Multi-Object Navigation with dynamically learned neural implicit representations

Pierre Marza [1]   Laetitia Matignon[2]   Olivier Simonin[1]   Christian Wolf[3]
[1]INSA Lyon   [2]UCBL   [3]Naver Labs Europe
{pierre.marza, olivier.simonin}@insa-lyon.fr
laetitia.matignon@univ-lyon1.fr, christian.wolf@naverlabs.com
Project Page: https://pierremarza.github.io/projects/dynamic_implicit_representations/

Abstract

Understanding and mapping a new environment are core abilities of any autonomously navigating agent. While classical robotics usually estimates maps in a stand-alone manner with SLAM variants, which maintain a topological or metric representation, end-to-end learning of navigation keeps some form of memory in a neural network. Networks are typically imbued with inductive biases, which can range from vectorial representations to birds-eye metric tensors or topological structures. In this work, we propose to structure neural networks with two neural implicit representations, which are learned dynamically during each episode and map the content of the scene: (i) the Semantic Finder predicts the position of a previously seen queried object; (ii) the Occupancy and Exploration Implicit Representation encapsulates information about explored area and obstacles, and is queried with a novel global read mechanism which directly maps from function space to a usable embedding space. Both representations are leveraged by an agent trained with Reinforcement Learning (RL) and learned online during each episode. We evaluate the agent on Multi-Object Navigation and show the high impact of using neural implicit representations as a memory source.

1. Introduction

Autonomous navigation in complex unknown 3D environments from visual observations requires building a suitable representation of the environment, in particular when the targeted navigation task requires high-level reasoning. Whereas classical robotics builds these representations explicitly through reconstructions, possibly supported through machine learning, end-to-end training learns them automatically either from reward, by imitation learning or through self-supervised objectives.

While spatial representations can emerge even in unstructured agents, as shown in the form of grid-cells in artificial [16, 3] and biological agents [25], spatial inductive biases can support learning actionable spatial representations and decrease sample complexity. Popular inductive biases are metric maps [44, 5, 26], topological maps [6, 12] and recently, self-attention, adapting transformers [55] to sequential decision making and navigation [20, 18, 13, 47]. The chosen representation should support robust estimation of navigable space even in difficult conditions, mapping fea-

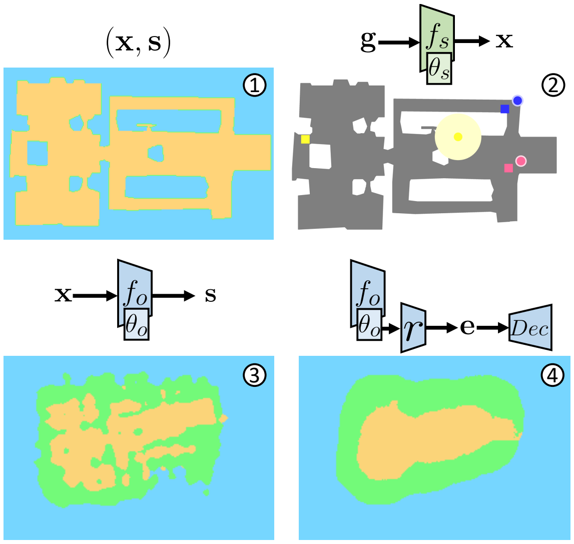

Figure 1. We propose **two implicit representations** as inductive biases for autonomous agents — both are learned online during each episode. ② a semantic representation f_s predicts positions $\mathbf{x}$ from goals $\mathbf{g}$ given as semantic codes. We show Ground Truth object positions (rectangles) and predictions (round; radius shows uncertainty, unit-less, as an illustration). Blue and pink objects have been observed, but not the yellow target. ③ A structural representation f_o predicts occupancy and exploration $\mathbf{s}$ from positions $\mathbf{x}$; we provide a global read which directly maps from function space f_o (represented by trainable weights θ_o) to a context embedding $\mathbf{e}$ used by the agent. ④ shows the reconstruction produced by a decoder Dec during training. Orange=navigable, Green=Obstacles, Blue=Unexplored. ① a ground-truth map is shown for reference, simulating a fully explored scene.

tures and objects of interest, as well as querying and reusing this information at a later time. The representation should be as detailed as required, span the full (observed) scene, easy to query, and efficient to read and write to, in particular when training is done in large-scale simulations.

Our work builds on neural fields and implicit representations, a category of models which represent the scene geometry, and eventually the semantics, by the weights of a trained neural network [57]. They have the advantage of avoiding the explicit choice of scene representation (e.g. volume, surface, point cloud etc.) and inherently benefit from the generalization abilities of deep networks to interpolate and complete unobserved information. Implicit representations have demonstrated impressive capabilities in novel view synthesis [40, 51], and have potential as a competitive representation for robotics [42, 32, 52, 1]. Their continuous nature allows them to handle level of detail efficiently through a budget given as the amount of trainable weights. This allows to span large environments without the need of discretizing the environment and handling growing maps.

We explore and study the potential of implicit representations as inductive biases for visual navigation. Similar to recent work in implicit SLAM [52], our representations are dynamically learned in each episode. Going beyond, we exploit the representation dynamically in one of the most challenging visual navigation tasks, *Multi-Object Navigation* [56]. We introduce two complementary representations, namely a query-able *Semantic Finder* trained to predict the scene coordinates of an object of interest specified as input, and an *Occupancy and Exploration Implicit Representation*, which maps 2D coordinates to occupancy information, see Figure 1. We address the issue of the efficiency of querying an implicit representation globally by introducing a new global read mechanism, which directly maps from function space, represented through its trainable parameters, to an embedding summarizing the current status of occupancy and exploration information, useful for navigation. Invariance w.r.t. reparametrization of the queried network is favored (but not enforced) through a transformer based solution. Our method does *not* require previous rollouts on the scene for pre-training or building a representation.

Our work targets a fundamental aspect of visual and semantic navigation, the mapping of key objects of interest. *MultiON* is currently one of the few benchmarks which evaluates it. As argued in previous literature [5, 56], only sequential tasks, where objects have to be found in a given order, allow object level mapping to emerge directly from reward. This follows from the observation that an agent trained to find and retrieve a single object per episode (from reward) is not required to map seen target objects, as observing them directly leads to a reactive motion towards them.

Our contributions can be summarized as follows: (i) We propose two implicit representations for semantic, occupancy and exploration information, which are trained online during each episode; (ii) We introduce a new global read procedure which can extract summarizing context information directly from the function itself; (iii) We show that the representations obtain performance gains compared to classical neural agents; (iv) We evaluate and analyze key design choices, the representation's scaling laws and its capabilities of lifelong learning.

2. Related Work

Visual Navigation — is a rich problem that involves perception, mapping and decision making, with required capacities being highly dependent on the specific task. A summary of reasoning in navigation has been given in [2], differentiating, for instance, between waypoint navigation (*Pointgoal*) [2] or finding objects of semantic categories (*ObjectGoal*) [2]. More recent tasks have been explicitely designed to evaluate and encourage mapping objects of interest during navigation itself [4, 56]. They are of sequential nature and use external objects, which are not part of the scanned 3D scenes but randomly placed. In this work we address *Multi-Object Navigation (MultiON)* [56].

Mapping and Representations — Classical methods often rely on SLAM [8, 34] which has been proposed in different variants (2D or 3D metric, topological, eventually with semantics) and observations (LIDAR, visual etc.). The objective is to integrate observations and odometry estimates over a trajectory and reconstruct the scene. Differentiable variants have been proposed recently [28, 30]. Mapping can also be discovered through interactions by a blind agent [7]. Visual Navigation can also be framed as an end-to-end learning problem, where representations are learned automatically from different signals, in particular RL. Memory can take the form of vectorial representations in recurrent units [41, 27, 62], with hybrid variants including mapping [12, 48, 17]. Recent work tends to augment agents with more structured memories. Examples are spatial metric tensors, which can contain occupancy [11], semantics [10] or be fully latent, effectively corresponding to inductive biases of the neural agents [44, 5, 26]. Other alternatives are topological maps [6, 12] or self-attention and transformers [55] adapted to navigation [20, 18, 13, 47].

Implicit representations — were initially targeting 3D reconstruction [39, 45, 14]. The core idea is to replace the need for discretizing 3D space into voxels [38], 3D points [19] or meshes [23], by an implicit representation of the 3D structure of the scene through the parameters of a learned neural network. Recent work [40, 51] achieved state-of-the-art performance on novel view synthesis with neural implicit representations. The NeRF paper introduced a differentiable volume rendering loss allowing to supervise 3D scene reconstruction from only 2D supervision [40]. For a more detailed overview of recent advances in the rapidly growing field, we refer the reader to [57].

Implicit representations in robotics — are a recent phenomenon, used to represent density [1] or to perform visuomotor control [33]. Related to goal-oriented navigation, some work targets SLAM with neural implicit representations [52], follow-up adding semantics [59], learned from sparse semantic annotations of the scene. [60] is also built on top of [52] and allows a user to interactively provide semantic annotation for the implicit representation to be trained on in real time. [63] proposes a hierarchical implicit representation of a scene to scale to larger environments and obtain a more detailed reconstruction. [15] combines feature-based SLAM and NeRF. Our work goes beyond implicit SLAM and does not stop at reconstructing a scene. We not only build implicit representations dynamically during the episode, we also use them in a down-stream navigation task *without* requiring any initial rollout for pre-training or building a representation. We also combine two different implicit representations targeting semantics vs. scene structure.

Analyzing the neural network function space — implicit representations are instances of function spaces, which are represented through their trainable parameters. Previous work performed analyses by predicting accuracy from network weights [54, 35, 36] or the generality gap between train and test performance from hidden activations [29, 58]. A direction pioneered by Hypernetworks [24] directly predict the network weights. Recently, [61] generate the weights of a CNN from support samples in the context of few-shot learning. More related to our work, [43] learns to predict the weights of an implicit representation based on external factors in the context of spatio-temporal dynamics encoding. In this work, we learn a direct mapping between an implicit representation, represented by its weights, to an actionable embedding summarizing the scene globally.

3. Navigating with implicit representations

We target the *Multi-Object Navigation* task [56], which requires an agent to navigate in a photo realistic 3D environment from RGB-D observations $\mathbf{o}_t$ and reach a sequence of target objects (colored cylinders) in a particular order. Goal categories $\mathbf{g}_t$ are given at each time step t. In an RL setting, the agent receives positive reward for each successfully reached object as well as when the geodesic distance towards the goal decreases, and a small negative reward for each step, favoring short paths.

We follow and augment a base end-to-end architecture used in many recent RL approaches, including [56, 37, 41], with the RGB-D observation, class of the current target and previous action as input to the agent. Temporal information is aggregated with a GRU unit whose output is fed to actor and critic heads. We equip this agent with two implicit representations, trained to hold and map essential information necessary for navigation: the positions of different objects of interest, and occupancy / exploration information, as shown in Figure 1,

- f_s The goal of the *Semantic Finder* $f_s(.;\theta_s)$ parameterized by trainable weights θ_s is to predict the absolute position of an object as $\mathbf{x} = [\mathbf{x}_x\ \mathbf{x}_y\ \mathbf{x}_z] = f_s(\mathbf{q};\theta_s)$ specified through an input query vector $\mathbf{q}$. Uncertainty u is also estimated — see Section 3.1 for details. $\mathbf{x}$ is then converted into coordinates relative to the agent to be fed to the GRU. Compared to classical metric representations [26, 44, 12, 5], querying the location of an object can be done through a single forward pass.

- f_o The *Occupancy and Exploration Representation* $f_o(.;\theta_o)$ parameterized by trainable weights θ_o encodes information about free navigable space and obstacles. It predicts occupancy $\mathbf{s}$ as a classification problem with three classes *{Obstacle, Navigable, Unexplored}*, as $\mathbf{s} = f_o(\phi;\theta_o)$, where ϕ is a position feature vector encoded from coordinates $\mathbf{x}$ — see Section 3.2 for details.

The *Occupancy and Exploration Representation* can in principle be queried directly for a single position, but reading out information over a large area directly this way would require multiple reads. We propose to compress this procedure by providing a trainable global read operation $r(.;\theta_g)$, which predicts an embedding $\mathbf{e}$ containing a global context about what has already been explored, and positions of navigable space. The prediction is done directly from the trainable parameters of the implicit representation, as $\mathbf{e} = r(\theta_o;\theta_r)$. Here θ_o is input to r, whereas θ_r are its parameters.

Given representations f_s and f_o, a single forward pass of the agent at time step t and for a goal g_t involves reading the representations and providing the input to the policy. The current RGB-D observation o_t is also encoded by the convolutional network c (different from the projection module p used to generate samples for training the *Semantic Finder*). Previous action a_{t-1} and current goal g_t are passed through embedding layers, named $L(.)$ in the following equations. These different outputs are fed to the policy,

$$\mathbf{x}_t = f_s(g_t;\theta_{s,t}),\ \mathbf{e}_t = r(\theta_{o,t};\theta_r),\ \mathbf{c}_t = c(o_t;\theta_c), \quad (1)$$

$$\mathbf{h}_t = GRU(\mathbf{h}_{t-1}, \mathbf{x}_t, u_t, \mathbf{e}_t, L(a_{t-1}), L(g_t), \mathbf{c}_t;\theta_G), \quad (2)$$

$$a_t = \pi(\mathbf{h}_t;\theta_\pi), \quad (3)$$

where we added indices $\cdot_t$ to relevant variables to indicate time. Please note that the trainable parameters $\theta_{s,t}$ and $\theta_{o,t}$ of the two implicit representations are time dependent, as they depend on the observed scene and are updated dynamically, whereas the parameters of the policy π and the global reader r are not. Here, GRU corresponds to the update equations of a GRU network, where we omitted gates for ease of notation. The notation $a_t = \pi(.)$ predicting action a_t is

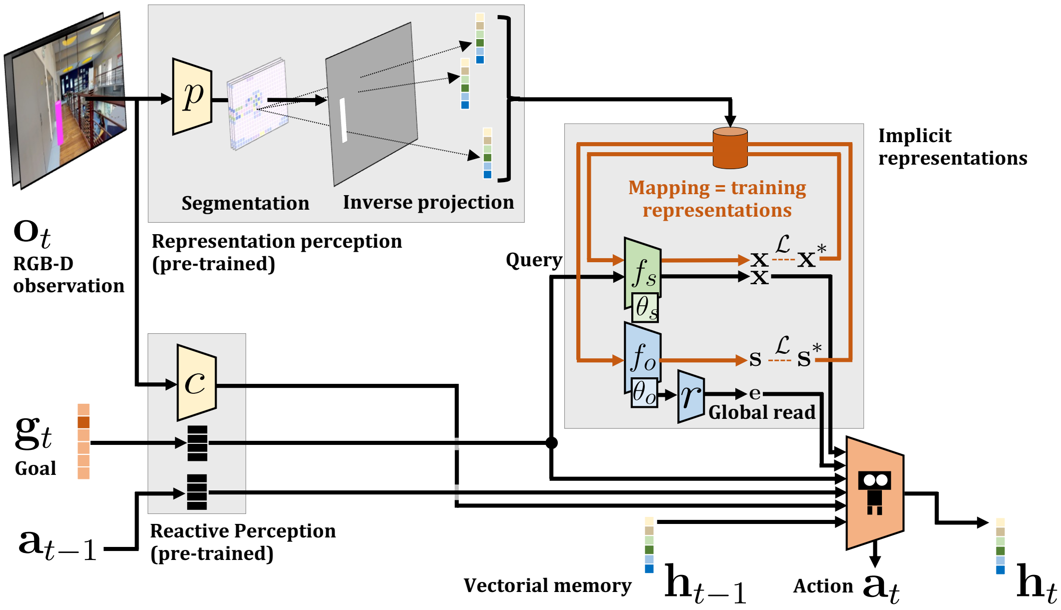

Figure 2. **Navigating with implicit representations**. Red connections → indicate the training process of the two implicit representations (=mapping), which is also done during agent deployment. Black connections → show the forward pass of the agent. ▮ are discrete learned embeddings (LUT). Policy training is not shown in this figure.

also a simplification, as we train the agent with PPO, an actor-critic method — see Section 3.4.

Mapping means training! — The implicit representations f_s and f_o maintain a compact and actionable representation of the observed scene, and as such need to be updated at each time step from the current observation o_t. Given their implicit nature and implementation as neural networks, updates are gradient based and done with SGD. The implicit representations are therefore *trained from scratch at each episode even after deployment*.

Training a representation from observations obtained sequentially during an episode also raises a serious issue of catastrophic forgetting [22], as places of the scene observed early might be forgotten later in the episode [52, 59]. We solve this by maintaining two replay buffers throughout the episode, one for each representation. Training samples are generated from each new observation and added to the replay buffers at each time step. Both representations are then trained for a number of gradient steps (n_s for the *Semantic Finder* and n_o for the *Exploration and Occupancy Representation*). Details on the two representations and their training are given in Sections 3.1 and 3.2. The global reader r is not trained or finetuned online but rather trained once offline.

3.1. The Semantic Finder f_s

While recent work on implicit representations for robotics focused on signed distance functions [42, 32], occupancy [52] or density, assuming light density approximates mass density [1], the aim of this model is to localize an object of interest within the scene, which can be seen as inverse operation to classical work. From a query vector given as input, the *Semantic Finder* predicts the position of the object, which is particularly useful for an agent interacting with an environment in the context of a goal conditioned task. It is implemented as a 3-layer MLP with ReLu activations in the intermediate layers and a sigmoid activation for the output. Hidden layers have 512 neurons. The query vector $\mathbf{q}$ corresponds to the 1-in-K encoding of the target object class, which during navigation is directly determined by the object goal g_t provided by the task.

Mapping/Training — The implicit representation is updated minimizing the L1 loss between the prediction $\mathbf{x}_i = f_s(\mathbf{q}_i, \theta_s)$ and the supervised coordinates $\mathbf{x}_i^*$ (we avoided the term "*ground-truth*" here on purpose), $\mathcal{L}_s = \sum_i \|\mathbf{x}_i^* - \mathbf{x}_i\|_1$, where the sum goes over the batch sampled from the scene replay buffer. Coordinates $\mathbf{x}_i^*$ are normalized $\in [0, 1]$.

The data pairs (x_i^*, q_i) for training are created from each observation o_t during each time step, each data point corresponding to an observed point. Pixels in o_t are inversely projected into 3D coordinates in the scene using the depth channel, the camera intrinsics, as well as agent's coordinates and heading that are assumed to be available, as in [56]. The query vector $\mathbf{q}$ corresponds to a 9-dimensional vector encoding a distribution over object classes (8 target objects and the "*background*" class). Let us recall that while the training of the representation is supervised, this supervision cannot use "ground-truth" information available only during training. All supervision information is required to be predicted from the data available to the agent even after deployment. We predict object class information through a semantic segmentation model p, which is applied to each current RGB-D observation $o_t \in \mathbb{R}^{h\times w\times 4}$, recovering the output segmentation map $m_t \in \mathbb{R}^{k\times l\times 9}$. The model has been pre-trained on the segmentation of the different target objects, i.e. coloured cylinders, and is not fine-tuned during training of the agent itself.

Training data pairs (x_i^*, q_i) are sampled from this output. The supervised coordinates x_i^* are simply the mean 3D coordinates of each feature map cell, after inverse projection. The query vector $\mathbf{q}_i$ is the distribution over semantic classes. After the replay buffer is updated, a training batch must be sampled to update the neural field. One fourth of the samples in the batch of size b correspond to the $b/4$ last steps. The rest are sampled from the previous steps in the replay buffer. Uniform sampling is also performed among pairs collected at a given time step.

Estimating uncertainty — is an essential component, as querying yet unseen objects will lead to wrong predictions, which the agent needs to recognize as such, and discard. The estimation of uncertainty in neural networks is an open problem, which has been previously addressed through different means, including drop out as a Bayesian approximation [21], variational information bottlenecks [50], density estimation [31], and others. In this work, we approximate a density estimate in the scene replay buffer by calculating the minimum Euclidean distance between the input query and all embeddings in the replay buffer at the current time step. The method is simple and efficient and does not require explicitly fitting a model to estimate the marginal distribution $p(\mathbf{q})$, in particular as the uncertainty representation is latent, can be un-normalized as not required to be a probability.

3.2. Occupancy and Exploration Implicit Representation f_o

Unlike f_s, the occupancy representation f_o is closer to classical implicit representations in robotics, e.g. [52, 42, 32, 1], which map spatial coordinates to variables encoding information on navigable area like occupancy or signed distances. Different to previous work, our representation also includes exploration information, which changes over time during the episode. Once explored, a position changes its class, which makes our neural field *dynamic*. Another difference with f_s is that the latter deals with 3D coordinates while f_o is a topdown 2D representation. Inspired by [57, 53], the model uses Fourier features ϕ extracted from the 2D coordinates $\mathbf{x}$ previously normalized $\in [0, 1]$,

$$\phi = (\cos(\mathbf{x}2^0), \sin(\mathbf{x}2^0), ..., \cos(\mathbf{x}2^{\frac{p}{4}}), \sin(\mathbf{x}2^{\frac{p}{4}})). \quad (4)$$

The network f_o is a 3-layer MLP with ReLu intermediate activations and a softmax function at the output layer. Hidden layers have 512 neurons, and $p = 40$.

Mapping/Training — The implicit representation is updated minimizing the Cross Entropy loss between the prediction $\mathbf{s}$ of the neural field and the supervised label $\mathbf{s}^*$ of three classes *{Obstacle, Navigable, Unexplored*}, as $\mathcal{L}_o = -\sum_{c=1}^{3} \mathbf{s}^*_\mathbf{c} \log \mathbf{s}_c$. As for the *Semantic Finder*, training data pairs $(\mathbf{s}^*, \mathbf{s})$ are created through inverse perspective projection of the pixels of the observation o_t into 3D scene coordinates. Thresholding the z (height) coordinate decides between *Navigable* and *Obstacle* classes. Points with a z coordinate higher than a certain threshold are discarded. The replay buffer is balanced between both classes, and only samples of the last 1000 steps are kept. Samples of the *Unexplored* class are not stored.

The replay buffer is sampled similarly to the one for the *Semantic Finder*. However, additional samples for the *Unexplored* class are created by sampling uniformly inside the scene, for speed reasons simply ignoring conflicts with explored areas and treating them as noisy labels.

3.3. Global Occupancy Read r — handling reparametrization invariance

The global Occupancy reader r allows to query the occupancy information of the scene globally, beyond point-wise information, and as such is a trainable mapping from the space of functions $f_o(.;.)$ to an embedding space $\mathbf{e}$. In particular, two functions f_o and f'_o s.t. $f_o(\mathbf{x}) = f'_o(\mathbf{x})\ \forall \mathbf{x}$ should be mapped to identical or close embeddings. However, as the occupancy networks f_o are implemented as MLPs, any given instance $f_o(.;\theta_o)$ parameterized by trainable weights θ_o can be reparametrized by any permutation of hidden units, which leads to permutations of the rows and columns, respectively, of two weight matrices, its own and the one of the preceding layer. This reparameterization keeps the represented functions identical, although their representations as weight vectors are different.

To favor learning a global occupancy reader which is invariant w.r.t these transformations, we implement it as a transformer model with self-attention [55] — this, however, does not enforce full invariance. The model takes as input a sequence of tokens $(w_1, ..., w_N)$, where $w_i \in \mathbb{R}^a$ is a learned linear embedding of the incoming weights of one neuron within the implicit representation f_o, and N is the number of neurons of f_o. Each token is summed with a positional encoding in the form of Fourier features. An additional "*CLS*" token with learned embedding is concatenated to the input sequence. The reader is composed of 4 self-attention layers, with 8 attention heads. The output representation of the "*CLS*" token is used as the global embedding of the implicit representation.

Training — The global reader r is trained with full supervision from a dataset of $25k$ trajectories composed of MLP weights $\theta_{o,i}$ and absolute maps $\mathbf{M}_i$, $i..1..25k$. Each map is a metric tensor providing occupancy information extracted from the corresponding implicit representation, i.e. $\mathbf{M}_i(\mathbf{x}_y, \mathbf{x}_x) = f_o(\mathbf{x}, \theta_{o,i})$. The dataset also contains an ego-centric version $\mathbf{M}'_i$ of each map, which is centered on the agent and oriented depending on its current heading. The reader r is trained in an Encoder-Decoder fashion, where r plays the role of the encoder,

$$\mathbf{e}_i = r(\theta_{o,i}), \quad \hat{\mathbf{M}}_i = Dec(e_i, p_i), \tag{5}$$

where p_i is the agent pose (position and heading), necessary to decode ego-centric information. We minimize a cross entropy loss on the prediction of ego-centric maps,

$$\mathcal{L}_g = -\sum_i \sum_k \sum_l \sum_{c=1}^3 \mathbf{M}'^{*}_{i,c}(k,l) \log \mathbf{M}'_{i,c}(k,l) \tag{6}$$

Directly training this prediction proved to be difficult. We propose a procedure involving several steps, detailed in section 3.5. After the training phase, the reader g is used in the perception + mapping module of the agent as given in equation (1), and kept frozen during agent RL training.

3.4. Training the Agent

The agent is trained with RL, more precisely Proximal Policy Optimization (PPO) [49]. The inner training loops of the implicit representations are supervised (**red arrows** in Figure 2) and occur at each time step in the forward pass, whereas the RL-based outer training loop of the agent occur after N acting steps (**black arrows** in Figure 2). As the perception module used to generate training data from RGBD-observations for the *Semantic Finder* f_s is independent of the visual encoder c in the agent (see Figure 2), and as its query $\mathbf{q}_t$ is fixed to the navigation goal $\mathbf{g}_t$, there is no need to track the weights θ_s at each time step in order to backpropagate the PPO loss (outer training). This is a key design choice of our method.

Training assumptions — We do not rely on the existence of global GT maps for occupancy, as $f_o()$ and $r()$ were trained on observation data from agent trajectories only. However, similar to [37, 46], we exploit object positions in simulation, during training only; Moreover, we require pixel-wise segmentation masks during training. We believe that this does not change requirements, as the goal is to fully exploit 3D photo-realistic simulators as a data source and see how far the field can go with this. Generalization requirements are unchanged: we require our agent to be able to generalize to new unseen scenes. Generalization to unseen object categories is not targeted, and in *MultiON* task setup not possible for any agent, as object positions and labels are required for reward calculation.

3.5. Training the global reader

The proposed training procedure for the global reader r (performed *before* training the agent) can be split into 3 phases. The architecture for the convolutional decoder $Dec()$ is kept the same in all of them. The hyperparameters of its different layers are detailed in the Supplementary Material. It is composed of 6 transpose convolution layers along with batch norm layers and ReLU activations, except for the last layer with a softmax activation. Figure 3 provides an overview of the 3 steps involved in the training of the global reader.

Fully convolutional autoencoder — The first step is to train a fully convolutional autoencoder on the set of absolute maps $\mathbf{M}_i$, $i..1..25k$. Only the decoder weights are kept.

Global reader training on absolute maps — The second step consists in training the global reader to output embeddings fed to the frozen decoder from the previous step. The objective is to reconstruct absolute maps from the weights of the implicit representation. The global reader weights are kept after this training phase.

Global reader finetuning on egocentric maps — The global reader, whose weights are initialized from the weights obtained in the previous step, is now trained along with the same decoder from the first step (also used in the second step) on the set of egocentric maps $\mathbf{M}'_i$, $i..1..25k$. Both global reader and decoder are finetuned. The output of the global reader is not directly fed to the decoder, but is passed through linear layers in order to fuse information about first the position of the agent, and then its heading because this time the right operations of shift and rotation must be applied in order to reconstruct egocentric maps.

Integrating the pre-trained global reader into the agent — After this pre-training in 3 steps, the weights of the global reader are frozen and not updated during the RL training. However, a linear layer is learnt to project the 576-dim embedding from the global reader into a 256-dim representation fed to the GRU. This linear layer is trained from reward signals.

4. Experiments

***MultiON* task** — we target the 3-ON version of the *MultiON* task [56], where the agent deals with sequences of 3 objects, each belonging to one of 8 classes (cylinders of different colors). At each step, the observation $\mathbf{o}_t$ is an RGB-D image of size 256×256×4 and the target class is a one-in-K (K=8) vector. The action space is discrete: {*Move forward* $0.25m$, *Turn left* 30°, *Turn right* 30°, *Found*}. An episode is considered successful if the agent finds all of the targets before the time limit ($2,500$ environment steps), and chooses the *Found* action for each one at a distance closer to

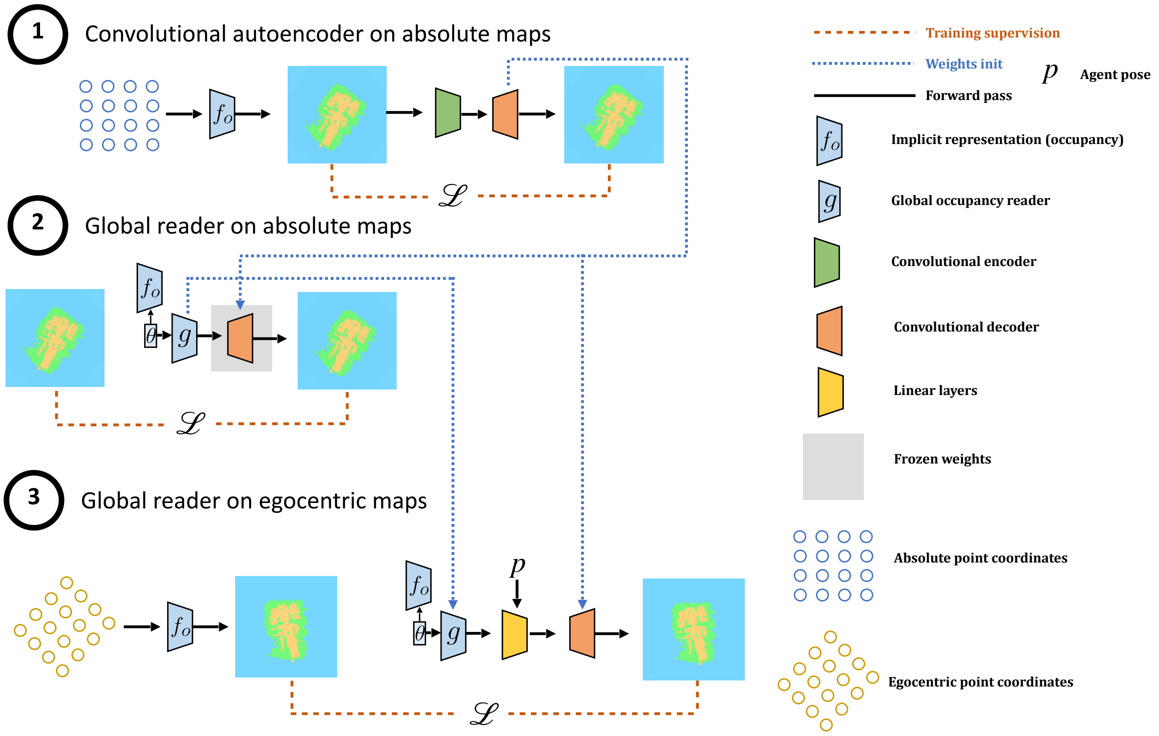

Figure 3. **Training the Global Reader** in 3 steps: ① training a convolutional auto-encoder on absolute maps; ② extension to inputs equal to weights θ_o = the gobal reader; ③ extention to predicting ego-centric maps, better suited to navigation problems.

	0 – 30		30 – 50		50 – 70		— Val —				— Test —			
	S	O	S	O	S	O	**Success**	**Progress**	**SPL**	**PPL**	**Success**	**Progress**	**SPL**	**PPL**
	–	–	–	–	–	–	33.2± 1.2	49.0± 1.1	21.2± 0.5	31.6± 1.2	42.3± 1.5	56.7± 0.9	28.1± 1.0	37.8± 1.8
μ	–	–	✓	–	✓	–	37.8± 1.6	52.3± 0.9	26.35± 1.5	36.5± 0.8	**47.0**± 1.7	**60.5**± 1.6	34.5± 0.8	44.2± 1.0
	–	–	✓	–	✓	✓	**38.5**± 4.6	**52.5**± 4.8	**28.2**± 2.1	**38.3**± 1.7	46.7 ± 3.0	60.1 ± 3.1	**35.1** ± 1.4	**44.8** ± 1.0
	–	–	–	–	–	–	32.1	47.7	21.6	32.6	41.0	55.9	28.9	39.0
↑	–	–	✓	–	✓	–	38.1	51.9	27.3	37.1	48.6	61.5	35.2	44.2
	–	–	✓	–	✓	✓	**43.1**	**56.8**	**30.5**	**40.1**	**49.7**	**63.4**	**36.4**	**45.9**

Table 1. **Impact of the implicit representations:** navigation performance on *MultiON val* and *MultiON test*. S=f_s activated, O=f_o activated in the corresponding training period (see text). Top/μ: means over 3 runs; Bottom/↑: best validation seeds over 3 runs.

$1.5m$. Calling *Found* incorrectly terminates episode as a failure. An access to perfect odometry information (localization and heading) was assumed in [56] as the standard protocol.

Dataset and metrics — The agent is trained on the Matterport3d [9] dataset. We followed the standard train/val/test split over scenes (denoted *MultiON train*, *MultiON val*, *MultiON test*): 61 training, 11 validation and 18 test scenes. The train, val and test splits are respectively composed of $50,000$, $12,500$ and $12,500$ episodes per scene. Reported results on the val and test sets (Tables 1 and 2) were computed on a subset of $1,000$ randomly sampled episodes. We report standard metrics as used in the navigation literature (and in [56]): *Success*: percentage of successful episodes — all objects are reached respecting order, time; *Progress*: percentage of objects successfully found (respecting order, time); *SPL*: Success weighted by Path Length, extending the original SPL metric [2] to *MultiON*; *PPL*: Progress weighted By Path Length (the official *MultiON* challenge metric).

***Global reader* dataset** — The *Global reader* r was trained on a dataset of $25k$ trajectories obtained from rollouts performed by a baseline agent [37]. 95% was used for training and the rest for validation. On these trajectories we first trained the occupancy representation f_o "*in-situ*", i.e. as if it were deployed on the agent, and we recorded training samples i for training the reader r: pairs of network weights $\theta_{o,i}$ and associated maps M_i obtained by iteratively querying the implicit representation. Ego-centric maps were generated from the absolute ones and both were cropped around their center.

Perception module dataset — The perception module

	Agent	ρ	α	γ	**Success**	**Progress**	**SPL**	**PPL**	AUX	ORC
(a)	OracleMap† [56]	–	✓		50.4± 3.5	60.5± 3.1	40.7± 2.2	48.8± 1.9	–	✓
(b)	OracleEgoMap† [56]	–	✓		32.8± 5.2	47.7± 5.2	26.1± 4.5	37.6± 4.7	–	✓
(c)	NoMap† [56]	–	✓		16.7± 3.6	33.7± 3.3	13.1± 2.4	26.0± 1.7	–	–
(d)	ProjNMap† [26]	–	✓		25.9± 1.1	43.4± 1.0	18.3± 0.6	30.9± 0.7	–	–
(e)	NoMap	✓	–		42.3± 1.5	56.7± 0.9	28.1± 1.0	37.8± 1.8	–	–
(f)	ProjNMap [26]	✓	–		39.7± 2.3	55.4± 1.4	28.7± 1.1	40.1± 1.9	–	–
(g)	Implicit (Ours) *w/ curriculum w/ pre-train*	✓	–	–	46.7± 3.0	60.1± 3.1	35.1± 1.4	44.8± 1.0	–	–
(h)	ProjNMap + *AUX* [37]	N/A	✓	N/A	57.7 ± 3.7	70.2 ± 2.7	37.5 ± 2.0	45.9 ± 1.9	✓	–
(i)	Implicit (Ours) *w/o curriculum w/ pre-train + AUX*	✓	✓	✓	58.3± 0.8	69.4± 1.1	**43.8**± 1.0	**52.1**± 1.6	✓	–
(j)	Implicit (Ours) *w/o curriculum w/o pre-train*	–	✓	✓	54.8± 3.6	68.0± 3.4	41.7± 1.9	51.3± 1.6	–	–
(k)	Implicit (Ours) *w/o curriculum w/o pre-train + AUX*	–	✓	✓	57.9± 2.0	69.5± 0.6	**43.3**± 2.2	**51.9**± 3.7	✓	–

Table 2. **Comparison with SOTA methods** on *MultiON test*. †=performance taken from [37]. "*AUX*" = auxiliary losses using privileged information [37]. "*ORC*"=non-comparable, uses oracle information. ρ = pre-training of input encoders from [37]. α = finetuning of input encoders with RL. γ = implicit representations are accessible to the agent since the beginning of RL training (*w/o curriculum*).

Uncertainty	**Success**	**Progress**	**SPL**	**PPL**
–	35.4± 3.0	49.7± 3.3	29.4± 2.0	40.9± 2.4
✓	**43.4**± 3.1	**58.0**± 3.0	**35.1**± 0.8	**46.4**± 1.0

Table 3. **Uncertainty:** comparing training w/ semantic input only, no occupancy, from the beg. of training, w/ and w/o uncertainty.

p was trained to segment the different target objects. The generated dataset is composed of $132k$ pairs of RGB-D observations and segmentation masks. Samples for 4 scenes were kept as a validation set.

Training details — We use the reward function given in [56] for RL/PPO training (see supplementary material), and train all agents for $70M$ steps as in [37]. For all agents in Table 1 and some in Table 2 (*w/ pre-train*: ✓ in ρ column), the encoders (visual encoder c, as well as goal and previous action embedding layers, see Figure 2) are pre-trained with a baseline, which corresponds to the *ProjNeuralMap* agent trained with auxiliary losses [37]. This is done to faster training, as it will be shown later (in Table 2) that the same final performance can be reached without this initial pre-training of encoders. Training and evaluation hyper-parameters, as well as architecture details have been taken from [56]. All reported quantitative results are obtained after 3 training runs for each model.

Impact of the implicit representations — Table 1 shows the impact of the two implicit representations on navigation (top: means over 3 runs; bottom: best validation seeds over 3 runs). To keep compute requirements limited and decrease sample complexity, in these ablations we do not train the full agent from scratch, in particular since the early stages of training are spent on learning basic interactions. We decompose training into three phases: $0{-}30M$ steps (no implicit representations, i.e. all entries to the agent related to f_s and f_o are set to 0); $30M{-}50M$ steps (training includes the *Semantic Finder* f_s) and finally $50M{-}70M$ steps (full model). This 3-steps approach is used to train all agents in Table 1, and will be denoted as *curriculum* (See Table 2, *w/ curriculum*: – in γ column). All metrics on both val and test sets are improved, with the biggest impact provided by the *Semantic Finder*, which was expected. We conjecture that mapping object positions is a more difficult task, which is less easily delegated to the vectorial GRU representation, than occupancy. We also see an impact of the occupancy representation, which not only confirms the choice of the implicit representation f_o itself, but also its global read through $r(\theta_o)$. Training curves are given in the supplementary material.

Uncertainty — has an impact on agent performance, as we show in the ablation in Table 3. Indeed, when training an agent with the semantic input since the beginning of training (*w/o curriculum*) and no occupancy input (as the uncertainty is only related to semantic information), feeding the agent with the computed uncertainty about the output of the *Semantic Finder* brings a boost in performance.

Comparison with previous SOTA methods — is done in Table 2. The performance entries of these baselines are taken from [37], which describes the winning entry of the *CVPR 2021 MultiON* competition. Our method outperforms the different competing representations, even when they benefit from the same pre-training scheme and are thus completely comparable. NoMap with pre-training corresponds to the first row of Table 1. The difference between (g) and (i) is the use of the auxiliary tasks in [37], but also that the implicit representations are available to the agent during the whole training period for (i), i.e. no decomposition into 3 phases as in the ablations in Table 1 (*w/o curriculum*). Moreover, compared to (g), in (i) the weights of the pre-trained encoders are finetuned. We see that the gains of our representations are complementary to the auxiliary losses in [37]. (j) and (k) confirm this finding, showing that most of the gain compared with (h) comes from the implicit representations, with the auxiliary losses bringing an additional boost. (j) and (k) also show that, even though pre-training can help speed up RL training, similar test performance is achieved without it.

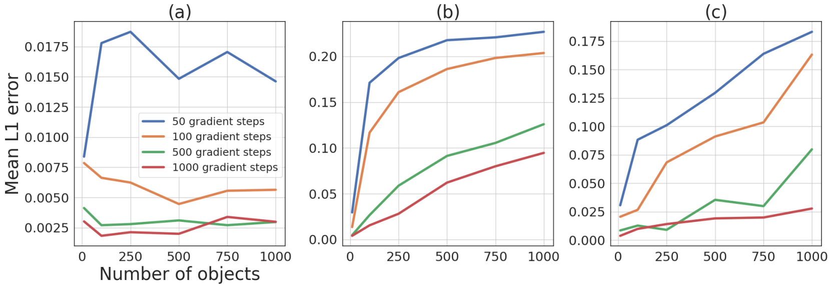

Figure 4. **Capacity of the semantic representation:** we report mean distance prediction error (normalized $\in [0, 1]$) as a function of the number of stored objects. Replay buffers are composed of dummy queries: (a) one-hot queries with same dimension as number of objects; (b) random query with dimension 9; (c) random query with same dimension as number of objects.

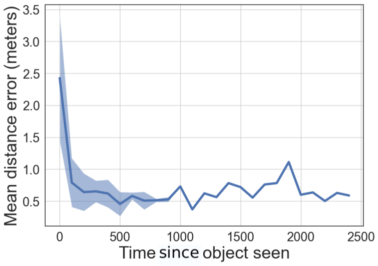

Figure 5. **Lifelong learning of the semantic repr. f_s:** we report mean error in meters, test set, 600 episodes, as a function of the number of time steps since the object was first seen in the episode (t=0). The error falls immediately and stays low over the episode.

Reconstruction performance of the Global Reader r — although the task of reconstructing egocentric maps from occupancy functions f_o (represented by θ_o) is only used as a proxy task to train the Global reader r, we see it is a reliable proxy for the quality of extracting the global latent vector $\mathbf{e}$ fed to the agent.

Accuracy	Jaccard Index
83.4	56.5

Table 4. **Performance of the global reader r:** We report accuracy and Jaccard index.

In Table 4 we report reconstruction performance measured as accuracy and mean Jaccard Index on the validation split of the dataset used to train the reader. We judge that an accuracy of 83.4% is surprisingly high, given that the global reader needs to reconstruct the content of the representation directly from its parameters θ_o, that *each implicit representation has been initialized randomly*, and that the reader is required to be invariant w.r.t. to reparameterization (see Section 4). The task is even made harder as neural weights can be considered as an absolute representation of the env. and the reader must combine it with information about the agent pose to reconstruct an egocentric map.

Capacity of the Semantic Finder — Unlike all other results presented in this paper, this experiment is performed independently of the official *MultiON* benchmark. We construct a synthetic dataset to evaluate the capacity of the *Semantic Finder*. More details about the generated data can be found in the supplementary material. As the granularity of the implicit representations is handled through the budget in terms of trainable parameters, we evaluate the capacity of the *Semantic Finder* f_s to store large numbers of objects in Figure 4: In (a), we can see that increasing the number of objects up to 1000 does not increase error with sufficient gradient updates. However, since inputs are 1-in-K, increasing the number of objects also increases capacity. In (c), we see that this does not hold for random objects (not 1-in-K), whose less structured storage requires more capacity. In (b), we keep the input dimension fixed for random objects, which further increases error.

Evaluating catastrophic forgetting — we evaluate the capacity of the *Semantic Finder* f_s to hold the learned information over the full length of the sequence in spite of the fact that it is continuously trained. Figure 5 shows the evolution of the mean error in distance for the predicted position of queried target objects as a function of time. The error quickly goes below $1.5m$ once the object has been seen the first time (t=0 in the plot), which is distance threshold required by the *MultiON* task, and stays there, providing evidence that the model does not suffer from catastrophic forgetting.

Runtime performance — inspite of requiring to continously train the representations, we achieve 45 fps during parallelized RL training, including the environment steps (simulator rendering), forward passes, representation training and RL training on a single V100 GPU. The average time of one agent forward pass, including updates of implicit representations is 20ms on a V100 GPU, which is equivalent to 50 fps, enough for real-time performance.

5. Conclusion

We introduce two implicit representations to map semantic, occupancy and exploration information. The first estimates the position of an object of interest from a vector query, while the second encapsulates information about occupancy and explored area in the current environment. We also introduce a global read directly from the trainable weights of this representation. Our experiments show that both implicit representations have a positive impact on the navigation performance of the agent. We also studied the scaling laws of the semantic representation and its behavior in the targeted lifelong learning problem. Future work will target differentiating through the implicit representations all the way back to the perception modules.

Acknowledgement — We thank ANR for support through AI-chair grant “Remember” (ANR-20-CHIA-0018).

References

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 2022. 2, 3, 4, 5

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv preprint*, 2018. 2, 7

[3] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M.J. Chadwick, T. Degris, J. Modayil, G. Wayne, H. Soyer, F. Viola, B. Zhang, R. Goroshin, N. Rabinowitz, R. Pascanu, C. Beattie, S. Petersen, A. Sadik, S. Gaffney, H. King, K. Kavukcuoglu, D. Hassabis, R. Hadsell, and D. Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557, 2018. 1

[4] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Deep reinforcement learning on a budget: 3d control and reasoning without a supercomputer. In *ICPR*, 2020. 2

[5] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Egomap: Projective mapping and structured egocentric memory for deep RL. In *ECML-PKDD*, 2020. 1, 2, 3

[6] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Learning to plan with uncertain topological maps. In *ECCV*, 2020. 1, 2

[7] G. Bono, L. Antsfeld, A. Sadek, G. Monaci, and C. Wolf. Learning with a Mole: Transferable latent spatial representations for navigation without reconstruction. *arXiv:2306.03857*, 2023. 2

[8] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2017. 2

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *I.C. on 3D Vision*, 2018. 7

[10] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *In Neural Information Processing Systems*, 2020. 2

[11] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 2

[12] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 1, 2, 3

[13] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation. 2022. 1, 2

[14] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[15] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In *ICRA*, 2023. 3

[16] C.J. Cueva and X.-X. Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. In *ICLR*, 2018. 1

[17] S. Dey, A. Sadek, G. Monaci, B. Chidlovskii, and C. Wolf. Learning whom to trust in navigation: dynamically switching between classical and neural planning. In *IROS*, 2023. 2

[18] Heming Du, Xin Yu, and Liang Zheng. VTNet: Visual Transformer Network for Object Goal Navigation. In *ICLR*, 2021. 1, 2

[19] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 2

[20] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, 2019. 1, 2

[21] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2014. 5

[22] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR*, 2014. 4

[23] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018. 2

[24] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 3

[25] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436:801–6, 09 2005. 1

[26] João F. Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *CVPR*, 2018. 1, 2, 3, 8

[27] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*, 2017. 2

[28] Krishna Murthy Jatavallabhula, Soroush Saryazdi, Ganesh Iyer, and Liam Paull. gradSLAM: Automagically differentiable SLAM. In *ICRA*, 2020. 2

[29] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018. 3

[30] Peter Karkus, Shaojun Cai, and David Hsu. Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation. In *CVPR*, 2021. 2

[31] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11), 2021. 5

[32] Xueting Li, Shalini De Mello, Xiaolong Wang, Ming-Hsuan Yang, Jan Kautz, and Sifei Liu. Learning Continuous Environment Fields via Implicit Functions. In *ICLR*, 2022. 2, 4, 5

[33] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *CoRL*, 2022. 3

[34] Iker Lluvia, Elena Lazkano, and Ander Ansuategi. Active Mapping and Robot Exploration: A Survey. *Sensors*, 21(7):2445, 2021. 2

[35] Charles H Martin and Michael W Mahoney. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *SDM*, 2020. 3

[36] Charles H Martin, Tongsu Serena Peng, and Michael W Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 2021. 3

[37] Pierre Marza, Laetitia Matignon, Olivier Simonin, and Christian Wolf. Teaching agents how to map: Spatial reasoning for multi-object navigation. In *IROS*, 2022. 3, 6, 7, 8

[38] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2

[40] 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*, 2020. 2

[41] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *ICLR*, 2017. 2, 3

[42] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. iSDF: Real-Time Neural Signed Distance Fields for Robot Perception. 2022. arXiv: 2204.02296. 2, 4, 5

[43] Shaowu Pan, Steven L Brunton, and J Nathan Kutz. Neural implicit flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data. *arXiv preprint arXiv:2204.03216*, 2022. 3

[44] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2018. 1, 2, 3

[45] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[46] A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *ICCV*, 2021. 6

[47] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent. *arXiv:2205.06175*, 2022. 1, 2

[48] A. Sadek, G. Bono, B. Chidlovskii, A. Baskurt, and C. Wolf. Multi-Object Navigation in real environments using hybrid policies. In *ICRA*, 2023. 2

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. 6

[50] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid Exploration for Open-World Navigation with Latent Goal Models. In *CORL*, 2021. 5

[51] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 2

[52] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2, 3, 4, 5

[53] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 5

[54] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020. 3

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 5

[56] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. In *NeurIPS*, 2020. 2, 3, 4, 6, 7, 8

[57] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *arXiv preprint arXiv:2111.11426*, 2021. 2, 3, 5

[58] Scott Yak, Javier Gonzalvo, and Hanna Mazzawi. Towards task and architecture-independent generalization gap predictors. *arXiv preprint arXiv:1906.01550*, 2019. 3

[59] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 3, 4

[60] Shuaifeng Zhi, Edgar Sucar, Andre Mouton, Iain Haughton, Tristan Laidlow, and Andrew J Davison. ilabel: Interactive neural scene labelling. *arXiv preprint arXiv:2111.14637*, 2021. 3

[61] Andrey Zhmoginov, Mark Sandler, and Max Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning. *arXiv preprint arXiv:2201.04182*, 2022. 3

[62] 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*, 2017. 2

[63] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. *CVPR*, 2022. 3