Scene Graph Contrastive Learning for Embodied Navigation

Kunal Pratap Singh Jordi Salvador Luca Weihs Aniruddha Kembhavi
Allen Institute for AI
{kunals, jordis, lucaw, anik}@allenai.org

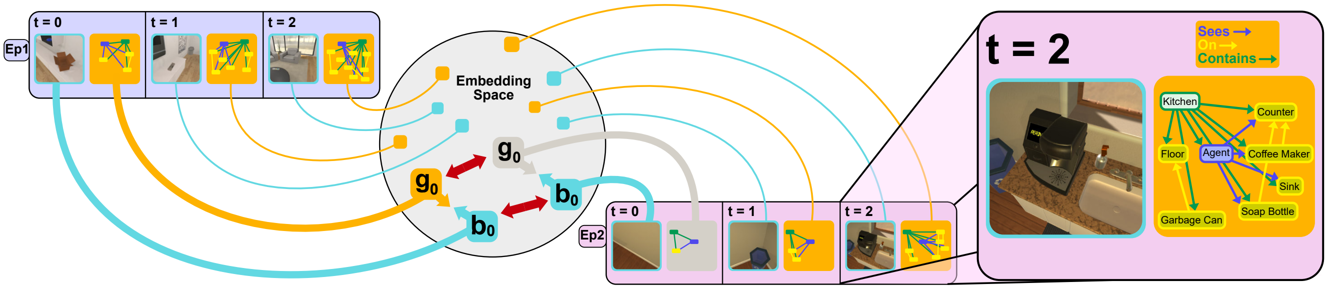


Figure 1: **Scene Graph Contrastive (SGC) Learning.** We propose to use scene graphs as an auxiliary supervisory signal for embodied agents. We iteratively build a scene graph based on the agent's observations. The agent, the current room, and the objects are represented as nodes in the graph, as shown in the magnified $t = 2$ window. Edges of the graph encode various relationships like *Sees*, *On* and *Contains*. We optimize the agent belief to be closer to the graph representation at that time step. Ep1 and Ep2 denote two different episode rollouts. g_0 and b_0 denote the belief and the graph representation at the t=0.

Abstract

Training effective embodied AI agents often involves expert imitation, specialized components such as maps, or leveraging additional sensors for depth and localization. Another approach is to use neural architectures alongside self-supervised objectives which encourage better representation learning. However, in practice, there are few guarantees that these self-supervised objectives encode task-relevant information. We propose the Scene Graph Contrastive (SGC) loss, which uses scene graphs as training-only supervisory signals. The SGC loss does away with explicit graph decoding and instead uses contrastive learning to align an agent's representation with a rich graphical encoding of its environment. The SGC loss is simple to implement and encourages representations that encode objects' semantics, relationships, and history. By using the SGC loss, we attain gains on three embodied tasks: Object Navigation, Multi-Object Navigation, and Arm Point Navigation. Finally, we present studies and analyses which demonstrate the ability of our trained representation to encode semantic cues about the environment.

1. Introduction.

Researchers have pursued designing embodied agents with general neural architectures and training them via end-to-end reinforcement learning (RL) to flexibly complete a range of complex tasks. In practice, however, training agents to perform long horizon tasks using only terminal rewards has been ineffective and inefficient [37], particularly in complex visual environments with high-dimensional sensor inputs and large action spaces. This has led to the use of several common "tricks" to improve training, *e.g.* manually engineered shaped rewards, use of off-the-shelf vision models to pre-process images, imitation learning with expert trajectories, and the use of special purpose mapping architectures [9, 10, 69, 37, 22, 56, 40].

Looking to reduce the need for such "tricks", one promising line of work has looked into training agents with RL and *auxiliary losses* to encourage the production of powerful and useful environment representations. These include self-supervised losses like forward prediction [31], contrastive predictive coding [32], and inverse dynamics [55], that are task and environment independent. The cost of this generality is that there are no guarantees that the resulting representations will encode task-relevant features like the semantic grounding of objects, which is often key to agent success. Moreover, in practice, these losses tend to work well in video game and gridworld environments but are not effective in more complex visual worlds. On the other hand, supervised auxiliary losses such as disturbance avoidance [53], depth generation [52] are frequently designed to help with specific tasks but are not generally useful for new tasks.

In this work, we propose the Scene Graph Contrastive (SGC) loss. SGC uses, as its supervisory signal, a non-parametric scene graph that develops and transforms iteratively as the agent interacts with its environment. The agent, objects, and rooms are represented as nodes, agent-object (*e.g. Sees* and *Touches*) and object-object (*e.g. Contains* and *Above*) relationships are edges and, category and spatial coordinates are represented as node attributes. SGC does not employ any graph decoders which tend to be complex, challenging, and expensive to train [20, 49]. Instead, it uses a contrastive learning approach in which the agent must "pick out" the graph corresponding to the observations it has seen – a much simpler learning mechanism which still encourages the agent to develop a graph-aware belief state, see Figure 1. Additionally, it prevents the need for a scene graph during evaluation. As we iteratively build the ground truth scene graph in an episode, we naturally generate hard negative samples as graphs from nearby spatio-temporal states will only be subtly different from one another.

The SGC loss has several desirable characteristics. Firstly, it encourages the belief representations to summarize object semantics, relationships, and history, information that can be intuitively useful for completing navigation-based embodied tasks. We present results that demonstrate the efficacy of these representations on multiple tasks. Second, it requires the scene graph *only at training time*, which allows us to use any available supervisory data useful for learning powerful representations for embodied tasks. Third, it is simple to implement. SGC does not require designing complex specialized decoders for predicting the scene graph and instead leverages well-studied, graph encoder networks and contrastive losses.

We evaluate the SGC loss by training agents on three complex navigation-based tasks. These tasks include Object Navigation (ObjectNav) [18] (evaluated across four benchmarks), Multi-Object Navigation (MultiON) [68], and Arm Point Navigation (ArmPointNav) [23]. Across each of these tasks, SGC provides very significant absolute improvements of **10%** in ObjectNav, **9%** in MultiON and **3.5%** in ArmPointNav over models trained with pure RL. We find that these agents learn to represent many semantic cues about their environment, which we show via two studies. First, we demonstrate that SGC-trained agents can be quickly fine-tuned to novel goal object categories that were observed previously in their environments but never used as target objects for the task. Second, we present linear probing experiments to study how the SGC loss impacts agents' understanding of free-space and object semantics. We find that the representations learned by the SGC-trained agent outperform those of an RL-only trained agent; suggesting that the SGC loss encourages agents to better represent these concepts.

In summary, our contributions include: (1) a proposal to use scene graph as a supervisory signal for training embodied agents, (2) the formulation of a Scene Graph Contrastive (SGC) loss which avoids the need to use complex graph decoders, and (3) a suite of experimental results which demonstrate that the SGC loss leads to significant performance and sample efficiency gains across multiple embodied tasks.

2. Related Works.

Embodied AI in practice. The Embodied AI community has been working on several embodied tasks such as navigation [7, 39, 28, 51], instruction following [62, 4, 42], manipulation [23, 74], embodied question answering [17, 30], and rearrangement [69, 6]. Open source simulators [41, 18, 64, 45] and benchmarks [62, 6, 69] have enabled tremendous progress on these tasks. Recently, large-scale training [19] and stronger visual backbones [40] have shown promising transfer across various environments. However, training performant agents often requires "tricks" like manually engineered shaped rewards, imitation learning with expert trajectories [37, 56], and use of special purpose mapping modules [9, 10], which tend to be task specific. In contrast, we propose a supervisory signal that encourages agents to learn better representations and show it to be effective across multiple navigation-based tasks.

Auxiliary Tasks in Reinforcement Learning. Auxiliary tasks in tandem with the RL task objective have shown promising results in improving sample efficiency and asymptotic task performance for visual reinforcement learning. These can be supervised tasks that provide external signals like depth maps [52, 29, 71], game internal states [43] and reward prediction [36]. Various unsupervised/self-supervised auxiliary objectives like auto-encoders [44, 34, 78], forward [31] and inverse dynamics [55], spatio-temporal mutual information maximization [2, 35], contrastive learning [32, 33, 80, 63], derive supervision from the agent's own experience. Recently, [80, 79] have shown that self-supervised auxiliary tasks can improve sample efficiency on embodied navigation tasks [3, 7].

Unfortunately, these approaches may not effectively encode task-relevant features, and therefore fail to provide improvements on complex tasks in photo-realistic environments [41, 59]. To alleviate this, we propose to use scene graphs as an auxiliary supervisory signal and show that it leads to more performant agents across different tasks.

Scene Graphs. Building rich scene representations has been an active area of research including approaches to build graphs from static images [16, 47, 50, 75, 76, 81] and ones that contain temporal information from videos [15, 38, 48, 54, 60, 65]. These methods capture 2D spatial relationships between objects. There has also been work that aims to encode 3D relationships [5, 14, 25, 67, 82].

Scene graphs have been used in embodied settings for

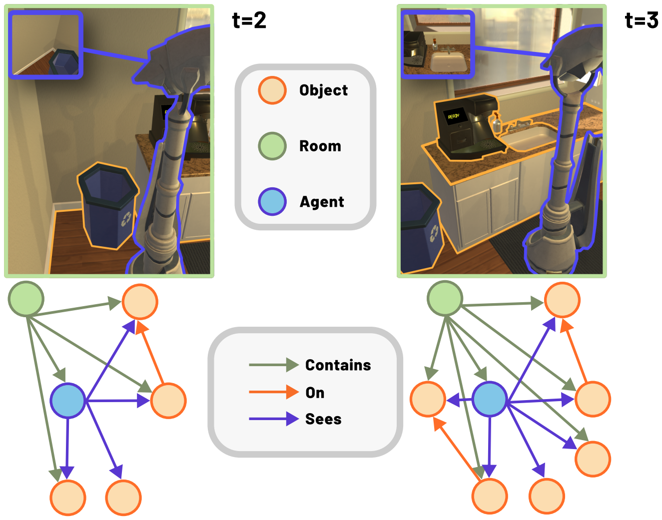


Figure 2: **Iterative Graph Building.** Illustrative example of how a scene graph is built. The agent, the room, and the objects are all added as nodes to the graph. As the agent moves in the environment, we add the objects that it sees, for instance at t=3, the coffee machine, sink, and countertop. We also add edges that signify various relationships like *Contains*, *On*, and *Sees*.

learning physics engines [8], visual navigation [21, 77, 11, 12, 58, 73], manipulation [83] and building actionable representations [46, 57]. There has also been work [26] to encode scene graph relationships for use in downstream tasks like object tracking and room rearrangement. However, contrary to this stream of work, we do not attempt to generate scene graphs or use it as an input to the agent. Instead, we use it as an auxiliary training signal in a contrastive learning setup. This avoids using complex graph decoding while enabling graph-aware belief representations.

3. Approach.

In this section, we present the Scene Graph Contrastive (SGC) loss for aiding embodied agent training. We begin by discussing the approach of iteratively building a scene graph from the agent's observations. Subsequently, we discuss how we use scene graphs as a training signal in our contrastive learning framework. Lastly, we describe our model architecture and how we train embodied agents for various tasks with our proposed loss.

3.1. Iterative Scene Graph.

Embodied AI simulators provide a rich trove of information like scene semantics, object positions, geometry, and spatial relationships. We attempt to distill this information by building a scene graph consistent with the agent's explored environment and use it to construct a supervisory signal. We define this scene graph as a non-parametric, object-centric, directed, graph representation. Figure 2 shows an example of how we iteratively build the scene based on the agent's exploration in the environment.

Node Features. We build a scene graph that iteratively updates based on the agent's path through the environment. We begin with the agent as the first node in the graph. Then, we add all the objects as nodes that are visible and within a threshold distance of the agent. Every instance of a particular object type is treated as a separate node. Additionally, if we're operating in house-sized environment, we add the room in which the agent is currently present as a node in the graph. Note that, once an object node is added in the graph, it continues to exist on the subsequent time-step graphs as well, even if the object goes out of view. This allows the scene graph to retain the history of an episode, which can be a useful attribute for long-horizon tasks.

Additionally, we also assign node-specific features to each node. These comprise of a concatenation of (1) an embedding of the object's type and, (2) the (x, y, z) 3-D coordinates of the object, *i.e.* its position. Each object position is defined relative to the agent, to encode spatial awareness about the environment with respect to the agent's current state. Note that these node features are updated after every agent step as they would otherwise quickly become invalid.

Edge Features and Relationships. We use the edges of the scene graphs to encode various relationships between the nodes. These relationships can be Agent-Object like *Sees*, *Holds* or *Touches*. Other agent-agnostic relationships include Object-Object relationships like *On* or *Near*. We also have Agent conditioned Object-Object relationships like *Right*, *Left*, *Above* which depend on the object positions relative to the agent perspective. We provide a list of all relationships and how they are estimated in the supplementary materials. We also have a relationship *Contains*, between the rooms of a house and other nodes, that determine whether the agent or a particular object is present in that room. At each time step, we compute these relationships between all the nodes based on their positions and geometry. The information needed to compute these relationships, i.e. object and agent poses, is readily available in most open source simulators [41, 18, 19, 64, 61, 74, 27], making it straightforward to construct this scene graph.

If a relationship is true, for instance, *Sees*(Agent, Apple), we add a directed edge between the agent node and apple node with the attribute *Sees* set as true. Relationships are updated at each time step. This means that the set of edges between nodes is not static and may change between agent steps, *e.g.* the agent might lift an apple off of a table resulting in the apple no longer being *On* the table.

3.2. Scene Graph Contrastive Learning.

The scene graph described in the Section 3.1 is a rich source of information about the environment and can allow

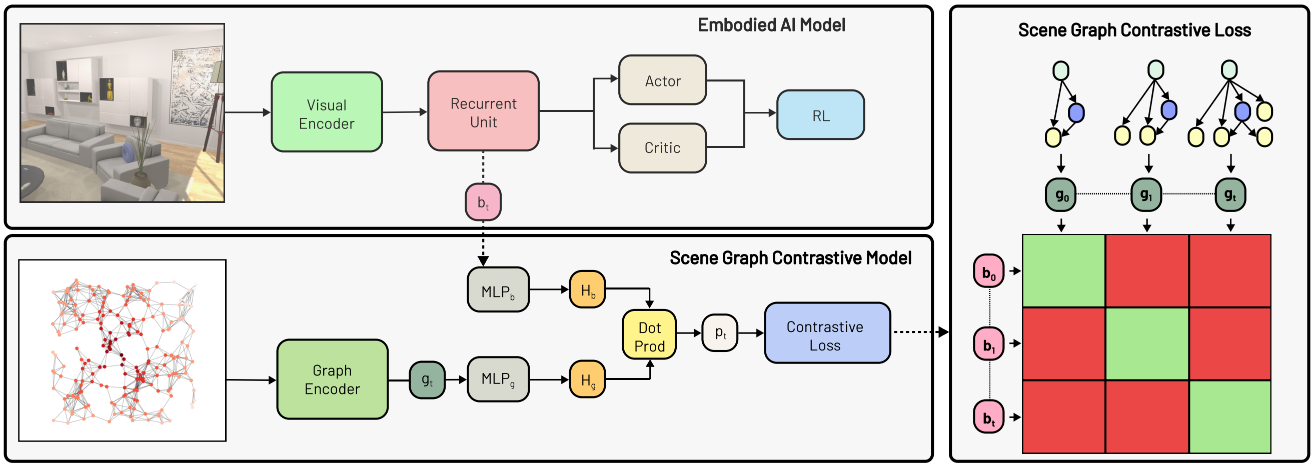


Figure 3: **Scene Graph Contrastive (SGC) framework.** We show a standard Embodied-AI model which takes, as input, an egocentric RGB observation and outputs a policy and scalar value for RL training. We supplement this with our Scene Graph Contrastive Model. It takes as input, the belief, b_t, and graph representation g_t, at each time step. The prediction of this model, p_t is passed into the Scene Graph Contrastive Loss. The SGC Loss optimizes the model to predict which graph embedding belongs to a particular time step, t.

the agent to perform a range of tasks with ease. However, the privileged metadata information required for the scene graph's construction is often unavailable during inference time when deploying to unseen environments or real-world settings. Therefore we propose to use it as an auxiliary *training-only* signal. This alleviates its need during inference, allowing us to deploy our trained policies even in the absence of privileged metadata information.

One common approach for leveraging supervisory information for representation learning is simply to build a decoder module that directly attempts to predict that supervisory information. Building such a decoder in our setting is computationally expensive and cumbersome: attempting to directly predict a scene graph with an unknown number of nodes would, similarly as for language prediction, require an iterative decoding mechanism which would, one by one, add nodes and edges to a graph until some stopping criterion was reached. While directly predicting such a graph would certainly have advantages, its computational cost makes it unappealing in a RL setting. We consider another approach.

Intuitively, we would like an agent's representation of their environment, commonly called the agent's belief, to be similar to a scene graph representation of the agent's environment. We achieve this by training a contrastive loss that effectively asks the agent to *pick out* the scene graph corresponding to its observations at particular time step from among other distractor graphs. Unlike decoding, encoding a scene graph is a significantly easier task, see Sec. 3.3.

We collect scene graphs from parallel agent rollouts, which implies that we have some scene graphs in the batch that are from the same episode, and some from entirely different episodes. As discussed in Section 3.1, we iteratively build the scene graphs as the agent observes new objects in an episode. This enables us to automatically generate hard negative samples as graphs from nearby time steps of the same episode differ very slightly from one another.

3.3. Training Embodied Agents with SGC.

We propose to train Embodied-AI models using scene graphs as auxiliary supervision as described in Section 3.2. As shown in Figure 3, we consider a typical Embodied-AI model. It consists of a visual encoder to process the observations from the environment. Following [40] we use a frozen CLIP-ResNet50 encoder for encoding our visual observations. We also have a recurrent unit, specifically a GRU for keeping a memory of these visual features, followed by a linear actor-critic layer for reinforcement learning. We refer to the GRU output as beliefs, denoted by b_t, where t is the time step.

To enable auxiliary learning we propose a **Scene Graph Contrastive Model** as shown in Figure 3. It consists of a graph encoder, comprised of three Graph Attention Layers [66], followed by a global max-pool across node features. The graph encoder produces a representation, g_t, for the current time step's scene graph. Following [13], we use two multi-layer perceptrons, denoted by MLP_b and MLP_g in Figure 3, to encode the beliefs, namely we let:

$$H_{b_t} = \text{MLP}_b(b_t) \in \mathbb{R}^D, \quad H_b = [H_{b_1} \ \dots \ H_{b_T}] \in \mathbb{R}^{D\times H} \ ,$$
$$H_{g_t} = \text{MLP}_g(g_t) \in \mathbb{R}^D, \quad H_g = [H_{g_1} \ \dots \ H_{g_H}] \in \mathbb{R}^{D\times H} \ .$$

Without loss of generality, above we consider the case where our loss is computed using H sequential agent steps; in practice, our loss will be computed on a batch of such trajectories. As shown in Figure 3, we take a dot product of embeddings H_b and H_g, to generate a prediction matrix P:

$$p_{t,s} = H_{b_t} \cdot H_{g_s}, \quad P = H_b^T H_g = [p_{t,s}] \in \mathbb{R}^{H \times H}.$$

We take a softmax across the columns of P, and pass each row to a cross entropy loss. The ground truth for this loss is a diagonal matrix as shown in the Scene Graph Contrastive (SGC) Loss panel in Figure 3, *i.e*. the entries of P where $t = s$. This classification objective attempts to predict which graph embedding belongs to a particular time step. Note that the SGC loss, is optimized as an auxiliary objective alongside usual reinforcement learning losses; in our experiments we use the DD-PPO RL loss [72].

It is worth noting that, constructing a scene graph at every step can be computationally expensive. To avoid this overhead, we randomly sample time steps at which we generate the graph, and only compute the loss at those steps. We show the effectiveness of SGC through various experiments and analysis in Section 4.

4. Results.

4.1. Experiment Setup.

Dataset. We use the ProcTHOR [19] framework to train agents for various embodied tasks. ProcTHOR provides 10,000 training houses, which we refer as *ProcTHOR-Train*. We train all our agents on these environments. ProcTHOR also provides *ProcTHOR-Val*, 1000 validation houses that the agent does not see during training. We use the AllenAct [70] framework to train our models.

Model variants. For each task, we train two agents:

• *RL* [19] This agent is trained with pure reinforcement learning (RL), specifically DD-PPO [72]. We use the model from [19] that achieved state-of-the-art results on various embodied navigation benchmarks.

• *RL+SGC*: To demonstrate the efficacy of our Scene Graph Contrastive (SGC) loss, we train this agent with SGC as an auxiliary objective to RL as described in Section 3.3.

Note that both agents are trained with the *same* hyperparameter setups. We provide the hyperparameter and training details in the appendix. We use a frozen CLIP-ResNet50 as our visual encoder, and a GRU as the recurrent unit and train these agents on three embodied tasks, namely Object Navigation, Multi-Object Navigation, and Arm Point Navigation for 350M, 180M, and 90M steps, respectively.

Note on ProcTHOR [19] results. We would like to remark that due to recent updates in AI2-THOR [41], the simulated LoCoBot agent is now allowed to look down by up to 60° (in previous versions this was 30°). The models presented in [19] were trained before this update, therefore we use the authors' code to retrain the agents and present updated numbers. We have confirmed with the authors of [19] about this change and validated our results. Additionally, we present training curves for Object Navigation training in the supplementary to justify the efficacy of the proposed SGC loss. Note that, for a given task, we use the same hyperparameters for all the models trained on it for a fair comparison.

Benchmark	Model	SR	SPL	EL
RoboTHOR	RL+SGC (ours)	**53.2**	**32.8**	245
	RL [19]	41.0	28.0	193
ArchitecTHOR	RL+SGC	**53.8**	**34.8**	204
	RL [19]	48.7	33.4	152
AI2-iTHOR	RL+SGC (ours)	**71.4**	**59.3**	124
	RL [19]	62.6	53.6	75
ProcTHOR-Val	RL+SGC (ours)	**70.8**	**48.6**	173
	RL [19]	62.4	45.5	80

Table 1: **Results on Object Navigation.** SR, SPL, and EL indicate the success rate, success weighted by path length and episode length.

4.2. Object Navigation.

Task. Object Navigation (ObjectNav) requires an agent to locate a specified object category. The agent begins the episode at a random location and is given a target object category, for instance, apple. All our ObjectNav agents are trained with a simulated LoCoBot (Low Cost Robot) [1], and use egocentric RGB images as input. We provide details about the action space in the supplementary.

Metrics. An episode is considered successful if the agent takes an End action and the target object category is visible and within 1m of the agent. We report the success rate (SR) and Success weighted by path length (SPL) [3] for this task. We also report the average episode lengths (EL) for the trajectories traversed by the agent.

Results. In Table 1, we evaluate our trained models across 4 ObjectNav datasets. To reiterate, we train our models on *ProcTHOR-Train*. First, to show in-domain generalization, we evaluate both agents on *ProcTHOR-Val* and achieve an improvement of **12%** in SR and **4.8%** in SPL.

Moreover, following [19], to investigate the cross-domain generalization of our approach, we perform *zero-shot* evaluations on the RoboTHOR, AI2-iTHOR, and ArchitecTHOR ObjectNav datasets. *Zero-shot* here implies, that neither of the models have been trained on scenes from these datasets. As shown in Table 1, we observe that using SGC provides a clear gain across all domains, thereby indicating its effectiveness in producing generally performant ObjectNav models. We see a substantial improvement of **12%** on RoboTHOR, **9%** on AI2-iTHOR, and **5%** on ArchitecTHOR in the SR metric.

Another interesting insight from our experiments is that ObjectNav models trained with SGC consistently traverse longer trajectories, as reflected by the EL metric. On further investigation, we find that this can be attributed to our

Benchmark	Model	SR	SPL	EL
MultiON-2	RL + SGC (ours)	**36.7**	**23.7**	255
	RL	34.4	18.2	298
MultiON-3	RL + SGC	**21.4**	**11.6**	354
	RL	13.3	7.62	338

Table 2: **Results on Multi-Object Navigation.** SR, SPL, and EL indicate the success rate, success weighted by path length and episode length.

RL+SGC agent producing fewer false positives by waiting for episodes to timeout instead of taking the END action when the target object is not visible. Our conjecture is that an agent trained with SGC keeps exploring its environment unless its very certain it has found the target or exhausts the maximum number of steps allowed. For instance, in RoboTHOR, we find that agents trained with just RL execute the END action incorrectly in **45%** episodes. On the other hand, RL+SGC does so in only **14.9%** episodes. Our conjecture is that the SGC loss enriches the agent's understanding of the environment and prevents it from misrecognizing objects and pre-maturely ending episodes. We observe this trend across all ObjectNav datasets.

4.3. Multi-Object Navigation.

Task. We implement the Multi-Object Navigation (MultiON) task originally proposed in [68] in ProcTHOR [19] environments. We create two variants, MultiON-2 and MultiON-3, which require the agent to navigate to 2, and 3, objects in an episode respectively. Similar to ObjectNav, we train our agents with a simulated LoCoBot [1] and use egocentric RGB images as input. We provide details about the action space in the supplementary. We provide the first goal object category to the agent at the beginning of the episode. Once an agent successfully finds the first target, by calling the FOUND action with the object visible and nearby, we provide the next target object.

Metrics. The target object must be visible and within 1m of the agent for the FOUND action to be successful. An episode is successful if the agent can finds all the target objects. It is considered a failure otherwise. Following [68], we report the Success Rate , SPL and Episode Length metrics.

Results. We collect a validation dataset in the *ProcTHOR-Val* environments. We use 200 houses that the agent *never* sees during training. As mentioned before, we present results on two MultiON variants, MultiON-2 and MultiON-3, where the agent needs to navigate to 2 and 3 target objects respectively. As shown in Table 2, for MultiON-2, we observe an improvement of **2.3%** in Success Rate, and **5.5%** in SPL. We replicate a similar setup for MultiON-3, and see a large **8%** improvement in Success Rate, and **4%** in SPL.

Benchmark	Model	SR	SRwD	EL
ArmPointNav	RL + SGC (ours)	**51.2**	**25**	150
	RL [19]	47.9	22.6	147

Table 3: **Results on Arm Point Navigation.** SR, SRwD and EL indicate the success rate, success rate without disturbance metrics and episode length.

4.4. Arm Point Navigation.

Task. To evaluate our approach on a mobile manipulation-based task, we train models to complete ArmPointNav, a visual mobile manipulation task proposed in [24]. This task requires an arm-equipped agent to move a target object from its starting location to a goal location. These locations are given to the agent in the agent's relative coordinate frame. Note that we do not use any other visual inputs besides egocentric RGB images. The action space consists of navigation actions and arm-based action for manipulating objects. We provide the complete action space in the supplementary.

Metrics. An episode is considered successful if the target object reaches the goal location. We report two metrics, Success Rate (SR) and Success Rate without Disturbance (SRwD). SRwD indicates how often the agent can complete the task without colliding with non-target objects.

Results. We evaluate on the AI2-iTHOR test tasks from [24], and report performance for both our models. We see a gain of **3.3%** on SR and **2.4%** on SRwD.

4.5. Ablation and Analysis.

4.5.1 SGC *v.s.* Other Auxiliary Objectives

As shown in Tables 1, 2 and 3, training with the SGC loss as an auxiliary objective with RL improves performance across various embodied tasks. However, to investigate how it compares to other auxiliary objectives, we present a comparison with two additional baselines:

- *RL+CPCA-16* [32]: A self-supervised objective that has shown sample efficiency improvements in PointNav [80].
- *RL+Visibility*: We implement an auxiliary loss in which the agent must predict whether a set of objects are visible or not at a every time step. This supervisory loss can be considered task-specific to ObjectNav as it directly informs the agent whether its seeing a certain object or not, intuitively a strong signal for this task.

We train these baselines with the same hyperparameter setup as the models presented in Table 1. Table 4 presents evaluation results on the *ProcTHOR-Val* ObjectNav benchmark. We find that RL+Visibility actually performs worse than RL, meaning that adding an auxiliary loss does not necessarily lead to performance benefits. We suspect that, as objects of most categories will not be visible to the agent at a given time step, the visibility loss is overwhelmed by negative examples and thus fails to provide a strong super-

Model	SR	SPL
RL + SGC (ours)	**70.8**	**48.6**
RL + CPCA-16	66.2	45.9
RL + Visibility	54.8	40.3
RL [19]	62.4	45.5

Table 4: **Comparing SGC with other auxiliary losses.** SR and SPL indicate the success rate and success weighted by path length on the *ProcTHOR-Val* ObjectNav benchmark.

visory signal. This emphasizes the challenge of designing good auxiliary losses: intuition often fails.

RL+CPCA-16 outperforms RL and RL+Visibility, but still lags RL+SGC by 4% in success rate. Our SGC loss is computed at only 20% of the timesteps for the purpose of computational efficiency. On the other hand, we computed the RL+CPCA-16 loss at every time step without subsampling; despite this, SGC outperforms.

4.5.2 Adapting to Novel Object categories.

Today's ObjectNav agents, including the models presented in this work, are trained to find a fixed set of object categories. However, in practice, we may wish to adapt our agents so as to enable them to navigate to novel object types outside their existing vocabulary. One, brute force, solution is to simply retrain from scratch every time we're presented with a new set of object categories. This would require an vast amount of compute and time, making it unfeasible.

We present an alternative, we take an ObjectNav model, trained on a set of object categories, and attempt to quickly finetune it on a set of new object categories. We achieve this by freezing the recurrent unit, usually a GRU, thereby preserving the belief representation. After freezing, the only parameters being optimized are the Actor-Critic head, and target object type encoder. This method builds upon the intuition that, after training the belief representations once, the GRU should learn to summarize information about the environment into actionable representations. Therefore, the success of this approach is dependent on, and hence indicative of, the quality of the belief representations.

To analyze the quality of the learned belief representations of our agents, we finetune them to navigate to novel object categories that were previously not used as targets. Note that these new categories were, however, present in the training environments. We follow the methodology described above and keep the belief representation frozen. We randomly sample 5 object types that were not in the initial training object categories, and then fine-tune both models, RL and RL + SGC, for 2 million steps in *ProcTHOR-train* environments with just reinforcement learning (DD-PPO [72]). To evaluate these fine-tuned models, we collect a validation dataset in the *ProcTHOR-Val* environments

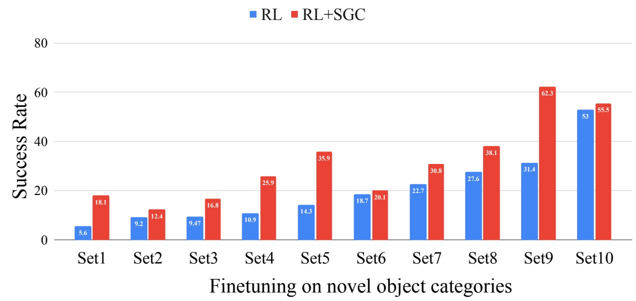


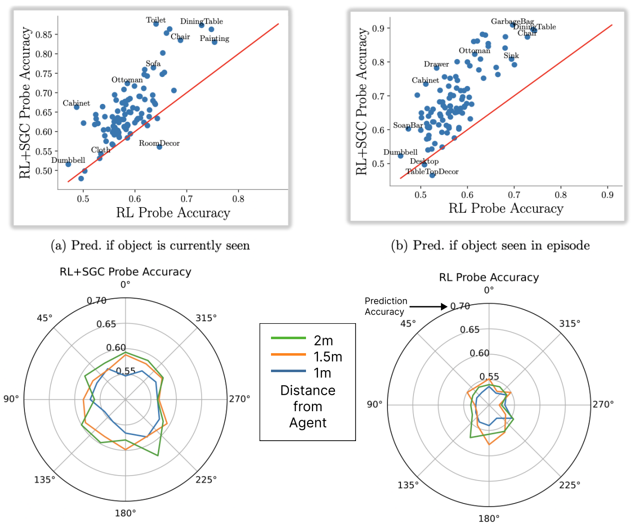
Figure 4: **Adapting to Novel Object types.** We sample 10 sets of 5 novel object types, and finetune RL + SGC and RL initialized models with DD-PPO [72] for 2 million steps. We observe consistent success rate improvements across all sets. For this visualization, we arrange the sets in the ascending order of success for the RL initialized model.

with the newly sampled target object categories.

Figure 4 displays the validation-set success rates of finetuned models corresponding to 10 randomly sampled sets of 5 object categories. We also conduct a paired t-test to ensure that the large observed difference between RL+SGC and RL is statistically significant. We indeed find that the gains in SR and SPL are significant at 0.01 and 0.05 levels, respectively. Note that RL+SGC initialized models generalize better to objects that are both easy (Set1), and hard (Set10), to navigate to. The results indicate that the beliefs trained with SGC are able to encode general semantic information about the environment, allowing the model to generalize to novel object categories much faster than the RL model. Details on the object pool and the 10 sets in Figure. 4 are provided in the supplementary materials.

4.5.3 Probing Learned Representations

To understand how SGC impacts the representations learned by our agent, we perform two linear probing experiments. Specifically, we evaluate two ObjectNav agents, our trained RL+SGC and RL agents, along fixed trajectories set in the ProcTHOR training scenes. At each step, we save both the agent's current belief states and additional metadata regarding what areas around the agent are free-space and what objects are visible to the agent. We then partition this data into training, validation, and testing splits, and train linear probes upon the frozen agent beliefs to predict the saved metadata. In particular, we fit binary logistic regression models to predict, for each object category: whether or not that object is currently visible and, to test agent memory, whether or not the agent has seen the object previously during the episode. We also fit such models to predict, at every step, whether or not various locations around the agent are "free-space" (*i.e.* can be occupied by the agent without collision). We summarize our test-set results in Figure. 5. We find that the RL+SGC agent is, al-



(a) Pred. if object is currently seen

(b) Pred. if object seen in episode

(c) RL+SGC Reachability

(d) RL Reachability

Figure 5: **Linear probing**. (a) Balanced accuracy of predicting if a given object is currently visible to the agent. (b) As for (a) but predicting if an object was ever seen till that point. (c) and (d) denote the accuracy for predicting if a location is reachable by the agent for RL+SGC and RL methods respectively. In {blue, orange, green}, we denote the accuracy along different directions for radii of {1m, 1.5m, 2m} around the agent oriented towards 0°.

most uniformly across object categories, better able to both predict which categories are visible and if they were previously seen. Similarly, the RL+SGC agent's beliefs are uniformly better at predicting free space about the agent, especially when predicting free-space *behind* the agent. Together this suggests that the SGC loss encourages developing both a semantic and geometric understanding of the environment. See supplementary materials for further details.

4.5.4 Importance of positional information

As discussed in Section 3.1, we encode the 3-D spatial positions of objects relative to the agent at each node in the graph. The knowledge of spatial positions can allow the agent to disambiguate between object instances of the same type, and enable the agent to spatially locate objects it had seen at previous time steps. To investigate the importance of this positional information, we remove it from our scene graph and train an ObjectNav model which we refer to as RL+SGC-*no position*. As shown in Table 5, we observe a 4% drop in performance on RoboTHOR ObjectNav. This suggests that encoding the positional information within the graph likely enables the agent to have better spatial awareness about its own state and the objects that it has seen, and thus leads to better performance at ObjectNav.

Model	SR	SPL
RL [19]	41	28
RL + SGC-*no hist.*	34.8	26.1
RL + SGC-*no position.*	48.1	28.4
RL + SGC (ours)	**53.2**	**32.8**

Table 5: **Variants of SGC.** SR and SPL metrics on RoboTHOR ObjectNav benchmark.

4.5.5 Importance of retaining history.

Section 3.1 discusses how we build an iterative scene graph from the agent's exploration of the environment. Once an object is added as a node to graph, it continues to exist in the graph, even if it goes out of view. We update the relationships and node position features between each pair of nodes at every step. One disadvantage of retaining the history of nodes is increasingly larger scene graphs as the episode progresses. This leads to some computational and memory overhead which lead us to investigate the importance of preserving the history in the scene graph. We train a model with a variant of the SGC loss that removes the history of nodes, thereby constructing a graph with only objects that are visible at the current time step. We refer to this model as RL + SGC-*no hist.* We summarize the results in Table 5. We find that when SGC is trained without retaining the history of nodes, it ends up performing worse than just RL. We believe that SGC-*no hist.* would encourage the agent's belief representation to only remember information about its observations at a given time step.

5. Conclusion.

We propose Scene Graph Contrastive (SGC) learning as a *general-purpose*, *supervisory* signal for training embodied agents. SGC employs non-parametric scene graphs as a *training-only* signal in a contrastive learning framework. It effectively asks the agent to pick-out the graph corresponding to its present and past observations, thereby encouraging the agent to develop a graph-aware belief state. We evaluate SGC, by training agents on three embodied tasks, Object Navigation, Multi-Object Navigation and Arm Point Navigation and show performance improvements across all of them. Additionally, we evaluate the quality of our belief representations by showing adaptation to novel object categories and via a linear probing analysis.

Limitations and future work. Building an iterative scene graph adds a computational overhead during training. Engineering solutions to speed these up can allow denser sampling of graphs and lead to a potentially stronger training signal. We use vanilla Graph Attention Networks [66] to encode scene graphs. Stronger graph encoders models have been proposed and may provide an even richer graph representation and lead to better embodied agents.

References

[1] Carnegie mellon university. locobot: an open source low cost robot. *http://www.locobot.org/*. 5, 6

[2] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. *NeurIPS*, 2019. 2

[3] 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*, 2018. 2, 5

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CVPR*, 2018. 2

[5] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir Roshan Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. *ICCV*, 2019. 2

[6] Dhruv Batra, Angel X. Chang, S. Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied ai. *ArXiv*, 2020. 2

[7] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *ArXiv*, 2020. 2

[8] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016. 3

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020. 1, 2

[10] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 1, 2

[11] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Kumar Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. *CVPR*, 2020. 3

[12] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, F. Xia, Álvaro Soto, Marynel Vázquez, and Silvio Savarese. A behavioral approach to visual navigation with graph localization networks. *ArXiv*, 2019. 3

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020. 4

[14] Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. *ICCV*, 2019. 2

[15] Yuren Cong, Wentong Liao, Hanno Ackermann, Bodo Rosenhahn, and Michael Ying Yang. Spatial-temporal transformer for dynamic scene graph generation. In *ICCV*, 2021. 2

[16] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. 2017. 2

[17] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. *CVPR*, 2018. 2

[18] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020. 2, 3

[19] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. 2022. 2, 3, 5, 6, 7, 8

[20] Apoorva Dornadula, Austin Narcomey, Ranjay Krishna, Michael Bernstein, and Fei-Fei Li. Visual relationships as functions: Enabling few-shot scene graph prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[21] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation. *ArXiv*, 2020. 3

[22] Kiana Ehsani, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Object manipulation via visual target localization. In *ECCV*, 2022. 1

[23] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. *CVPR*, 2021. 2

[24] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A Framework for Visual Object Manipulation. In *CVPR*, 2021. 6

[25] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM SIGGRAPH*, 2011. 2

[26] Samir Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. *CVPR*, 2022. 3

[27] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin T. Feigelis, Daniel Bear, Dan Gutfreund, David D. Cox, Antonio Torralba, James J. DiCarlo, Josh Tenenbaum, Josh H. McDermott, and Dan Yamins. Threedworld: A platform for interactive multi-modal physical simulation. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. 3

[28] Chuang Gan, Yiwei Zhang, Jiajun Wu, Boqing Gong, and Joshua B. Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. *ICRA*, 2020. 2

[29] Daniel Gordon, Abhishek Kadian, Devi Parikh, Judy Hoffman, and Dhruv Batra. Splitnet: Sim2sim and task2task transfer for embodied visual navigation. *ICCV*, 2019. 2

[30] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. *CVPR*, 2018. 2

[31] Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aäron van den Oord. Shaping belief states with generative environment models for rl. In *NeurIPS*, 2019. 1, 2

[32] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo Ávila Pires, Tobias Pohlen, and Rémi Munos. Neural predictive belief representations. *ArXiv*, 2018. 1, 2, 6

[33] Zhaohan Daniel Guo, Bernardo Ávila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. *ICML*, 2020. 2

[34] David R Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *NeurIPS*, 2018. 2

[35] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019. 2

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

[37] Unnat Jain, Iou-Jen Liu, Svetlana Lazebnik, Aniruddha Kembhavi, Luca Weihs, and Alexander G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 15121–15131. IEEE, 2021. 1, 2

[38] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *CVPR*, 2020. 2

[39] Peter Karkus, Shaojun Cai, and David Hsu. Differentiable slam-net: Learning particle slam for visual navigation. *CVPR*, 2021. 2

[40] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *CVPR*, 2022. 1, 2, 4

[41] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 2, 3, 5

[42] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. *ECCV*, 2020. 2

[43] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, 2017. 2

[44] Sascha Lange and Martin A. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. *IJCNN*, 2010. 2

[45] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR, 2021. 2

[46] Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. Embodied semantic scene graph generation. In *CoRL*, 2021. 3

[47] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. *ICCV*, 2017. 2

[48] Chenchen Liu, Yang Jin, Kehan Xu, Guoqiang Gong, and Yadong Mu. Beyond short-term snippet: Video relation detection with spatio-temporal global context. *CVPR*, 2020. 2

[49] Hengyue Liu, Ning Yan, Masood Mortazavi, and Bir Bhanu. Fully convolutional scene graph generation. In *CVPR*, 2021. 2

[50] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016. 2

[51] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. *ICCV*, 2021. 2

[52] Piotr Wojciech Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, L. Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *ICLR*, 2017. 1, 2

[53] Tianwei Ni, Kiana Ehsani, Luca Weihs, and Jordi Salvador. Towards disturbance-free visual mobile manipulation. *CoRR*, abs/2112.12612, 2021. 1

[54] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. *CVPR*, 2021. 2

[55] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *ICML*, 2017. 1, 2

[56] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. *CVPR*, 2022. 1, 2

[57] Antoni Rosinol, Arjun Gupta, Marcus Abate, J. Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *ArXiv*, 2020. 3

[58] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *ICLR*, 2018. 3

[59] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *ICCV*, 2019. 2

[60] Xindi Shang, Tongwei Ren, Jingfan Guo, Hanwang Zhang, and Tat-Seng Chua. Video visual relation detection. *ACM international conference on Multimedia*, 2017. 2

[61] Bokui Shen, Fei Xia, Chengshu Li, Roberto Mart'in-Mart'in, Linxi (Jim) Fan, Guanzhi Wang, S. Buch, Claudia. Pérez D'Arpino, Sanjana Srivastava, Lyne P. Tchapmi, Micael Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. *IROS*, 2021. 3

[62] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *CVPR*, 2020. 2

[63] A. Srinivas, Michael Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020. 2

[64] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *NeurIPS*, 2021. 2, 3

[65] Yao-Hung Hubert Tsai, Santosh Kumar Divvala, Louis-Philippe Morency, Ruslan Salakhutdinov, and Ali Farhadi. Video relationship reasoning using gated spatio-temporal energy graph. *CVPR*, 2019. 2

[66] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph attention networks. *ICLR*, 2017. 4, 8

[67] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. *CVPR*, 2020. 2

[68] 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, 6

[69] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. *CVPR*, 2021. 1, 2

[70] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020. 5

[71] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. *CVPR*, 2019. 2

[72] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. ICLR, 2020. 5, 7

[73] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart J. Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. *ICCV*, 2019. 3

[74] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. *CVPR*, 2020. 2, 3

[75] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *CVPR*, 2017. 2

[76] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *ECCV*, 2018. 2

[77] Wei Yang, X. Wang, Ali Farhadi, Abhinav Kumar Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *ICLR*, 2019. 3

[78] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *AAAI*, 2021. 2

[79] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectnav. In *ICCV*, 2021. 2

[80] Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation. In *CoRL*, 2020. 2, 6

[81] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, 2018. 2

[82] Yang Zhou, Zachary While, and Evangelos Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. *ICCV*, 2019. 2

[83] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *ICRA*, 2021. 3