2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# AutoNeRF: Training Implicit Scene Representations with Autonomous Agents

Pierre Marza[1,2], Laetitia Matignon[2], Olivier Simonin[1], Dhruv Batra[3,5], Christian Wolf[4], Devendra S. Chaplot[6]

***Abstract*— Implicit representations such as Neural Radiance Fields (NeRF) allow to map color, density and semantics in a 3D scene through a continuous neural function. However, these models typically require manual and careful human data collection for training. This paper addresses the problem of active exploration for autonomous NeRF construction. We study how an agent can learn to efficiently explore an unknown 3D environment so that the data collected during autonomous exploration enables the learning of a high-quality neural implicit map representation. The quality of the learned representation is evaluated on four robotics-related downstream tasks: classical viewpoint rendering, map reconstruction, planning, and pose refinement. We compare the impact of different exploration strategies including frontier-based and learning-based approaches (end-to-end and modular) with different reward functions tailored to this problem. Empirical results show that NeRFs can be trained on actively collected data using just a single episode of experience in an unseen environment and that AutoNeRF, a modular exploration policy trained with reinforcement learning, enables obtaining a higher-quality NeRF for the considered downstream robotic tasks. Finally, we show that with AutoNeRF an agent can be deployed to a previously unknown scene and then automatically improve its navigation performance by adapting to the scene through a cycle of exploration, reconstruction, and policy finetuning.**

## I. Introduction

Exploration is a key challenge in building autonomous navigation agents that operate in unseen environments. In the last few years, there has been a significant amount of work on training exploration policies to maximize coverage [1], [2], [3], find goals specified by object categories [4], [5], [6], images [7], [8] or language [9], and for embodied active learning [10]. Among these methods, modular learning methods have been very effective at various embodied tasks [1], [4], [11]. These methods learn an exploration policy that can build an explicit semantic map of the environment which is then used for planning and downstream embodied AI tasks such as Object Goal or Image Goal Navigation.

Concurrently, in the computer graphics and vision communities, there has been a recent but large body of work on learning implicit map representations, particularly based on Neural Radiance Fields (NeRF) [12], [13], [14]. Prior methods [15], [16] demonstrate strong performance in novel view synthesis and are appealing from a scene understanding point of view as a compact and continuous representation of appearance and semantics in a 3D scene. However, most approaches require training frames (RGB, depth, semantics) that should be collected manually. Can we train embodied agents to explore an unseen environment efficiently to collect data that can be used to create implicit map representations or NeRFs autonomously? In this paper, our objective is to tackle this problem of active exploration for autonomous NeRF construction. If an embodied agent is able to build an implicit map representation autonomously, it can then use it for a variety of downstream tasks such as planning, pose estimation, and navigation. Just a single episode or a few minutes of exploration in an unseen environment can be sufficient to build an implicit representation later used for improving the performance of the agent in that environment for several tasks without any additional supervision.

In this work, we introduce AutoNeRF, a modular policy trained with Reinforcement Learning (RL) that can explore an unseen 3D scene to collect data for training a NeRF model autonomously (Figure 1). While most prior work evaluates NeRFs on rendering quality, we propose a range of downstream tasks to evaluate them (and indirectly, the exploration policies used to gather data for training these representations)

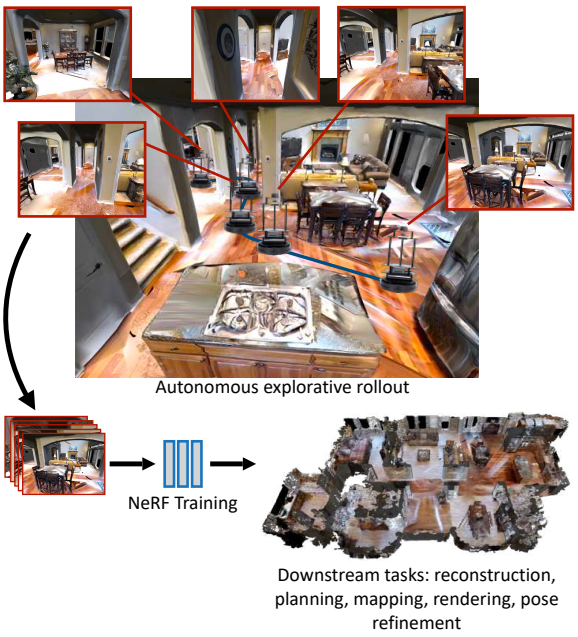


**Fig. 1:** We propose a method for automatically generating 3D models of a scene by training NeRFs from data collected by autonomous agents. We compare classical and RL-trained exploration policies, with different reward functions and evaluate the implicit representations on rendering, mapping, planning, pose refinement.

[1]INSA Lyon, CITI Lab, INRIA Chroma team, [2]Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 [3]Meta AI, [4]Naver Labs Europe, [5]Georgia Tech, [6]Mistral AI

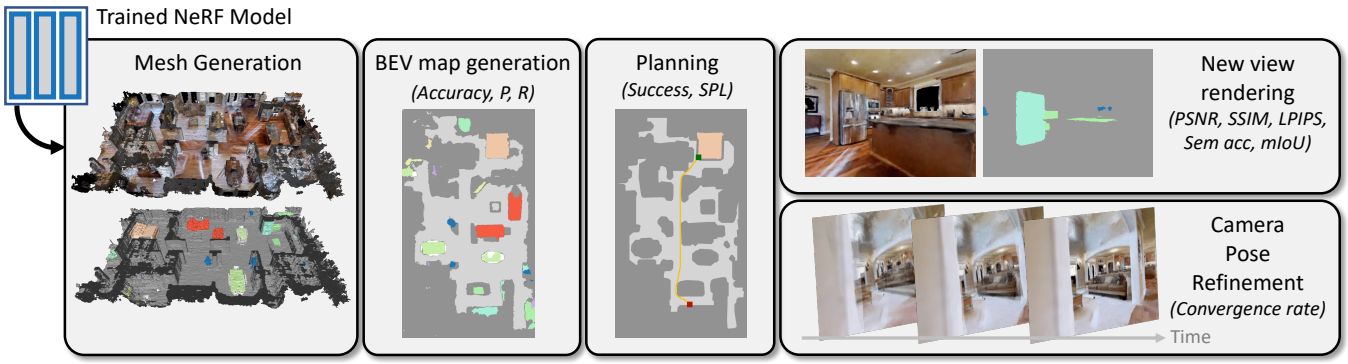


**Fig. 2: Downstream tasks** — the model trained from autonomously collected data is used for several downstream tasks related to robotics: Birds-eye-view map generation and planning on this map; new view generation of RGB and semantic frames; camera pose refinement (visual servoing). Mesh generation for the covered scene (color or semantic mesh) is only performed as a qualitative evaluation.

for Embodied AI applications. Specifically, we use geometric and semantic map prediction accuracy, planning accuracy for Object Goal and Point Goal navigation and camera pose refinement (Figure 2). We show that AutoNeRF outperforms the well-known frontier exploration algorithm as well as state-of-the-art learnt policies, and also study the impact of different reward functions on the downstream performance of the NeRF model. We also study how AutoNeRF can be used to autonomously adapt policies to a specific scene at deployment by providing a high-quality reconstruction of large-scale environments that can be loaded into a simulator to improve the performance of any given agent.

Our contributions can be summarized as follows: (i) We address the autonomous collection of NeRF training data with an egocentric agent in house-scale scenes. We show that a modular RL-trained policy can achieve such a task, and compare its performance with state-of-the-art exploration methods. (ii) We study how to properly evaluate the performance of a trained NeRF model in the context of robotics: we consider several robotics-related downstream tasks that provide a quite complete picture. (iii) We finally explore applications such as house-scale scene reconstruction, and more specifically, how an agent can autonomously improve its navigation skills by finetuning itself on a scene.

More resources to explore our work, including access to released code (https://github.com/PierreMarza/autonerf), can be found at https://pierremarza.github.io/projects/autonerf/.

## II. Related Work

**Neural fields —** represent the structure of a 3D scene with a neural network. They were initially introduced in [17], [18] as an alternative to discrete representations such as voxels, point clouds or meshes. Neural Radiance Fields (NeRF) [12] then introduced a differentiable volume rendering loss allowing to supervise 3D scene reconstruction from 2D supervision, achieving state-of-the-art performance on novel view synthesis. Follow-up work has addressed faster training and inference [13], or training from few images [14].

**Neural fields in robotics —** implicit representations have also been proposed for real-time SLAM [19], [20], [21]. [20] introduced a hierarchical implicit representation to represent large scenes and [21] performed SLAM without requiring depth information. [15] augmented NeRFs with a semantic head trained from sparse and noisy 2D semantic maps. Implicit representations can map occupancy, explored area, and semantic objects to navigate towards [22], or the density of a scene for drone obstacle avoidance [23]. They have also been used for camera pose refinement through SGD directly on a loss in rendered pixel space [24]. In contrast to the literature, we investigate training these representations from data collected by autonomous agents directly and explore the effect of the choice of policy on downstream robotics tasks.

**Active learning for neural fields —** originally tackles the active selection of training data in fixed datasets of 2D frames: ActiveNeRF [25] estimates NeRF uncertainty by expressing radiance values as Gaussian distributions, and ActiveRMAP [26] maximizes an entropy-based information gain metric. More recent work ([27], [28]) studies robot navigation to actively collect data to train a 3D neural representation. Both visit space where the representation has highest uncertainty, but do not use neural networks to select global goals, while we specifically study the ability to learn to predict navigation waypoints. [28] builds a neural signed distance field from depth information, while we consider NeRF representations that are built without depth input, and reconstruct colors and semantics. Finally, [27] targets small scenes unlike what we show in this work.

**Autonomous scene exploration —** is generally defined as a coverage maximization problem, a baseline being Frontier Based Exploration (FBE) [29]. Different variants exist [30], [31] but the core principle is to maintain a frontier between explored and unexplored space and to sample points on it. Learning-based approaches are explored in recent work [1], [2], [3]. This work studies how different definitions of exploration impact the quality of an implicit scene representation.

## III. Background

### A. Modular exploration policies

The trained policy explores a 3D scene to collect a sequence of RGB and semantic frames, along with camera poses, that will be used to train a NeRF model. Following [4], [1], we adapt a modular policy composed of a *Mapping* process that builds a semantic map, a *Global Policy* that outputs a global

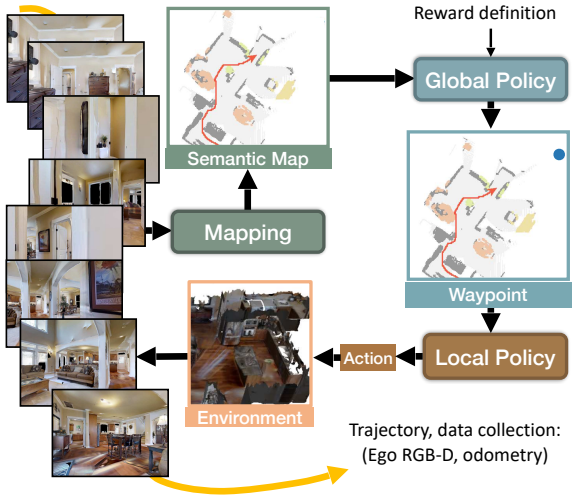


**Fig. 3:** We adapt the modular policy from [4]: a mapping module generates a semantic and occupancy top-down map from egocentric RGB-D observations and sensor pose. A high-level policy trained with RL predicts global waypoints, which are followed by a handcrafted low-level policy (fast marching). The sequence of observations comprises the data input to NeRF training.

waypoint from the semantic map as input, and finally, a *Local Policy* that navigates towards the global goal, see Figure 3.

**Semantic Map —** a 2D top-down map is maintained at each time step $t$, with several components: (i) an occupancy component $\mathbf{m}_t^{occ} \in \mathbb{R}^{M\times M}$ stores information on free navigable space; (ii) an exploration component $\mathbf{m}_t^{exp} \in \mathbb{R}^{M\times M}$ sets to 1 all cells which have been within the agent's field of view since the beginning of the episode; (iii) a semantic component $\mathbf{m}_t^{sem} \in \mathbb{R}^{S\times M\times M}$, where $M\times M$ is the spatial size and $S$ denotes the number of channels storing information about the scene. Additional maps store the current and previous agent locations. Egocentric maps are updated by inverse projection from the depth frames and pooling to the ground plane, and are integrated over time taking into account agent poses estimated from sensor information. The semantic maps additionally use predictions obtained with Mask R-CNN [32].

**Policies —** intermediate waypoints are predicted by the *Global Policy*, a convolutional neural network taking as input the stacked maps (we follow [4]) and is trained with RL/PPO [33]. A *Local Policy* navigates towards the waypoint taking discrete actions for 25 steps following the path planned using the *Fast Marching Method*.

### B. Neural radiance fields

**Vanilla Semantic NeRF —** Neural Radiance Fields [12] are composed of MLPs predicting the density $\sigma$, color $c$ and, eventually as in [15], the semantic class $s$ of a particular $3D$ position in space $\mathbf{x} \in \mathbb{R}^3$, given a 2D camera viewing direction $\phi \in \mathbb{R}^2$. NeRFs have been designed to render new views of a scene provided a camera position and viewing direction. The color of a pixel is computed by performing an approximation of volumetric rendering, sampling $N$ quadrature points along the ray. Given multiple images of a scene along with associated camera poses, a NeRF is trained with Stochastic Gradient Descent minimizing the difference between rendered and ground-truth images.

**Semantic Nerfacto —** we leverage recent advances to train NeRF models faster while maintaining high rendering quality and follow what is done in the Nerfacto model from [34], that we augment with a semantic head. The inputs $\mathbf{x}$ and $\phi$ are augmented with a learned appearance embedding $\mathbf{e} \in \mathbb{R}^{32}$. Both $\mathbf{x}$ and $\phi$ are first encoded using respectively a hash encoding function $h$ as $\tilde{\mathbf{x}} = h(\mathbf{x})$ and a spherical harmonics encoding function $sh$ as $\tilde{\phi} = sh(\phi)$. $\tilde{\mathbf{x}}$ is fed to an MLP $f_d$ predicting the density at the given 3D position, yielding $(\sigma, \mathbf{h_d}) = f_d(\tilde{\mathbf{x}}; \Theta_d)$, where $\mathbf{h_d}$ is a latent representation. $\mathbf{h_d}$ is fed to another MLP model $f_s$ that outputs a softmax distribution over the $S$ considered semantic classes as $\mathbf{s} = f_s(\mathbf{h_d}; \Theta_s)$ where $\mathbf{s} \in \mathbb{R}^S$. Finally, $\mathbf{h_d}$, $\tilde{\phi}$ and $\mathbf{e}$ are the inputs to $f_c$ that predicts the RGB value at the given 3D location, $\mathbf{c} = f_c(\mathbf{h_d}, \tilde{\phi}, \mathbf{e}; \Theta_c)$ where $\mathbf{c} \in \mathbb{R}^3$.

## IV. AutoNeRF

We present AutoNeRF, a method to collect NeRF training data with an autonomous embodied agent. The latter is initialized in an unseen environment and must gather data in a single episode with a fixed time budget. Collected observations are then used to train a neural implicit representation of the scene (density, RGB, semantics) which is finally evaluated on several robotics-related downstream tasks: new view rendering, mapping, planning and pose refinement.

**Task Specification —** The agent is initialized at a random location in an unknown scene and at each timestep $t$ can execute a discrete action in the space $\Lambda = \{$`FORWARD 25cm`, `TURN_LEFT 30°`, `TURN_RIGHT 30°`$\}$. At each step, the agent receives an observation $\mathbf{o}_t$ composed of an egocentric RGB frame and a depth map. The field of view of the agent is 90°. It also has access to odometry information. The agent can navigate for a limited number of 1500 discrete steps.

Training is done in two phases: (i) we first train an exploration policy to collect observations on a set of training scenes in a self-supervised manner, i.e. using intrinsic rewards. (ii) Secondly, we use the trained policy to collect data in unseen test scenes, one trajectory per scene, and train a NeRF model using this data. The trained NeRF model is then evaluated on the set of downstream tasks.

### A. Exploration Policy Training

We consider different reward signals for training the *Global Policy* tailored to our task of scene reconstruction, and which differ in the importance they give to different aspects of the scene. All these signals are computed in a self-supervised fashion using the metric map representations built by the exploration policy.

**Explored area** — (*Ours (cov.)*) optimizes the coverage of the scene, i.e. the size of the explored area, and has been proposed in the literature, e.g. in [4], [1]. It accumulates differences in the exploration component $\mathbf{m}_t^{exp}$,

$$r_t^{cov} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} \mathbf{m}_t^{exp}[i,j] - \mathbf{m}_{t-1}^{exp}[i,j]$$

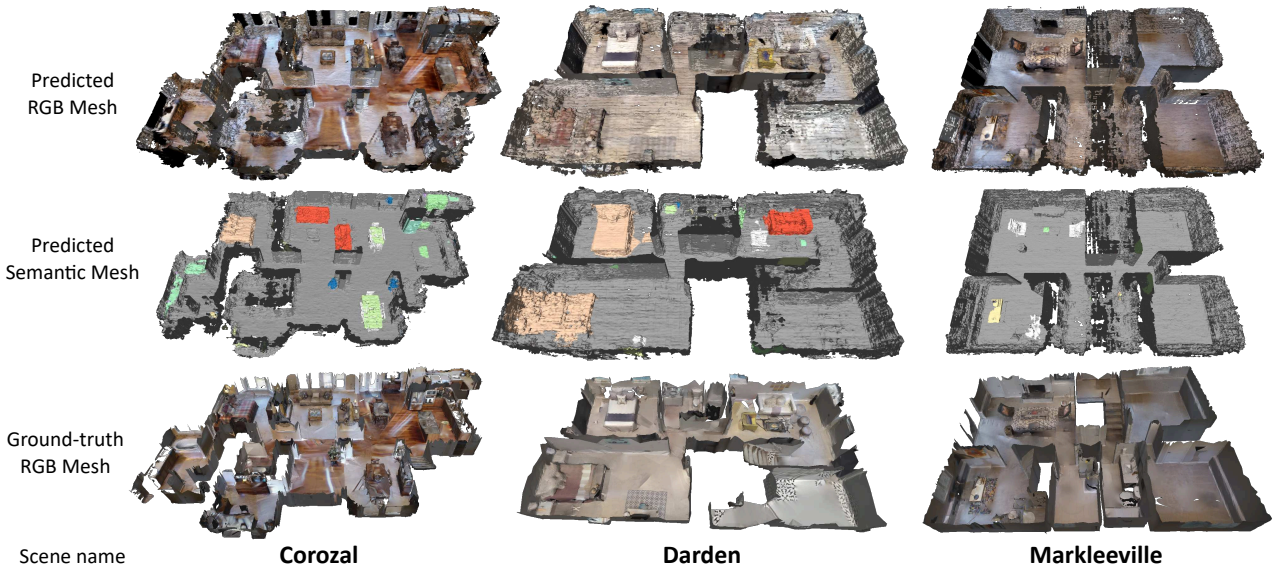


**Fig. 4: Mesh reconstruction**: reconstruction of 3 Gibson val scenes extracted from a NeRF model trained on data gathered by our modular policy. Both geometry, semantics, and appearance are satisfying. Exploration with the modular policy *Ours (obs).*

**Obstacle coverage** — (*Ours (obs.)*) optimizes the coverage of obstacles in the scene, and accumulates differences in the corresponding component $\mathbf{m}_t^{occ}[i,j]$. It targets tasks where obstacles are considered more important than navigable floor space, which is arguably the case when viewing is less important than navigating.

$$r_t^{obs} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} \mathbf{m}_t^{occ}[i,j] - \mathbf{m}_{t-1}^{occ}[i,j]$$

**Semantic object coverage** — (*Ours (sem.)*) optimizes the coverage of the $S$ semantic classes detected and segmented in the semantic metric map $\mathbf{m}_t^{sem}$. This reward removes obstacles that are not explicitly identified as a notable semantic class — see section V for their definition.

$$r_t^{sem} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\sum_{k=0}^{S-1} \mathbf{m}_t^{sem}[i,j,k] - \mathbf{m}_{t-1}^{sem}[i,j,k]$$

**Viewpoints coverage** — (*Ours (view.)*) optimizes for a dense and continuous representation of the scene usable to render arbitrary new viewpoints. To this end, we propose to maximize coverage not only in terms of agent positions but also agent viewpoints. Compared to [4], we introduce an additional 3D map $\mathbf{m}^{view}[i,j,k]$, where the first two dimensions correspond to spatial 2D positions in the scene and the third dimension corresponds to a floor plane angle of the given cell discretized into $V{=}12$ bins. A value of $\mathbf{m}_t^{view}[i,j,k] = 1$ indicates that cell $(i,j)$ has been seen by the agent from a (discretized) angle $k$. The reward maximizes its changes,

$$r_t^{view} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\sum_{k=0}^{V-1} \mathbf{m}_t^{view}[i,j,k] - \mathbf{m}_{t-1}^{view}[i,j,k]$$

### B. NeRF training

The sequence of observations collected by the agent comprises egocentric RGB frames $\{\mathbf{o}_t\}_{t=1\ldots T}$, semantic segmentations $\{\mathbf{s}_t\}_{t=1\ldots T}$ and associated poses $\{\mathbf{p}_t\}_{t=1\ldots T}$ in a reference frame, which we define as the starting position $t{=}0$ of each episode. In most experiments, we use pose and semantics information from simulation. We also study the difference between using Ground Truth (GT) semantics from a simulator and a Mask R-CNN [32] model (Table VII). Finally, we evaluate the performance under realistic actuation and sensing noise (Tables VIII, IX, X, XI).

An important property of this procedure is that no depth information is required for reconstruction. The implicit representation is trained by mapping pixel coordinates $\mathbf{x}_i$ for each pixel $i$ to RGB values and semantic values with the volume rendering loss described in Section III-B. The input coordinates $\mathbf{x}_i$ are obtained using the global poses $\mathbf{p}_t$ and intrinsics from calibrated cameras.

### C. Downstream tasks

Prior work on implicit representations generally focused on two different settings: (i) evaluating the quality of a neural field based on its new view rendering abilities given a dataset of (carefully selected) training views, and (ii) evaluating the quality of a scene representation in robotics conditioned on given (constant) trajectories, evaluated as reconstruction accuracy. We cast this task in a more holistic way and more aligned with our scene understanding objective. We evaluate the impact of trajectory generation (through exploration policies) directly on the quality of the representation, which we evaluate in a goal-oriented way through multiple tasks related to robotics (cf. Figure 2).

**Task 1: Rendering —** This task is the closest to the evaluation methodology prevalent in the neural field literature. We evaluate the rendering of RGB and semantic frames as proposed in [15]. Unlike the common method of evaluating an implicit representation on a subset of frames within the trajectory, we evaluate it on a set of uniformly sampled camera poses within the scene, independently of the trajectory taken by the policy used to collect training

data. This allows us to evaluate the representation of the complete scene and not just the ability of the NeRF model to interpolate between close poses within the training trajectory.

We render ground-truth images and semantic masks associated with sampled camera poses using the Habitat [35] simulator and compare them against NeRF renderings. RGB rendering metrics are PSNR (absolute errors), SSIM (retrieved structural information) and LPIPS [36] (distance in neural feature space). Rendering of semantics is evaluated in terms of average per-class accuracy and mean intersection over union (mIoU).

**Task 2: Metric Map Estimation —** While rendering quality is linked to the perception of the scene, it is not necessarily a good indicator of its structural content, which is crucial for robotic downstream tasks. We evaluate the quality of the estimated structure by translating the NeRF continuous representation into a format, which is very widely used in map-and-plan baselines for navigation, a top-down (bird's-eye-view=BEV) map storing occupancy and semantic category information and compare it with the ground-truth from the simulator. We evaluate obstacle and semantic maps using accuracy, precision, and recall. It is important to differentiate between the topdown map used by the policy during navigation and the mentioned BEV map predicted here for evaluation purpose. The former is used as a quickly-computed coarse representation of the environment to help with navigation, and the latter is used to evaluate the fine-grained precision of the NeRF scene representation.

**Task 3: Planning —** Using maps for navigation, it is difficult to pinpoint the exact precision required for successful planning, as certain artifacts and noises might not have a strong impact on reconstruction metrics, but could lead to navigation problems. We perform goal-oriented evaluation and measure to what extent path planning can be done on the obtained top-down maps.

We sample 100 points on each scene and plan from those starting points to two types of goals: selected end positions for *PointGoal* planning, and objects categories for *ObjectGoal* planning. The latter requires planning the shortest path from the given starting point to the closest object of each semantic class available on the given scene. For both tasks, we plan with the *Fast Marching Method* and report mean *Success* and *SPL* as introduced in [37]. For a given episode, *Success* is 1 if planning stops less than 1m from the goal, and *SPL* measures path efficiency.

**Task 4: Pose Refinement —** This task introduced in [24] involves correcting an initial noisy camera pose given a RGB frame captured by the camera. We address this problem by taking the trained NeRF model, freezing its weights and optimizing the input camera pose to minimize the reconstruction difference between the NeRF-rendered frame and the provided one, as done in [24]. This task is closely linked to visual servoing with a "eye-in-hand" configuration, a standard problem in robotics, in particular in its "direct" variant [38], where the optimization is directly performed over losses on the observed pixel space.

**Fig. 5: Navigating in the Habitat simulator**: the underlying mesh was extracted from the trained NeRF, *Ours (cov)*. Rendering quality and the generated BEV map are correct, as are free navigable space and collision handling. Temporal order indicated by →.

To generate episodes of starting and end positions, we take 100 sampled camera poses in each scene and apply a random transformation to generate noisy poses. The model is evaluated in terms of rotation and translation convergence rate, i.e. percentage of samples where the final difference with ground truth is less than $3^\circ$ in rotation and 2 cm in translation. We also report the mean translation and rotation errors for the converged samples.

## V. Experimental Results

**Modular Policy training —** is performed on one V100 GPU for 7 days. All modular policies are trained on the 25 training scenes of the Gibson [39]-tiny set. Mask R-CNN model is pre-trained on the MS COCO dataset [40] and finetuned on Gibson train scenes. We consider $S$=15 semantic categories: {*chair*, *couch*, *potted plant*, *bed*, *toilet*, *tv*, *dining table*, *oven*, *sink*, *refrigerator*, *book*, *clock*, *vase*, *cup*, *bottle*}.

**External baselines —** We compare our trained modular policies against the classical FBE [29] (*Frontier*), as well as end-to-end policies trained with RL. More specifically, we consider four end-to-end policies from [3], that all share the same architecture but were trained with different exploration-related reward functions: coverage (*E2E (cov.)*), curiosity (*E2E (cur.)*), novelty (*E2E (nov.)*), reconstruction (*E2E (rec.)*). Reward functions are presented in [3].

**Evaluation —** consists in running 5 rollouts with different start positions for each policy, in each of the 5 Gibson-tiny validation scenes (different from training scenes). A NeRF model is then trained on each trajectory data. As evaluation scenes are different from training ones (i.e. different 3D environments), we evaluate the generalization of the policy to unknown new scenes that were not seen at training time.

**NeRF models —** In our experiments, we consider two different NeRF variants presented in Section III-B. Most experiments are conducted with *Semantic Nerfacto*, as it provides a great trade-off between training speed and quality of representation. *Semantic Nerfacto* is built on top of the *Nerfacto* model from the nerfstudio [34] library. We augment the model with a semantic head and implement evaluation on test camera poses independently from the collected trajectory. Only the subsections V-A and V-B will involve training

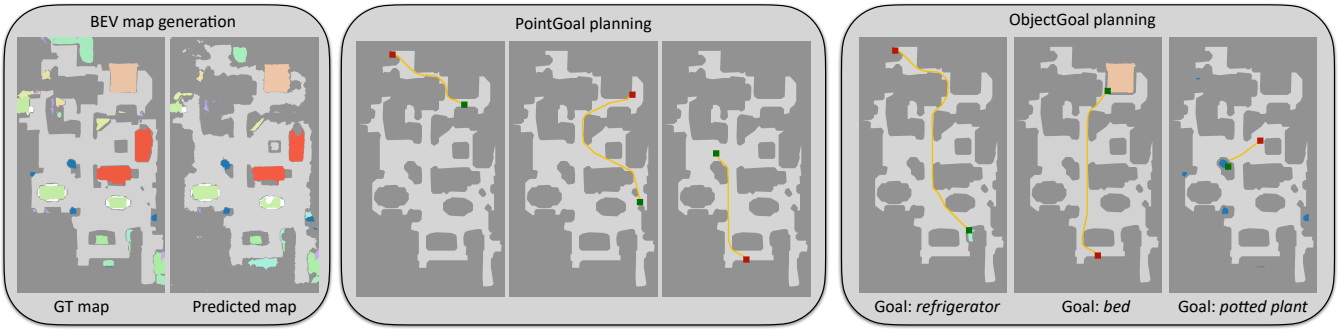


**Fig. 6: BEV map tasks**: Generation of semantic BEV maps (Left), *PointGoal* (Middle) and *ObjectGoal* planning (Right).

| Mesh | Success ↑ | SPL ↑ |
|---|---|---|
| **Original Gibson mesh** | 88.1 | 73.3 |
| **Rollout → NeRF training → Mesh generation** | 78.4 | 67.7 |

**TABLE I: Navigating in a NeRF-generated mesh**: Average PointGoal performance of a policy planning a path with the Fast Marching Method on 4 Gibson val scenes in Habitat, from either the original mesh (**Gibson**) or AutoNeRF mesh (**Rollout → NeRF train. → Mesh gen.**) trained from data collected by *Ours (obs.)*.

| Policy | Success ↑ | SPL ↑ |
|---|---|---|
| ***Finetuned on Gibson meshes (not comparable)***† | 99.7 | 97.9 |
| **Pre-trained (no finetuning)** | 90.2 | 82.9 |
| **Finetuned on AutoNeRF meshes** | **92.9** | **86.7** |

**TABLE II: PointGoal Finetuning**: finetuning a PointGoal agent on a mesh generated with AutoNeRF improves mean performance over a pre-trained policy on a specific scene. † an upper bound which finetunes on the original mesh. In a real use case involving a robot collecting data, this mesh would not be available (not comparable).

a *vanilla Semantic NeRF* model, more precisely the one introduced in [15] that also contains a semantic head. We chose this variant for these specific experiments to illustrate the possibility of providing high-fidelity representations of complex scenes. Results from *Semantic Nerfacto* are still very good (see Figures 6 and 7) but we found meshes to be of higher quality with a *vanilla Semantic NeRF* model trained for a longer time (12h).

### A. Reconstructing house-scale scenes

We illustrate the possibility of autonomously reconstructing complex large-scale environments such as apartments or houses from the continuous representations trained on data collected by agents exploring the scene using the modular policy. Figure 4 shows RGB and semantic meshes for 3 Gibson val scenes. Geometry, appearance, and semantics are satisfying. In Figure 5 we show that such meshes can be loaded into the Habitat simulator and allow proper navigation and collision computations. Both occupancy top-down map generation and RGB renderings are performed by the Habitat simulator from the generated mesh. To further evaluate their quality, we perform PointGoal navigation on NeRF-generated meshes in Habitat for 4 Gibson val scenes. The agent is based on the modular policy, restricted to the planning part performed with Fast Marching Method, relying on the map channels related to obstacles and explored area. Table I shows a performance drop between the navigation on original Gibson meshes ("soft upper bound") and reconstructed ones, but Success and SPL are close, showing that

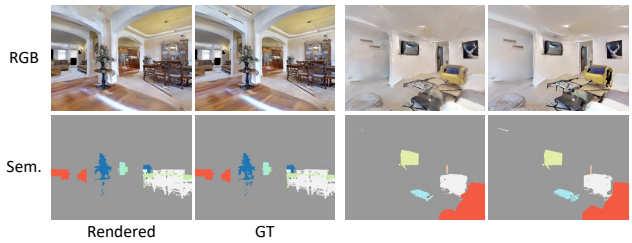


**Fig. 7: Quality of rendering**: RGB and semantic renderings compared with GT from simulation. Data collected by *Ours (obs)*.

our NeRF-generated mesh features a satisfying geometry, allowing to navigate properly when loaded within Habitat.

### B. Autonomous adaptation to a new scene

A long-term goal of Embodied AI is to train general policies that can be deployed on any new scene, but even such agents will likely struggle with some specificities of a given environment. Thus fine-tuning the general policy to a specific environment is a relevant solution that would be more secure and faster if done in simulation. We explore here the usage of AutoNeRF to explore an environment to train a NeRF, then extract from it a mesh representation that could be loaded into a simulator to fine-tune the robot general policy to the specific environment. More specifically, we consider a depth-only PointGoal navigation policy pre-trained on Gibson train scenes. This pre-trained policy is finetuned on 4 *unseen* scenes (4 Gibson val scenes) by using 4 meshes generated by AutoNeRF. The 4 finetuned policies are then evaluated on the original Gibson meshes. As shown in Table II, scene-specific finetuning on autonomously reconstructed 3D meshes allows to improve both Success and SPL. We also compare with finetuning directly on the Gibson mesh, which provides a non-comparable soft upper bound — in a real robotics scenario, these meshes would not be accessible. This shows that performance could still be improved, but it is important to note that reaching the performance of the upper bound might be about reconstructing fine details.

### C. Quantitative results on robotics downstream tasks

**Frontier-Based Exploration vs Modular Policy —** as can be seen in Tables III, IV, V, VI, RL-trained modular policies outperform FBE on all metrics. This is a somewhat surprising result, since FBE generally performs satisfying visual coverage of the scene, even though it can sometimes get

| Policy | RGB | | | Semantics | |
|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Per-class acc ↑ | mIoU ↑ |
| **Frontier** | 19.75 | 0.743 | 0.343 | 81.4 | 65.7 |
| **E2E (cov.)** | 20.94 | 0.750 | 0.332 | 80.1 | 63.9 |
| **E2E (cur.)** | 20.60 | 0.747 | 0.338 | 78.7 | 61.9 |
| **E2E (nov.)** | 23.36 | 0.801 | 0.268 | 84.6 | 71.4 |
| **E2E (rec.)** | 23.17 | 0.797 | 0.270 | 84.1 | 70.5 |
| **Ours (cov.)** | 24.89 | 0.837 | 0.218 | 90.2 | 81.2 |
| **Ours (sem.)** | 25.34 | 0.843 | 0.207 | **91.9** | 81.8 |
| **Ours (obs.)** | **25.56** | **0.846** | **0.203** | **91.8** | **83.2** |
| **Ours (view.)** | 25.17 | 0.842 | 0.211 | 91.3 | 82.0 |

**TABLE III: Rendering performance** on uniformly sampled viewpoints of the full scene after training on a single trajectory.

| Policy | Occupancy | | | Semantics | | |
|---|---|---|---|---|---|---|
| | Acc. ↑ | Prec. ↑ | Rec. ↑ | Acc ↑ | Prec. ↑ | Rec. ↑ |
| **Frontier** | 81.2 | 86.9 | 49.9 | 99.7 | 26.6 | 21.0 |
| **E2E (cov.)** | 77.1 | 86.2 | 50.4 | 99.7 | 22.1 | 16.1 |
| **E2E (cur.)** | 81.8 | 90.3 | 50.7 | 99.7 | 19.2 | 12.5 |
| **E2E (nov.)** | 83.1 | 88.7 | 61.3 | 99.7 | 25.5 | 18.3 |
| **E2E (rec.)** | 81.6 | 87.6 | 60.0 | 99.7 | 26.2 | 18.0 |
| **Ours (cov.)** | 86.8 | 89.1 | 74.7 | 99.8 | 35.1 | 27.1 |
| **Ours (sem.)** | 86.6 | 88.3 | 76.5 | 99.8 | 35.7 | 29.8 |
| **Ours (obs.)** | 86.4 | 89.4 | 76.5 | 99.8 | 36.2 | 29.8 |
| **Ours (view.)** | **88.1** | **90.9** | **77.0** | 99.8 | **37.4** | **30.2** |

**TABLE IV: Map estimation performance**: comparison of BEV maps estimated from the NeRF.

stuck because of map inaccuracies. This shows that vanilla visual coverage, the optimized metrics in many exploration-oriented tasks, is not a sufficient criterion to collect NeRF training data. FBE properly covers the scene but does not necessarily cover a large diversity of viewpoints, while the modular policy provides richer training data to the NeRF.

**End-to-end Policy vs Modular Policy —** Tables III, IV, V, VI also show that the modular policies outperform end-to-end RL policies on all considered metrics. Interestingly, *novelty* and *reconstruction* seem to be the best reward functions when training end-to-end policies if the final goal is to autonomously collect data to build a NeRF model.

**Comparing trained policies —** Rewarding modular policies with obstacles (*Ours (obs.)*) and viewpoints (*Ours (view.)*) coverage leads to the best overall performance when considering the different metrics. Explored area coverage (*Ours (cov.)*) leads to the highest *PointNav* performance, corroborating its importance for geometric tasks, whereas other semantic reward functions lead to higher *ObjectNav* performance, allowing a better semantic scene understanding.

**Semantics from Mask R-CNN —** Table VII shows the impact of using Mask R-CNN to compute the semantics training data of the NeRF model vs semantics from simulation. As expected, performance drops because Mask R-CNN provides a much noisier training signal, which could partly be explained by the visual domain gap between the real world and simulators. However, performance on the different downstream tasks is still reasonable, showing that one could autonomously collect data and generate semantics training signal without requiring additional labeling.

**Navigating with sensor and actuation noise —** So far, we have followed task specifications in [4], among which are perfect odometry and actuation. We thus conduct an

| Policy | PointGoal | | ObjectGoal | |
|---|---|---|---|---|
| | Succ. ↑ | SPL ↑ | Succ. ↑ | SPL ↑ |
| **Frontier** | 22.4 | 21.4 | 9.6 | 9.1 |
| **E2E (cov.)** | 30.0 | 29.3 | 8.9 | 8.3 |
| **E2E (cur.)** | 29.8 | 29.2 | 8.5 | 8.0 |
| **E2E (nov.)** | 32.3 | 31.9 | 11.4 | 10.8 |
| **E2E (rec.)** | 32.8 | 32.6 | 10.5 | 10.0 |
| **Ours (cov.)** | **39.5** | **39.0** | 14.8 | 14.3 |
| **Ours (sem.)** | 37.7 | 37.4 | **16.0** | **15.4** |
| **Ours (obs.)** | 38.2 | 37.8 | **15.8** | **15.3** |
| **Ours (view.)** | 39.0 | 38.6 | **15.9** | **15.3** |

**TABLE V: Planning performance** on the BEV maps estimated from the NeRF obtained with the *Fast Marching* method.

| Policy | Conv. rate ↑ | Rot. Error (°) ↓ | Trans. Error (m)↓ |
|---|---|---|---|
| **Frontier** | 7.2 | 0.383 | 0.00955 |
| **E2E (cov.)** | 15.4 | 0.319 | 0.00775 |
| **E2E (cur.)** | 12.5 | 0.325 | 0.00799 |
| **E2E (nov.)** | 19.4 | 0.315 | 0.00774 |
| **E2E (rec.)** | 19.3 | 0.292 | 0.00734 |
| **Ours (cov.)** | 20.2 | **0.283** | **0.00734** |
| **Ours (sem.)** | **23.0** | 0.319 | 0.00784 |
| **Ours (obs.)** | 22.5 | 0.305 | 0.00765 |
| **Ours (view.)** | 21.1 | 0.316 | 0.00769 |

**TABLE VI: Pose refinement performance**: optimizing camera viewpoints given a rendered target viewpoint.

| Task | Metrics | Sim. | Mask R-CNN |
|---|---|---|---|
| **Rendering** | Per-class acc. | 91.8 | 65.4 |
| | mIoU | 83.2 | 61.1 |
| **Map comparison** | Sem acc. | 99.8 | 99.7 |
| | Sem prec. | 36.2 | 14.1 |
| | Sem rec. | 29.8 | 8.5 |
| **Planning** | ObjGoal Succ. | 15.8 | 6.8 |
| | ObjGoal SPL | 15.3 | 6.5 |

**TABLE VII: NeRF semantic maps**: impact of the choice of ground-truth semantics vs. semantics estimated by Mask R-CNN when data is collected by *Ours (obs.)*.

experiment where we add noise using realistic models from [1]. We correct odometry using the pose estimation module in [1], allowing our modular policy (*Ours (obs.)*) to properly explore even under noise. We then refine camera poses with bundle adjustment before using them to train NeRF models. Tables VIII, IX, X, XI show that performance obviously decreases when adding noise, but is still satisfying in all metrics. It is important to note that training NeRF models on noisy poses is still considered an active and challenging problem [41]. Better performance will come from new techniques making NeRF models more robust, which is orthogonal to the contribution in this work.

### *D. Qualitative results on robotics downstream tasks*

**BEV maps —** Figure 6 gives examples of the BEV maps generated from the continuous representation: structural details and dense semantic information are nicely recovered (Left). Planned trajectories are close to the shortest paths, for both PointGoal tasks (Middle) and ObjectGoal (Right).

**Semantic rendering —** Figure 7 compares the segmentation maps and RGB frames rendered with the continuous representation (trained with semantic masks from simulation) to the GT maps from the simulator. Again, the structure of the objects and even fine details are well recovered, and only very local noise is visible in certain areas.

| | | RGB | | | Semantics | |
|---|---|---|---|---|---|---|
| **Policy** | **Noise** | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Per-class acc ↑ | mIoU ↑ |
| **Ours (obs.)** | – | 25.56 | 0.846 | 0.203 | 91.8 | 83.2 |
| **Ours (obs.)** | ✓ | 20.87 | 0.761 | 0.264 | 85.4 | 73.6 |

**TABLE VIII: Rendering performance with sensor and actuation noise** – *Ours (obs)*.

| | | Occupancy | | | Semantics | | |
|---|---|---|---|---|---|---|---|
| **Policy** | **Noise** | Acc. ↑ | Prec. ↑ | Rec. ↑ | Acc ↑ | Prec. ↑ | Rec. ↑ |
| **Ours (obs.)** | – | 86.4 | 89.4 | 76.5 | 99.8 | 36.2 | 29.8 |
| **Ours (obs.)** | ✓ | 86.8 | 89.8 | 69.6 | 99.7 | 29.9 | 24.1 |

**TABLE IX: Map estimation performance with sensor and actuation noise** – *Ours (obs)*.

## VI. Conclusion

This work introduces a task involving navigating in a 3D environment to collect NeRF training data. We show that RL-trained modular policies outperform classic Frontier Exploration and other end-to-end RL baselines, and compare different training reward functions. We also suggest evaluating NeRFs from a scene-understanding point of view and with robotics-oriented tasks: rendering, BEV map generation, planning, and camera pose refinement. Finally, we show that it is possible with the considered method to reconstruct house-scale scenes, even allowing safe scene-specific fine-tuning of another policy in simulation.

**Acknowledgement** — This work has been partially done as a PhD internship at Meta AI and co-financed by the ANR AI-chair grant "Remember" (ANR-20-CHIA-0018).

| | | PointGoal | | ObjectGoal | |
|---|---|---|---|---|---|
| **Policy** | **Noise** | Succ. ↑ | SPL ↑ | Succ. ↑ | SPL ↑ |
| **Ours (obs.)** | – | 38.2 | 37.8 | 15.8 | 15.3 |
| **Ours (obs.)** | ✓ | 34.5 | 33.8 | 12.9 | 12.4 |

**TABLE X: Planning performance with sensor and actuation noise** – *Ours (obs)*.

| **Policy** | **Noise** | Conv. rate ↑ | Rot. Error (°) ↓ | Trans. Error (m)↓ |
|---|---|---|---|---|
| **Ours (obs.)** | – | 22.5 | 0.305 | 0.00765 |
| **Ours (obs.)** | ✓ | 7.9 | 0.405 | 0.01125 |

**TABLE XI: Pose refinement performance with sensor and actuation noise** – *Ours (obs)*.

## References


[1] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *ICLR*, 2020.

[2] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," in *ICLR*, 2019.

[3] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman, "An exploration of embodied visual exploration," *IJCV*, 2021.

[4] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal nav. using goal-oriented semantic exploration," in *NeurIPS*, 2020.

[5] P. Marza, L. Matignon, O. Simonin, and C. Wolf, "Teaching agents how to map: Spatial reasoning for multi-object nav." in *IROS*, 2022.

[6] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *CVPR*, 2022.

[7] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *CVPR*, 2020.

[8] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta, "No rl, no simulation: Learning to navigate without navigating," in *NeurIPS*, 2021.

[9] S. Y. Min, D. S. Chaplot, P. Ravikumar, Y. Bisk, and R. Salakhutdinov, "Film: Following instructions in language with modular methods," in *ICLR*, 2022.

[10] D. S. Chaplot, M. Dalal, S. Gupta, J. Malik, and R. R. Salakhutdinov, "Seal: Self-supervised embodied active learning using exploration and 3d consistency," in *NeurIPS*, 2021.

[11] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Science Robotics*, 2022.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[13] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ToG*, 2022.

[14] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *CVPR*, 2021.

[15] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison, "In-place scene labelling and understanding with implicit scene repr." in *ICCV*, 2021.

[16] S. Zhi, E. Sucar, A. Mouton, I. Haughton, T. Laidlow, and A. J. Davison, "ilabel: Revealing objects in neural fields," *RAL*, 2022.

[17] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstr. in function space," in *CVPR*, 2019.

[18] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.

[19] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *ICCV*, 2021.

[20] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *CVPR*, 2022.

[21] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys, "Nicer-slam: Neural implicit scene encoding for rgb slam," *arXiv*, 2023.

[22] P. Marza, L. Matignon, O. Simonin, and C. Wolf, "Multi-object nav. with dynamically learned neural implicit repr." in *ICCV*, 2023.

[23] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *RA-L*, 2022.

[24] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "inerf: Inverting neural radiance fields for pose estimation," in *IROS*, 2021.

[25] X. Pan, Z. Lai, S. Song, and G. Huang, "Activenerf: Learning where to see with uncertainty estimation," in *ECCV*, 2022.

[26] H. Zhan, J. Zheng, Y. Xu, I. Reid, and H. Rezatofighi, "Activermap: Radiance field for active mapping and planning," *arXiv*, 2022.

[27] J. Zeng, Y. Li, Y. Ran, S. Li, F. Gao, L. Li, S. He, J. Chen, and Q. Ye, "Efficient view path planning for autonomous implicit reconstruction," in *ICRA*, 2023.

[28] Z. Yan, H. Yang, and H. Zha, "Active neural mapping," in *ICCV*, 2023.

[29] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *CIRA*, 1997.

[30] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," *Advanced Robotics*, 2013.

[31] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang, "Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields," *TOG*, 2017.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv*, 2017.

[34] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi *et al.*, "Nerfstudio: A modular framework for neural radiance field development," *arXiv*, 2023.

[35] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," in *ICCV*, 2019.

[36] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[37] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *arXiv*, 2018.

[38] E. Marchand, "Direct visual servoing in the freq. domain," *RA-L*, 2020.

[39] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *CVPR*, 2018.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[41] P. Truong, M.-J. Rakotosaona, F. Manhardt, and F. Tombari, "Sparf: Neural radiance fields from sparse and noisy poses," in *CVPR*, 2023.