

# IGL-Nav: Incremental 3D Gaussian Localization for Image-goal Navigation

Wenxuan Guo<sup>1\*</sup> Xiuwei Xu<sup>1\*</sup> Hang Yin<sup>1</sup> Ziwei Wang<sup>2</sup>

Jianjiang Feng<sup>1†</sup> Jie Zhou<sup>1</sup> Jiwen Lu<sup>1</sup>

<sup>1</sup>Tsinghua University

<sup>2</sup>Nanyang Technological University

{gwx22, xxw21, yinh23}@mails.tsinghua.edu.cn ziwei.wang@ntu.edu.sg

{jfeng, jzhou, lujiwen}@tsinghua.edu.cn

## Abstract

Visual navigation with an image as goal is a fundamental and challenging problem. Conventional methods either rely on end-to-end RL learning or modular-based policy with topological graph or BEV map as memory, which cannot fully model the geometric relationship between the explored 3D environment and the goal image. In order to efficiently and accurately localize the goal image in 3D space, we build our navigation system upon the renderable 3D gaussian (3DGS) representation. However, due to the computational intensity of 3DGS optimization and the large search space of 6-DoF camera pose, directly leveraging 3DGS for image localization during agent exploration process is prohibitively inefficient. To this end, we propose IGL-Nav, an Incremental 3D Gaussian Localization framework for efficient and 3D-aware image-goal navigation. Specifically, we incrementally update the scene representation as new images arrive with feed-forward monocular prediction. Then we coarsely localize the goal by leveraging the geometric information for discrete space matching, which can be equivalent to efficient 3D convolution. When the agent is close to the goal, we finally solve the fine target pose with optimization via differentiable rendering. The proposed IGL-Nav outperforms existing state-of-the-art methods by a large margin across diverse experimental configurations. It can also handle the more challenging free-view image-goal setting and be deployed on real-world robotic platform using a cellphone to capture goal image at arbitrary pose. Project page: <https://gwxuan.github.io/IGL-Nav/>.

## 1. Introduction

Image-goal navigation, which requires an agent initialized in unknown environment to navigate to the location and orientation specified by an image [39], is a fundamental problem in a wide range of robotic tasks. This task requires the



Figure 1. IGL-Nav effectively guides the agent to reach free-view image goal via incremental 3D gaussian localization.

agent to precisely understand spatial information, as well as to reason how to explore the scene with past observations, which is hard to learn with end-to-end RL [32] due to low sample efficiency and catastrophic forgetting.

Recent advances in visual navigation have witnessed significant progress in modular-based approaches [2, 3, 7, 14, 23, 34, 35], which establish an explicit memory to cache observed environmental information and derive navigation policies based on the memory representations. While these approaches demonstrate enhanced capabilities for long-horizon reasoning and temporal dependency modeling in object-goal navigation tasks, their extension to image-goal navigation remains challenging. Unlike object-goal scenarios that primarily rely on high-level semantic understanding, image-goal navigation necessitates the preservation and processing of low-level visual features, including fine-grained texture patterns and color distributions. Consequently, conventional representation paradigms of memory such as topological graphs prove insufficient for effectively encoding the requisite environmental information in image-goal settings. To address these limitations, RNR-Map [14] introduces a renderable neural radiance map representation. Drawing inspiration from NeRF [20], this rep-

\*Equal contribution. † Corresponding author.

resentation enables photorealistic image rendering from arbitrary camera viewpoints. The renderable nature ensures the preservation of crucial low-level visual features, which has demonstrated superior performance in image-goal navigation. However, since NeRF is an implicit field with high computational cost, RNR-Map has to maintain the renderable representation in a 2D BEV map for efficient and explicit memory management. This 2D projection inherently loses critical 3D structural information, forcing RNR-Map to impose strict constraints on goal image acquisition, specifically requiring horizontal camera angles to ensure alignment with its BEV map. This significantly reduces its applicability in real-world scenarios. Therefore, an efficient 3D-aware memory representation is still desirable for image-goal navigation.

In this paper, we propose to leverage 3D Gaussian Splatting (3DGS) [10] as the scene representation for image-goal navigation. The 3DGS representation demonstrates exceptional suitability for the task: (1) as an explicit representation, 3DGS can be easily initialized with the observed RGB-D image and be incrementally accumulated in 3D space; (2) it supports efficient differentiable rendering, which can be used to localize the camera pose of goal image with iterative optimization. Despite these compelling properties, adapting 3DGS representations for image-goal navigation presents significant challenges. While 3DGS achieves rendering speeds orders of magnitude faster than NeRF, their optimization process remains computationally prohibitive for real-time online inference required in navigation tasks. Furthermore, goal image localization within scene-level 3DGS maps becomes intractable due to the exponential search space complexity inherent in 6-DoF camera pose estimation. To this end, we propose IGL-Nav, an Incremental 3D Gaussian Localization framework that (1) progressively constructs 3DGS through feed-forward prediction, eliminating offline optimization; and (2) enables efficient hierarchical goal search by harnessing both geometric and photometric attributes of 3DGS through our novel coarse-to-fine localization strategy. Extensive experiments on various datasets in Habitat simulator show our IGL-Nav significantly outperforms previous state-of-the-art image-goal navigation methods. Moreover, benefit from our explicit 3D representation, IGL-Nav is also able to handle the more practical free-view image-goal setting, where there is no assumption on both camera intrinsics and extrinsics of the goal image. We further deploy our method on real-world robot, where a casually taken photo from a cellphone can be used as goal to guide the agent navigating to specified location in complicated and large-scale environments.

## 2. Related Work

**3D Gaussian Splatting.** 3DGS [10] has emerged as a powerful technique for 3D scene representation. It repre-

sents a scene as a dense set of points with gaussian embedding and leverages efficient rasterization techniques for high-fidelity, real-time rendering. Recently, feed-forward 3DGS models [4–6] have been proposed, primarily to address the issue of sparse-view scene reconstruction. Unlike traditional methods that iteratively optimize 3DGS parameters, feed-forward 3DGS predicts the gaussian distribution through a network, significantly improving modeling efficiency. In embodied AI, 3DGS has been applied to manipulation tasks, with dynamic 3DGS frameworks and gaussian world models used to model and predict robotic actions [19]. And systems like Gaussian-Grasper [38] leverage RGB-D inputs for language-guided grasping. 3DGS also helps bridge the gap between simulated and real-world environments for generalizing learned behaviors. Techniques such as Robo-GS [18] and SplatSim [22] improve Sim-to-Real transfer by leveraging efficient representation of 3DGS. The incremental 3DGS scene representation used in our IGL-Nav also follows the paradigm of feed-forward 3DGS, making it suitable for scene modeling based on online inputs in navigation tasks.

**Image-goal Navigation.** Image-goal navigation involves the agent navigating to the location where the goal image is captured [39], requiring precise alignment in both position and orientation. To address this challenge, researchers have employed various strategies. Some focus on optimizing reinforcement learning (RL) policies [1, 32, 33] that directly map observations to actions. Others concentrate on constructing detailed maps [3, 11, 14, 24], or on developing carefully crafted matching algorithms [29]. However, image-goal navigation requires that the query image be captured by the agent’s camera, which limits the camera’s intrinsic parameters, height, and the fact that it can only rotate around the Z-axis. Considering these constraints in practical applications, we propose free-view image-goal navigation, where the target image can be captured by any camera at free 3D position and orientation. In addition, instance image-goal navigation [12] is a similar task, where the target image focuses on specific categories of objects within the scene. In instance image-goal task, GaussNav [16] also uses a 3DGS-based scene representation. However, GaussNav requires first completing the exploration of the entire building to optimize the 3DGS representation, and then render images at multi poses for comparison with the target image. This approach limits efficiency in practical applications. In contrast, our IGL-Nav simultaneously performs exploration, incremental modeling, and target localization, and it incorporates a coarse-to-fine localization strategy, making full use of the 3DGS representation.

## 3. Approach

In this section, we first describe our task definition. Next, we explain several core modules of IGL-Nav, including in-

cremental scene representation with 3DGS, and coarse-to-fine target localization. Finally, we detail the overall navigation pipeline.

### 3.1. Problem Statement

We study the problem of free-view image-goal navigation, which is a more challenging and practical setting. In this task, a mobile agent is instructed with navigating to a specified location depicted by an image  $\mathbf{I}_g$ , taken by camera A with pose  $\mathbf{T}_g$ . The agent is equipped with camera B. It receives posed RGB-D video stream  $\{\mathbf{I}_t, \mathbf{D}_t, \mathbf{T}_t\}_{t=1}^T$  and is required to execute an action  $a \in \mathcal{A}$  at each time it receiving a new RGB-D observation.  $\mathbf{I}_t, \mathbf{D}_t, \mathbf{T}_t$  refer to RGB image, depth image and camera pose at time instant  $t$ .  $\mathcal{A}$  is the set of actions, which consists of `move_forward`, `turn_left`, `turn_right` and `stop`. The task is considered successfully completed if the agent terminates within a horizontal neighborhood of the target pose, satisfying  $\|\mathcal{P}(\mathbf{T}_{final}) - \mathcal{P}(\mathbf{T}_g)\|_2 < \epsilon$  within a maximum of  $T$  navigation steps. Here  $\mathcal{P}$  refers to 3D-to-BEV projection.

**Comparison with Relevant Tasks.** In the free-view image-goal navigation, there is no assumption on the correlation between camera A and B. For example, in real application scenarios, A can be a cellphone, and B is a RGB-D camera with totally different camera intrinsics. Previous image-goal setting [7, 14] can be regarded as a special case of our task where  $A \equiv B$  and  $\mathbf{T}_g$  is restricted to lie within camera B's achievable pose space. Instance-image-goal navigation [13] also aims to decouple camera A and B. However, this setting requires that there must be an instance located at image center, and only 6 categories of instances are supported. These limitations fundamentally constrain the system's operational flexibility and real-world deployment potential. In this paper, we conduct experiments on both conventional and free-view image-goal settings for a comprehensive evaluation of different approaches.

### 3.2. Incremental Scene Representation

We adopt 3DGS as our scene representation due to its explicit nature and efficient rendering capability. However, the original 3DGS are obtained through offline optimization on image set and thus hard to be applied in real-time tasks. Recent feed-forward methods [4–6] abandon optimization and directly predict pixel-aligned 3DGS parameters, but they still rely on multi-view images to reconstruct geometric information of the scene. In visual navigation, the agent needs to incrementally build scene representation along with its exploration, so the 3DGS should be generated in real-time and update as new images arrive. To accommodate streaming video input while effectively leveraging camera pose and depth priors, we present the first feed-forward 3DGS reconstruction model for monocular RGB-D sequences, which supports real-time 3DGS reconstruction

and incremental accumulation.

**Gaussian Parameters Prediction.** At time step  $t$ , the agent receives new RGB-D observations  $\mathbf{I}_t \in \mathbb{R}^{H \times W \times 3}$  and  $\mathbf{D}_t \in \mathbb{R}^{H \times W \times 1}$ . Our incremental reconstruction model is essentially a mapping  $f_\theta$  from observations to 3DGS parameters, including position  $\boldsymbol{\mu}_k$ , opacity  $\alpha_k$ , covariance  $\Sigma_k$  and spherical harmonics  $\mathbf{c}_k$ :

$$f_\theta : (\mathbf{I}_t, \mathbf{D}_t) \mapsto \{(\boldsymbol{\mu}_k, \alpha_k, \Sigma_k, \mathbf{c}_k)\}_{k=1}^{H \times W} \quad (1)$$

The 3DGS parameters are predicted in a pixel-aligned manner, thus an observation input of size  $H \times W$  corresponds to an output of  $H \times W$  gaussians.

The feed-forward model  $f_\theta$  is shown in Figure 2. We first concatenate the normalized RGB and depth images, and then extract dense monocular scene embedding  $\mathbf{E}'_t$  with a UNet-based encoder  $\mathcal{E}$ . Then 3DGS parameters are regressed through a gaussian head  $\mathcal{H}$ , composed of a few CNN and linear layers. This process can be expressed as:

$$\Delta \mathbf{C}_{2D}, \Delta \mathbf{D}, \alpha, \Sigma, \mathbf{c} = \mathcal{H}(\mathbf{E}'), \quad \mathbf{E}' = \mathcal{E}(\mathbf{I}, \mathbf{D}) \quad (2)$$

where  $\Delta \mathbf{C}_{2D}$  and  $\Delta \mathbf{D}$  are residuals of image coordinates and depth. We omit subscript  $t$  for simplicity. Using the camera intrinsic matrix  $\mathbf{M}$ , pose  $\mathbf{T}_t$  and inverse projection  $\text{Proj}^{-1}$ , we can compute the 3DGS positions as:

$$\boldsymbol{\mu} = \text{Proj}^{-1}(\mathbf{C}_{2D} + \Delta \mathbf{C}_{2D}, \mathbf{D} + \Delta \mathbf{D} | \mathbf{M}, \mathbf{T}_t) \quad (3)$$

We also lift  $\mathbf{E}'$  from 2D to the corresponding 3D positions. Finally, the 3DGS scene representation  $\mathbf{G}$  and the corresponding 3D embedding  $\mathbf{E}$  can be updated as:  $\mathbf{G}_t = \mathbf{G}_{t-1} \cup (\boldsymbol{\mu}_t, \alpha_t, \Sigma_t, \mathbf{c}_t)$  and  $\mathbf{E}_t = \mathbf{E}_{t-1} \cup \mathbf{E}'_t$ . When the number of 3DGS in the scene is large, we prune  $\mathbf{G}_t$  and  $\mathbf{E}_t$  based on opacity and 3DGS density to reduce memory footprint. Additionally, we can use  $\mathcal{E}$  to extract the 3D embedding  $\mathbf{E}_g$  of the target image  $\mathbf{I}_g$ . If depth and camera intrinsics are unavailable for  $\mathbf{I}_g$ , we simply use a monocular depth estimator [21] to predict them.

**Training and Loss.** Our feed-forward model can be trained using passive offline RGB-D video streams. We randomly sample training episodes from navigation training set. In each episode,  $K$  frames are randomly selected to predict 3DGS parameters, and images from other viewpoints are rendered for loss computation. The training loss is a linear combination of L-2 and LPIPS [37] losses.

### 3.3. Coarse-to-fine Localization

Since the target image is captured by an arbitrary camera at any pose (6-DoF), the search space of the target is extremely large. To perform efficient and accurate visual navigation, we design a coarse-to-fine target localization strategy. Coarse localization leverages the incremental scene embedding  $\mathbf{E}_t$  to predict the approximate target location in real-time during exploration. Once the agent is close to the target, fine localization is employed to accurately determine the accurate target position and guide the agent to reach it.

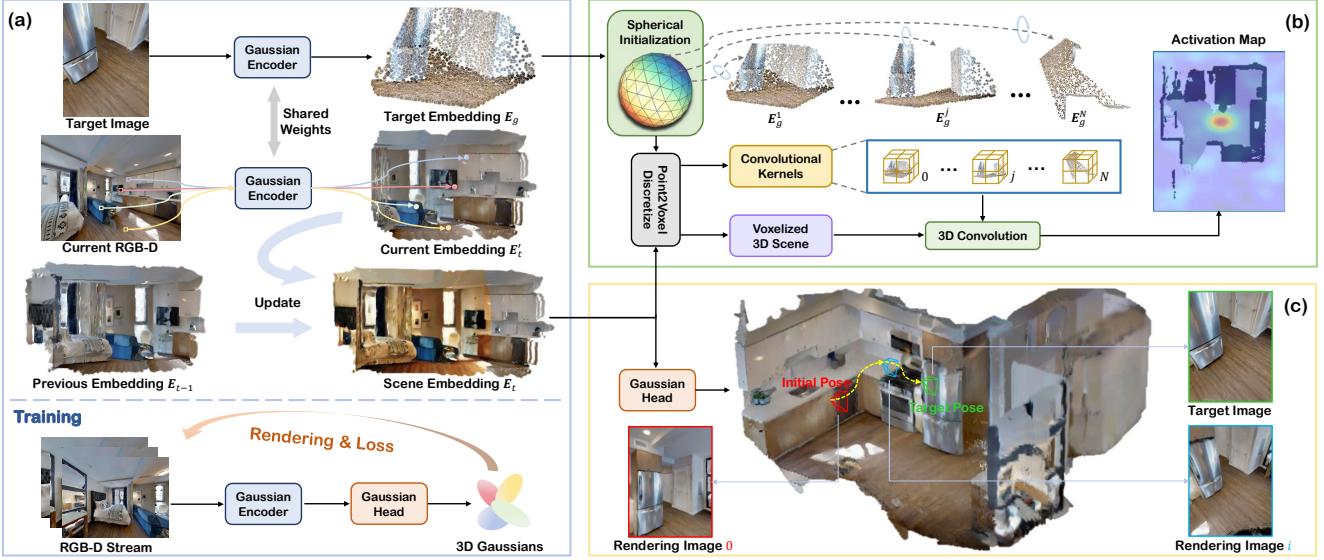


Figure 2. Illustration of IGL-Nav. (a) We maintain an incremental 3DGS scene representation with feed-forward prediction. (b) The coarse target localization is modeled as a 5-dimension matching problem, which is efficiently implemented by leveraging the target embedding as 3D convolutional kernel. (c) Fine target localization via differentiable 3DGS rendering and matching-constrained optimization.

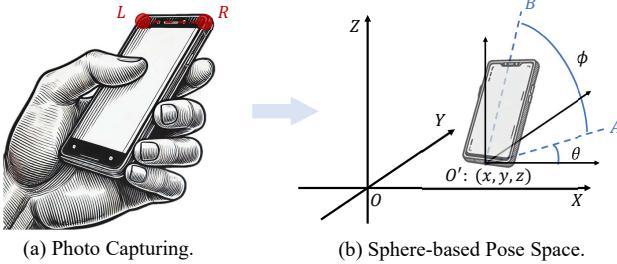


Figure 3. Modeling of the camera pose space. (a) Line  $LR$  is almost always parallel to the ground. (b) Line  $AO'$  is parallel to Plane  $XOY$ . Plane  $AO'B$  is perpendicular to Plane  $XOY$ .

### 3.3.1. Coarse Target Localization

Although camera A can capture the target image at an arbitrary pose, we observe that the top frame of the camera is almost always parallel to the ground when taking a photo, as shown in Figure 3. Therefore, we can represent the actual camera rotation with  $(\theta, \phi)$ , which denotes a rotation around the X-axis by  $\theta$  degrees, followed by a rotation towards the Z-axis by  $\phi$  degrees. Based on this observation, we define a sphere-based space  $\mathcal{S} : \{(x, y, z, \theta, \phi)\}$  to represent camera pose. Here  $(x, y, z)$  represents the position of camera A and  $(\theta, \phi)$  refers to A's rotation. We can thus represent the target pose  $T_g$  as  $(x_g, y_g, z_g, \theta_g, \phi_g)$ . The 3D embedding of the target  $E_g$  is initialized at the origin of the sphere-based space  $\mathcal{S}$  and should be aligned with the scene embedding  $E_t$  under translation  $(x_g, y_g, z_g)$  and rotation  $(\theta_g, \phi_g)$ , which are unknown.

To efficiently search the target camera pose in the five-dimensional space, we discretize  $\mathcal{S}$  to reduce the search

space. For  $(x, y, z)$ , the 3D space is voxelized into grids  $\{(x_i, y_i, z_i) \mid x_i = \lfloor \frac{x}{v} \rfloor, y_i = \lfloor \frac{y}{v} \rfloor, z_i = \lfloor \frac{z}{v} \rfloor\}$ , where  $v$  is voxel size. For  $(\theta, \phi)$ , we discretize the spherical surface into  $N$  vertices of a hierarchical mesh via  $\gamma$ -level subdivision of a regular icosahedron, as shown in Figure 2. In this way, we can rotate  $E_g$  according to the discretized sphere to obtain  $N$  3D embeddings  $\{E_g^1, \dots, E_g^N\}$ . By translating these embeddings to the discretized voxel grids and computing the extent of alignment between the translated embedding and  $E_t$ , the coarse target pose can be determined by:

$$\underset{i,k}{\text{maximize}} \quad \mathcal{A}(E_t, \mathcal{T}(E_g^k, (x_i, y_i, z_i))) \quad (4)$$

where  $\mathcal{A}$  computes the extent of alignment between two sets of 3D features.  $\mathcal{T}$  stands for translation operation.  $i$  and  $k$  are used to query the corresponding  $(x, y, z, \theta, \phi)$ .

However, the above operation is still hard to achieve real-time inference. We need to traverse all voxel grids and compare the translated 3D embedding with  $E_t$ . Assume there are  $V$  grids at all, then  $V \times N$  times comparisons should be performed. Moreover, during each comparison, we should compute the geometric similarity between two 3D pointclouds as well as their feature similarity, which is especially time-consuming and hard to be accelerated on GPUs. To solve this problem, we propose to further discretize the 3D embeddings  $E_t$  and  $E_g$ . For  $E_t$ , we can simply quantize the pointclouds into voxels, where voxel features are obtained by taking average of the pointcloud features inside each voxel. For  $\{E_g^1, \dots, E_g^N\}$ , we uniformly quantize them into  $L \times L \times L$  voxels. Note that although  $\{E_g^1, \dots, E_g^N\}$  are different pointclouds, they will share the

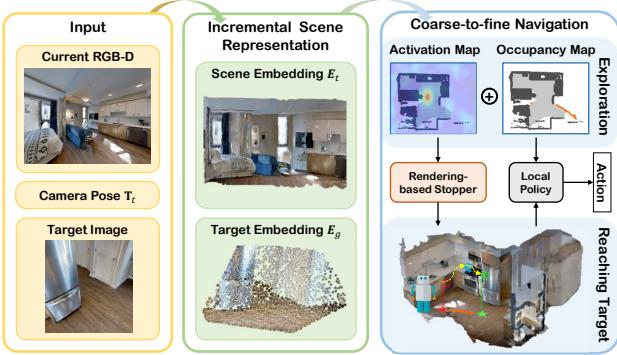


Figure 4. Navigation pipeline of IGL-Nav.

same shape after voxelization, which forms a 3D convolutional kernel  $\mathbf{K} \in \mathbb{R}^{L \times L \times L \times C_{in} \times C_{out}}$ . Here  $C_{in}$  refers to the output channel of  $\mathcal{E}$ ,  $C_{out}$  equals to the number of kernels  $N$ . Therefore, Eq (4) can be rewritten as:

$$\operatorname{argmax}_{x,y,z,k} \mathcal{C}(f_1(\mathcal{V}(\mathbf{E}_t)), f_2(\mathbf{K}))[x][y][z][k] \quad (5)$$

where  $\mathcal{C}$  means 3D convolution operation,  $\mathcal{V}$  quantizes scene embedding  $\mathbf{E}_t$  into  $X \times Y \times Z$  voxels. We use two MLP  $f_1 / f_2$  with input channel  $C_{in}$  and output channel  $C'$  to project scene embedding and convolutional kernel to a learnable feature space before convolution, which further aligns the embedding space of  $\mathbf{E}_t$  and  $\mathbf{E}_g$ . The activation map after 3D convolution is of shape  $X \times Y \times Z \times N$ , from which we query index of the maximum value and thus obtain a coarse localization of the target pose. To further improve computational efficiency, we use pillar-based voxelization [15, 36].

**Training and Loss.** Similar to the scene representation training, we train the coarse localization module using offline passive video streams. In each training segment, we randomly select a position and capture target images with arbitrary intrinsic parameters and orientations. We use focal loss [17] to supervise the activation map after 3D convolution. Additionally, we apply cross-entropy loss to supervise the outputs nearby target pose in the activation map.

### 3.3.2. Fine Target Localization

Our fine localization method aims to accurately determine the target's 6-DoF pose once the agent is close to the target region. It leverages the differentiable rendering ability of 3DGS to reach target pose via iterative optimization.

**Rendering-based Stopper.** First, we use a rendering-based stopper to determine if the agent is close to the target. Since the intrinsics of camera A and B may differ significantly, directly comparing the current observation with the target image with feature matching is difficult. Thanks to the real-time rendering capability of 3DGS  $G_t$ , we can render an image at camera B's current viewpoint with the same intrinsic parameters as camera A. We use a local feature matching method, LoFTR [27], to predict matching pairs

$(\mathbf{x}_g, \mathbf{x}_t)$  between the target image and the rendered image. Here  $(\mathbf{x}_g, \mathbf{x}_t)$  is the coordinate set of matched pixels. If the number of matching pairs exceeds a threshold  $\tau$ , it is considered that  $I_g$  appears in agent's field of view.

**Matching-constrained Optimization.** Via differentiable rendering, we can optimize the current camera pose with photometric loss between the rendered image and  $I_g$ , and between rendered depth and  $D_g$  ( $D_g / M_g$  are the depth / intrinsics of  $I_g$ , which are estimated by [21] if not available), as done in [9, 28]. Although this is an intuitive way to solve  $\mathbf{T}_g$ , we empirically find it leads to unsatisfactory performance in our case where the quality of  $G_t$  may degrade due to incremental accumulation without optimization. Fortunately, we observe the pixels that are successfully matched are of high quality. In order to overcome the imperfect details in the rendering results, we propose to only focus on the matching pairs in 3D space for accurate camera pose optimization. The problem can be formulated as:

$$\hat{\mathbf{T}} = \operatorname{argmin}_{\mathbf{T} \in SE(3)} \mathcal{L}(\mathbf{T} | I_g, D_g, M_g, G_t) \quad (6)$$

We iteratively optimize the pose  $\mathbf{T}$  to minimize the geometric discrepancy between rendering results and target image. At each iteration of optimization, we leverage current  $\mathbf{T}$  for rendering and obtain the matched points in Euclidean space:

$$(\mathbf{x}_g, \mathbf{x}), (\mathbf{d}_g, \mathbf{d}) = \mathcal{M}(I_g, D_g, \mathcal{R}(G_t | M_g, \mathbf{T})) \quad (7)$$

$$(\mathbf{X}_g, \mathbf{X}) = \operatorname{Proj}^{-1}((\mathbf{x}_g, \mathbf{x}), (\mathbf{d}_g, \mathbf{d}) | M_g) \quad (8)$$

where  $\mathcal{R}$  is differentiable rendering of color and depth. The matching and querying operation  $\mathcal{M}$  first adopts LoFTR to get matching pair  $(\mathbf{x}_g, \mathbf{x})$  between  $I_g$  and the rendered RGB image, and then queries the corresponding depth value  $(\mathbf{d}_g, \mathbf{d})$  from  $D_g$  and the rendered depth respectively. Then we formulate the optimization loss as:

$$\mathcal{L} = \frac{1}{Q} \sum_{i=0}^{Q-1} (|\mathbf{X}_g^i - \mathbf{X}^i|_2) \quad (9)$$

where  $Q$  is the number of matching pairs. Note that LoFTR predicts  $(\mathbf{x}_g, \mathbf{x})$  in a differentiable way, so gradient can be backpropagated through both the rendered color and depth images. In this way, we effectively align  $\mathbf{T}$  and  $\mathbf{T}_g$  by focusing on the most confident rendering results.

## 3.4. Navigation

We divide the navigation process into two stages: exploration based on coarse localization and target reaching based on fine localization. Figure 4 illustrates the workflow of IGL-Nav. We will describe each stage in this section.

**Exploration for Target Discovery.** When the agent is initialized in a new environment, its observations of the scene are insufficient. Therefore, we combine coarse target localization with frontier-based exploration to explore

Table 1. Image-goal Navigation Results. SR: Success Rate, SPL: Success weighted by Path Length. The best result in each column is **bold**, and the second best is underlined.

Method	Straight								Curved							
	Easy		Medium		Hard		Overall		Easy		Medium		Hard		Overall	
	SR	SPL														
DDPPO [30]	43.2	38.5	36.4	34.8	7.4	7.2	29.0	26.8	22.2	16.5	20.7	18.5	4.2	3.7	15.7	12.9
NRNS [7]	64.1	55.4	47.9	39.5	25.2	18.1	45.7	37.7	27.3	10.6	23.1	10.4	10.5	5.6	20.3	8.8
ZSEL [1]	-	-	-	-	-	-	-	-	41.0	28.2	27.3	13.9	18.6	9.3	25.9	17.6
OVRL [33]	53.6	34.7	48.6	33.3	32.5	21.9	44.9	30.0	53.6	31.8	47.6	30.2	35.6	22.0	45.6	28.0
NRNS + SLING [29]	85.3	74.4	66.8	49.3	41.1	28.8	64.4	50.8	58.6	16.1	47.6	16.8	24.9	10.1	43.7	14.3
OVRL + SLING [29]	71.2	54.1	60.3	44.4	43.0	29.1	58.2	42.5	68.4	47.0	57.7	39.8	40.2	25.5	55.4	37.4
RNR-Map [14]	76.4	55.3	<u>73.6</u>	46.1	<u>54.6</u>	30.2	<u>68.2</u>	43.9	<u>75.3</u>	<u>52.5</u>	<u>70.9</u>	<u>42.3</u>	<u>51.0</u>	<u>27.4</u>	<u>65.7</u>	<u>40.8</u>
FeudalNav [8]	82.6	<u>75.0</u>	71.0	<u>57.4</u>	49.0	<u>34.2</u>	67.5	<u>55.5</u>	72.5	51.3	64.4	40.7	43.7	25.3	60.2	39.1
IGL-Nav (Ours)	<b>87.9</b>	<b>82.5</b>	<b>80.8</b>	<b>69.0</b>	<b>61.7</b>	<b>40.9</b>	<b>76.8</b>	<b>64.1</b>	<b>82.8</b>	<b>77.7</b>	<b>80.7</b>	<b>70.0</b>	<b>57.0</b>	<b>39.6</b>	<b>73.5</b>	<b>62.4</b>

the scene and discover potential targets. Based on the posed RGB-D inputs, we maintain an online occupancy map to indicate explored, unexplored and occupied area in BEV, where the frontiers of explored area can be computed. At each time step, we select the nearest frontier to the agent and generate binary scores  $S_f$  on the BEV map, where points on the selected frontier are set to 1, others are set to 0. We then project the activation map obtained in our coarse target localization module to BEV to get  $S_a$ . We first filter the activation map  $S_a$  by a threshold  $\sigma_a$ , setting scores below the threshold to zero. The agent then prioritizes exploring the location with the highest value in  $S_a$ . If all values in  $S_a$  are zero, the agent selects the nearest location where the frontier score map  $S_f$  equals 1. We adopt Fast Marching Method [26] (FMM) for path planning and action generation given the to-be-explored location.

**Reaching Target.** During exploration, the agent gradually approaches the target. We use the rendering-based stopper to determine if the target appears in agent’s field of view. Once the target is detected, we switch to fine localization to compute the precise target pose. The XY coordinates of the computed pose is set to be destination, for which we apply FMM again for navigation.

## 4. Experiment

In this section, we first describe our experimental setting. Then we compare IGL-Nav with state-of-the-art image-goal navigation methods. Finally we conduct in-depth module-based analysis on our framework and further provide real-world deployment results.

### 4.1. Experimental Setup

We conduct experiments on image-goal navigation and the more challenging free-view image-goal navigation tasks.

**Datasets and Benchmarks.** For image-goal navigation, we follow the public Gibson [31] image-goal navigation dataset within the Habitat simulator [25] introduced by NRNS [7]. The Gibson dataset includes 72 houses for training and 14 for validation. The NRNS dataset contains two

path types (straight and curved), each with three difficulty levels (easy, medium, hard). For free-view image-goal navigation as introduced in Sec. 3.1, we collect a large amount of data with Gibson for validation. Given the significant impact of the camera’s field of view (FOV) on scene matching, we categorize our dataset into two FOV-based groups ( $50^\circ \sim 75^\circ$  and  $75^\circ \sim 100^\circ$ ), which can be intuitively understood as portrait and landscape orientations. Each category further includes three difficulty levels based on distance. Additionally, compared to the NRNS dataset, our free-view image-goal navigation dataset features target images captured from arbitrary angles and heights. Each of the six subsets contains 500 randomly sampled episodes.

**Compared Methods.** We compare IGL-Nav with existing state-of-the-art image-goal navigation methods [1, 7, 8, 14, 29, 30, 33]. For image-goal setting, we report results from the respective papers. For the proposed free-view image-goal setting, we evaluate open-sourced methods on this benchmark and compare with them. Since some methods [7, 29, 30, 33] only release test code, we perform zero-shot transfer to apply them to the new setting without re-training. We also report the zero-shot performance of IGL-Nav for fair comparison. For methods [7, 30] that provide training scripts, we train them on the free-view image-goal navigation data for comparison.

### 4.2. Comparison with State-of-the-art

We compare with state-of-the-art image-goal navigation methods on the two benchmarks described above. Table 1 demonstrates the results on image-goal navigation task. IGL-Nav establishes new state-of-the-art performance and outperforms previous methods by a large margin on all metrics, which validates the effectiveness of 3D gaussian representation and the proposed coarse-to-fine target localization strategy for image-goal navigation.

The results on free-view image-goal navigation task is shown in Table 2. As this task is much more challenging than conventional image-goal setting, we observe a significant performance drop on each metric. When directly

Table 2. Free-view Image-goal Navigation Results. SR: Success Rate, SPL: Success weighted by Path Length.

Method	Narrow FOV ( $50^\circ \sim 75^\circ$ )								Wide FOV ( $75^\circ \sim 100^\circ$ )							
	Easy		Medium		Hard		Overall		Easy		Medium		Hard		Overall	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL
<b><i>Zero-shot Transfer (Training on Image-goal Navigation Data)</i></b>																
DDPPO [30]	15.8	10.5	9.6	7.2	5.4	3.1	10.3	6.9	20.2	16.5	16.6	12.5	9.8	5.7	15.5	11.6
NRNS [7]	19.8	10.6	15.8	9.0	7.8	4.0	14.5	7.9	28.4	16.6	21.2	14.5	10.6	5.9	20.1	12.3
OVRL [33]	23.8	16.6	19.2	10.5	8.2	6.9	17.1	11.3	27.6	19.2	22.8	12.6	14.8	8.6	21.7	13.5
NRNS + SLING [29]	32.8	15.3	23.6	13.2	9.8	5.6	22.1	11.4	38.6	19.1	32.6	18.5	17.2	8.3	29.5	15.3
OVRL + SLING [29]	28.2	20.1	23.2	18.7	11.8	7.1	21.1	15.3	36.4	25.9	31.6	18.5	15.2	7.6	27.7	17.3
IGL-Nav (Ours)	<b>53.2</b>	<b>45.1</b>	<b>47.8</b>	<b>40.5</b>	<b>28.2</b>	<b>22.0</b>	<b>43.1</b>	<b>35.9</b>	<b>56.2</b>	<b>48.3</b>	<b>55.2</b>	<b>46.1</b>	<b>30.8</b>	<b>23.9</b>	<b>47.4</b>	<b>39.4</b>
<b><i>Supervised (Training on Free-view Image-goal Navigation Data)</i></b>																
BC + GRU	13.2	6.8	10.4	8.8	6.6	4.9	10.1	6.8	22.0	14.4	16.0	11.4	9.2	6.9	15.7	10.9
BC + Metric Map	22.8	15.9	20.6	15.6	7.4	5.2	16.9	12.2	25.4	19.5	22.8	18.5	4.8	3.5	17.7	13.8
DDPPO [30]	19.4	11.3	16.4	10.4	9.6	6.0	15.1	9.2	26.8	17.8	19.0	12.4	15.6	9.8	20.5	13.3
NRNS [7]	30.8	24.4	27.8	24.5	11.2	8.9	23.3	19.3	39.6	35.0	35.8	30.1	13.8	8.3	29.7	24.5
NRNS + SLING [29]	40.2	31.8	37.2	23.9	19.8	9.8	32.4	21.8	49.8	35.0	40.8	30.4	21.6	12.7	37.4	26.0
IGL-Nav (Ours)	<b>70.4</b>	<b>64.2</b>	<b>60.6</b>	<b>51.4</b>	<b>40.0</b>	<b>28.9</b>	<b>57.0</b>	<b>48.2</b>	<b>77.2</b>	<b>73.1</b>	<b>69.8</b>	<b>60.1</b>	<b>42.8</b>	<b>31.9</b>	<b>63.3</b>	<b>55.0</b>

Table 3. Performance of IGL-Nav when depth and camera intrinsics are unavailable.

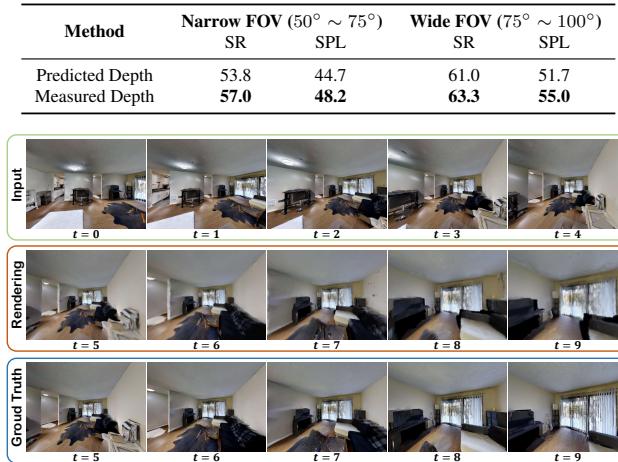


Figure 5. Rendering results of our incremental 3DGS.

transferred from image-goal to free-view image-goal setting, IGL-Nav still maintains a huge performance lead compared with other state-of-the-art methods. The performance of IGL-Nav can be further boosted with training data on the free-view image-goal task. Note that the zero-shot transferring performance of IGL-Nav is even better than other methods under supervised setting, which demonstrates the great generalization ability of our approach.

### 4.3. Analysis of IGL-Nav

We further conduct in-depth module-by-module analysis on our IGL-Nav framework with sufficient visualization results and ablation studies, which is divided into three parts according to our module design. All ablation studies are conducted on the free-view image-goal setting.

**Incremental 3DGS Prediction.** Following the setting of RNR-Map [14], we assume depth information and cam-

Table 4. Effects of different subdivision levels in coarse target localization to the final performance.

Level ( $\gamma$ )	Narrow FOV ( $50^\circ \sim 75^\circ$ )		Wide FOV ( $75^\circ \sim 100^\circ$ )	
	SR	SPL	SR	SPL
1	19.7	12.0	24.9	16.8
2	41.3	34.4	48.9	42.1
3	<b>57.0</b>	<b>48.2</b>	<b>63.3</b>	<b>55.0</b>

Table 5. Effects of different stoppers in fine target localization to the final performance.

Stopper	Narrow FOV ( $50^\circ \sim 75^\circ$ )		Wide FOV ( $75^\circ \sim 100^\circ$ )	
	SR	SPL	SR	SPL
IGL-Nav w/out Stopper	45.7	32.9	46.2	37.6
IGL-Nav w/ SLING [29]	49.0	40.7	52.4	45.0
IGL-Nav	<b>57.0</b>	<b>48.2</b>	<b>63.3</b>	<b>55.0</b>

era intrinsics are known in our experiments. When these information is unavailable, we can simply adopt a depth estimator [21] to predict them. As shown in Table 3, with predicted depth and camera intrinsics, the performance of IGL-Nav is still robust. We further visualize rendering results of our 3DGS representation in Figure 5. Although maintained in an incremental and feed-forward manner, our 3DGS still demonstrates photorealistic novel view synthesis capability.

**Coarse Target Localization.** In our coarse localization module, the sphere space is discretized with a regular icosahedron and its  $\gamma$ -level fractal. A larger value of  $\gamma$  leads to finer discretization of the spherical surface and results in a greater number of convolution kernels  $N$ . We study the effects of different levels to the final performance in Table 4. It is shown that using a 3-level subdivision achieves best performance, because a finer discretization will reduce quantization error and improve the accuracy of coarse localization. However, a larger  $\gamma$  results in high computational cost, making training inefficient.

**Fine Target Localization.** We compare different stoppers in Table 5. The first row refers to only using coarse

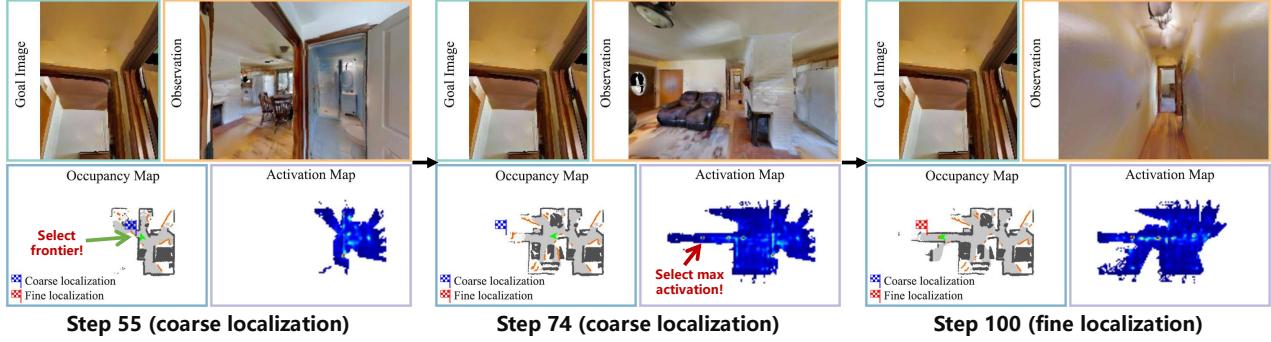


Figure 6. Visualization of navigation process in Habitat simulator.



Figure 7. Visualization of navigation process in the real world. The agent is successfully guided to a free-view goal image captured by a cellphone in complex indoor environments. IGL-Nav exhibits strong generalization ability and sim-to-real transfer performance.

target localization, and the second row refers to using the widely adopted SLING [29] as the stopper and fine localization module. It is shown that our 3DGS-based stopper and matching-constrained optimization is more suitable for the overall navigation system of IGL-Nav.

We also visualize the navigation process in Figure 6. The agent is guided with frontier location, activation map obtained with 3D convolution and iterative pose optimization during the exploration. It is shown that our IGL-Nav can effectively localize the target image even with partial observation, and accurately guide the agent to final location with fine-grained rendering-based optimization.

#### 4.4. Real-world Deployment

We further deploy IGL-Nav on real-world robotic platform to test its generalization ability. The model is directly taken from the free-view image-goal setting (supervised) without any finetuning on real-world data. As shown in Figure 1 and 7, we use a cellphone to capture the target image in a viewpoint that is unreachable by the robotic agent’s camera. Benefit from the flexible rendering capability of 3DGS rep-

resentation, the agent effectively reaches this free-view goal with the coarse-to-fine localization method.

## 5. Conclusion

In this paper, we have proposed IGL-Nav for efficient and 3D-aware image-goal navigation. We incrementally maintain a 3DGS scene representation in feed-forward manner, which is then utilized for coarse-to-fine target localization. We analyze the pose space of the goal image and discretize both the pose space and scene embedding to apply efficient 3D convolution-based coarse matching. When the agent is close to the goal, we switch to fine localization by optimizing the camera pose via differentiable rendering on the confident matching pairs. The proposed IGL-Nav significantly outperforms existing state-of-the-art methods on image-goal and free-view image-goal settings. Real-world experiments further demonstrate our generalization ability. A limitation of IGL-Nav is that it requires depth and camera intrinsics of goal image. However, as we show in experiments, using existing monocular depth estimation [21] to predict them can satisfactorily solve this problem.

## References

- [1] Ziad Al-Halah, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. In *CVPR*, pages 17031–17041, 2022. [2](#), [6](#)
- [2] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, pages 4247–4258, 2020. [1](#)
- [3] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, pages 12875–12884, 2020. [1](#), [2](#)
- [4] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, pages 19457–19467, 2024. [2](#), [3](#)
- [5] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *ECCV*, pages 338–355. Springer, 2024.
- [6] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, pages 370–386. Springer, 2024. [2](#), [3](#)
- [7] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. *NeurIPS*, 34:26661–26673, 2021. [1](#), [3](#), [6](#), [7](#)
- [8] Faith Johnson, Bryan Bo Cao, Ashwin Ashok, Shubham Jain, and Kristin Dana. Feudal networks for visual navigation. *arXiv preprint arXiv:2402.12498*, 2024. [6](#)
- [9] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *CVPR*, pages 21357–21366, 2024. [5](#)
- [10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 42(4):139–1, 2023. [2](#)
- [11] Nuri Kim, Obin Kwon, Hwiyeon Yoo, Yunho Choi, Jeongho Park, and Songhwai Oh. Topological semantic graph memory for image-goal navigation. In *CoRL*, pages 393–402. PMLR, 2023. [2](#)
- [12] Jacob Krantz, Stefan Lee, Jitendra Malik, Dhruv Batra, and Devendra Singh Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances. *arXiv preprint arXiv:2211.15876*, 2022. [2](#)
- [13] Jacob Krantz, Theophile Gervet, Karmesh Yadav, Austin Wang, Chris Paxton, Roozbeh Mottaghi, Dhruv Batra, Jitendra Malik, Stefan Lee, and Devendra Singh Chaplot. Navigating to objects specified by images. In *ICCV*, pages 10916–10925, 2023. [3](#)
- [14] Obin Kwon, Jeongho Park, and Songhwai Oh. Renderable neural radiance map for visual navigation. In *CVPR*, pages 9099–9108, 2023. [1](#), [2](#), [3](#), [6](#), [7](#)
- [15] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, pages 12697–12705, 2019. [5](#)
- [16] Xiaohan Lei, Min Wang, Wengang Zhou, and Houqiang Li. Gaussnav: Gaussian splatting for visual navigation. *T-PAMI*, 2025. [2](#)
- [17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. [5](#)
- [18] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, et al. Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation. *arXiv preprint arXiv:2408.14873*, 2024. [2](#)
- [19] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *ECCV*, pages 349–366. Springer, 2024. [2](#)
- [20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421. Springer, 2020. [1](#)
- [21] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *CVPR*, pages 10106–10116, 2024. [3](#), [5](#), [7](#), [8](#)
- [22] Mohammad Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhishek Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*, 2024. [2](#)
- [23] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *CVPR*, pages 18890–18900, 2022. [1](#)
- [24] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018. [2](#)
- [25] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, pages 9339–9347, 2019. [6](#)
- [26] James A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999. [6](#)
- [27] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, pages 8922–8931, 2021. [5](#)
- [28] Yuan Sun, Xuan Wang, Yunfan Zhang, Jie Zhang, Caigui Jiang, Yu Guo, and Fei Wang. icomma: Inverting 3d gaussians splatting for camera pose estimation via comparing and matching. *arXiv preprint arXiv:2312.09031*, 2023. [5](#)
- [29] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *CoRL*, pages 666–678. PMLR, 2023. [2](#), [6](#), [7](#), [8](#)

- [30] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019. [6](#), [7](#)
- [31] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079, 2018. [6](#)
- [32] Karmesh Yadav, Arjun Majumdar, Ram Ramrakhya, Naoki Yokoyama, Alexei Baevski, Zsolt Kira, Oleksandr Maksymets, and Dhruv Batra. Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav. *arXiv preprint arXiv:2303.07798*, 2023. [1](#), [2](#)
- [33] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. In *ICLRW*, 2023. [2](#), [6](#), [7](#)
- [34] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. In *NeurIPS*, 2024. [1](#)
- [35] Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Unigoal: Towards universal zero-shot goal-oriented navigation. In *CVPR*, pages 19057–19066, 2025. [1](#)
- [36] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, pages 11784–11793, 2021. [5](#)
- [37] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. [3](#)
- [38] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zeng-mao Wang, Lina Liu, et al. Gaussianrasper: 3d language gaussian splatting for open-vocabulary robotic grasping. *RA-L*, 2024. [2](#)
- [39] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. [1](#), [2](#)