based on the circumscribing criteria of Delaunay Triangulation; and `EMST`, *i.e.*, Euclidean Minimum Spanning Tree, with minimum number of edges to maintain a connected graph. It can be observed that the number of edges drop from 4.6 million to merely 24,011 when replacing exhaustive connectivity with EMST. This significantly reduces the planning time while maintaining similar performance trends for navigation success. Since MASt3R is a dense image matching method, we randomly subsample pixel correspondences by a factor of 10, which proportionally reduces the number of pixel nodes that get created in any given image. In Table III, we also consider a baseline that uses all the pixel correspondences (2nd and 3rd last rows with 'None' subsampling); we consider both EMST and Delaunay 3D based intra-image edge connectivity for this baseline. It can be observed that using all the correspondences substantially increases the number of nodes and edges, resulting in a high computation time for planning while reducing navigation success metrics. We finally compare against the ObjectReact baseline for additional context, which is much more efficient but not as performant. Overall, this study highlights that point-level geometry with sparse connectivity suffices for robot navigation, and can potentially scale to even larger environments.

D. Scalability: Computational and Storage Footprint

We evaluate the computational and storage requirements of our offline mapping and planning pipeline as the environment scale increases, using a video captured across a three-floor, 30×36m office building [6]. To ensure a fair comparison, all evaluations, including baseline runtimes for ObjectReact, were conducted on a server with an AMD Ryzen 9 7950X CPU and an NVIDIA RTX A4000 (16GB) GPU. As shown in Table IV, both the total runtime and memory consumption scale roughly linearly with the expanding size (floors) of the environment. With over 80,000 pixel nodes, MASt3R-Nav's map generation and dense costmap computation executes in under five minutes (294.41s), requiring 540.18 MB of storage.

TABLE IV: Scalability of runtime and memory with increasing scene size. Graph complexity is denoted as I / V / E, representing the number of Images (I), Vertices/Nodes (V), and Edges (E).

	Scene Scale		
Metric	**One Floor**	**Two Floors**	**Three Floors**
I / V / E (Ours)	178 / 21,810 / 42,096	417 / 51,823 / 100,132	639 / 80,217 / 154,865
I / V / E (ObjectReact)	178 / 3,031 / 12,127	417 / 6,949 / 28,162	639 / 9,352 / 37,503
Runtime (s)			
Depth Estimation	18.29	43.21	66.89
Matching	54.13	130.39	202.15
Graph Construction	0.98	2.63	4.15
Planning	0.04	0.11	0.18
Dense Costmaps	2.22	5.41	8.68
Total (Ours)	**78.74**	**189.01**	**294.41**
Total (ObjectReact)	36.90	87.89	136.09
Memory (MB)			
Graph	1.20	2.93	4.53
MASt3R Pointmap	149.70	350.40	535.65
Total (Ours)	**150.80**	**353.33**	**540.18**

E. Real-World Demonstration

To validate the practical applicability and sim-to-real transfer capabilities of our proposed method, we deployed our navigation pipeline in real world. We used a P3DX mobile robot, equipped with a RealSense camera for RGB images. Figure 4 shows a topdown view at the center, showcasing the robot's successful trajectory from the start marker (red) to the goal marker (green). We also show the RGB images, their corresponding WayPixel Costmaps, and the trajectory rollout predicted using our MASt3R-Nav controller. It is clear that despite being trained exclusively on the HM3D simulated dataset, our navigation pipeline performs effectively during inference on a real-world mobile robot in an unseen environment.

VI. CONCLUSION

We presented a visual navigation pipeline, MASt3R-Nav, with several desirable characteristics: a robust perception module driven by advances in 3D grounded image matching; a novel pixel-relative geometric representation for mapping and global path planning; an intermediate WayPixel Costmap representation as dense and informative interface between planning and control; and a learnt controller, PixelReact,

conditioned on WayPixel Costmaps to predict a trajectory rollout. Our pixel-relative approach surpasses prior state-of-the-art on multiple challenging navigation tasks, outperforming both image-relative and object-relative methods. This establishes the efficacy of relative pixel-level geometry for navigation without the need for globally-consistent 3D reconstruction, pose estimation, sensor depth or traversability estimation. Future work can consider a hybrid representation approach that combines the benefits of object-level abstraction and pixel-level geometry to solve highly challenging tasks such as reverse path tracing.

REFERENCES

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 2002.

[2] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[3] 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)*. IEEE, 2023, pp. 7226–7233.

[4] 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.

[5] S. Garg, K. Rana, M. Hosseinzadeh, L. Mares, N. Suenderhauf, F. Dayoub, and I. Reid, "Robohop: Segment-based topological map representation for open-world visual navigation," in *2024 International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[6] 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.

[7] S. Podgorski, S. Garg, M. Hosseinzadeh, L. Mares, F. Dayoub, and I. Reid, "Tango: Traversablility-aware navigation with local metric control for topological goals," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[8] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[9] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3d with mast3r," in *European Conference on Computer Vision*. Springer, 2024, pp. 71–91.

[10] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[11] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[13] 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*, 2013, pp. 1352–1359.

[14] L. Nicholson, M. Milford, and N. Sünderhauf, "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam," *IEEE Robotics and Automation Letters*, 2019.

[15] 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*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[16] 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*, 2019, pp. 5664–5673.

[17] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone, "Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9272–9279.

[18] H. Yin, X. Xu, Z. Wu, J. Zhou, and J. Lu, "Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation," *Advances in neural information processing systems*, vol. 37, pp. 5285–5307, 2024.

[19] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[20] R. Murai, E. Dexheimer, and A. J. Davison, "Mast3r-slam: Real-time dense slam with 3d reconstruction priors," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 695–16 705.

[21] B. P. Duisterhof, L. Zust, P. Weinzaepfel, V. Leroy, Y. Cabon, and J. Revaud, "Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion," in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 1–10.

[22] D. Shah and S. Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints," in *Proceedings of Robotics: Science and Systems*, 2022.

[23] 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.

[24] Y. Li and J. Košecka, "Learning view and target invariant visual servoing for navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 658–664.

[25] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678.

[26] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2050–2053.

[27] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[28] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE transactions on robotics and automation*, vol. 18, no. 4, pp. 534–549, 2002.

[29] 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)*. IEEE, 2024, pp. 5228–5234.

[30] S. Garg, K. Rana, M. Hosseinzadeh, L. Mares, N. Sünderhauf, F. Dayoub, and I. Reid, "Robohop: Segment-based topological map representation for open-world visual navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4090–4097.

[31] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 464–468, arXiv:1803.02155.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008, arXiv:1706.03762.

[33] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[34] S. Datta, O. Maksymets, J. Hoffman, S. Lee, D. Batra, and D. Parikh, "Integrating egocentric localization for more realistic point-goal navigation agents," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 313–328.