Improving Target-driven Visual Navigation with Attention on 3D Spatial Relationships

Yunlian Lyu · Yimin Shi · Xianggang Zhang

Abstract
Embodied Artificial Intelligence has become popular in recent years. Its task shifts from focusing on internet images to active settings, involving an embodied agent to perceive and act within 3D environments. In this paper, we study the Target-driven Visual Navigation (TDVN) in 3D indoor scenes using deep reinforcement learning techniques. The generalization of TDVN is a long-standing ill-posed issue, where the agent is expected to transfer intelligent knowledge from training domains to unseen domains. To address this issue, we propose a model that combines visual and relational graph features to learn the navigation policy. Graph convolutional networks are used to obtain graph features, which encodes spatial relations between objects. We also adopt a Target Skill Extension module to generate sub-targets, in order to allow the agent to learn from its failures. For evaluation, we perform experiments in the AI2-THOR. Experimental results show that our proposed model outperforms baselines under various metrics.

Keywords Deep reinforcement learning · Graph convolutional networks · Visual navigation · 3D scenes

1 Introduction
Artificial intelligence (AI) has brought great convenience to our lives. From a home service robot asked to “open the cabinet under the coffee machine and give me a cup inside”, to a device that helps its visually impaired wearer navigate an unfamiliar subway, a wide range
of abilities need to be demonstrated for the next generation of AI-powered assistants. To develop these skills, many researchers believe that the most effective way is to focus on embodied AI tasks in 3D environments [1], such as target-driven visual navigation (TDVN) [2], instruction following [3], and embodied question answering [4]. These tasks ground the system’s training using interactive environments instead of relying on static datasets (e.g. ImageNet [5], COCO [6], VQA [7]). Compared with internet image-based tasks, embodied AI tasks require special skills of active perception, long-term planning, and learning from interactions. Since Deep Reinforcement Learning (DRL) can naturally manage vision and motion, approaches based on DRL have attracted several researchers in these fields.

In this paper, we study target-driven visual navigation in 3D indoor scenes. The agent perceives its environment through egocentric views and can perform a series of atomic actions, such as moving ahead or turning right. A natural way to instruct a robot is to ask it to move near a particular location, such as a place (e.g. “go to sofa”) [8] or someone’s room (e.g. “go to kitchen”) [9]. Similar to [2], we communicate with the agent by showing it a single image of a distant target (see Fig. 1). We consider two types of targets: static targets and actionable targets (please refer to Sect. 4.1 for a detailed description). The agent is required to intelligently navigate to a destination from any starting positions according to the assigned target image. The agent observes its environment at each step, matches it with the given target image, and then determines its following action.

Training an intelligent agent like human beings faces lots of difficulties, such as few-shot learning, avoiding collisions, target-induced exploration. In TDVN, the challenge of generalization is a long-standing ill-posed issue, where the agent is unable to transfer intelligent knowledge from training domains to unseen domains [10]. Previous approaches usually use pixel-level visual features to learn optimal policy, which fails to perform well across unseen targets and scenes. To solve this problem, we consider the question: How do we humans navigate? Our visual system can perceive and process various signals from observations to navigate freely in diverse environments. It is able to transform 2D view inputs into powerful 3D knowledge to perceive surrounding 3D environments (see Fig. 2). The 3D knowledge from our visual system provides us multitudinous information:

- Object recognition, which detects and classifies objects in observations;

Fig. 1 Examples of target-driven visual navigation in 3D indoor scenes. Our targets consist of static targets (a) and actionable targets (b). We also show the egocentric view of the agent and its corresponding actions at some time steps. Red actions represent the final two actions to finish the given task.
Improving TDVN with Attention on 3D Spatial Relationships

Fig. 2 Illustration of powerful 3D knowledge in our visual system. It has at least four indispensable perception capabilities that enable navigation to be completed: (1) object recognition; (2) object attributes; (3) obstacles awareness; (4) relational graph.

- Object attributes (e.g., color, shape, size, distance, category, openable), which are essential for understanding objects;
- Obstacle awareness (e.g., a table is on the left), which warns illegal actions and keep safe exploration;
- Relational graph, which indicates object positions and spatial relationships (e.g., left, front, under, up, on) between objects.

Those knowledge allows us to achieve intelligent behaviors towards various targets in all kinds of environments. For instance, we can navigate without collisions, find possible positions and adapt to various variations. A relational graph has abilities to indicate where objects are usually located, spatial positions, and adjacent relationships of different objects. For example, Mug is often placed next to Coffee Machine. With this type of graph knowledge, we can achieve better transfer learning. Since observations can be converted into graphs, and our visual system also models what we perceive in the forms of graphs, this work studies relational graphs in TDVN to obtain better performance. We intend to incorporate 3D spatial relations into classical DRL frameworks to help agents establish a knowledge system for navigation. The 3D spatial relationships are encoded by Graph Convolutional Networks (GCN). Finally, our navigation policy depends on visual features and attention mechanism on relational graphs to make reasonable decisions.

Sparse reward is a well-known challenge in DRL and one of the main reasons that make it difficult for agents to learn faster. To solve the parse reward problem, we propose a Target Skill Extension (TSE) module. Our basic idea is to leverage failed explorations of the agent to boost the training process (see Sect. 3.6). We evaluate our methods in the AI2-THOR [11]. It is an interactable framework for embodied AI agents, and provides near photo-realistic 3D indoor environments. Our experimental results in AI2-THOR show that our method achieves significant improvements compared to several baselines in both SR and SPL metrics. More details about evaluation metrics in TDVN are described in Sect. 4.3.

Our contributions are summarized as follows: (1) We propose a model that combines visual and relational graph features to learn the navigation policy. (2) We present an end-to-end reinforcement learning framework realizing the attention mechanism. (3) We adopt a
Target Skill Extension module to allow the agent to learn from its failures. (4) We show that our proposed model outperforms baselines under various metrics via numerous experiments.

The rest of this paper is organized as follows. Related work is presented in Sect. 2. Our proposed methods for TDVN are illustrated in Sect. 3. Experimental details and results are discussed in Sect. 4. Finally, we conclude this paper and discuss some future works in Sect. 5.

2 Related Work

Traditional approaches decompose a navigation task into two sub-tasks by building a 3D map of the scene and then planning in this constructed map [12, 13]. In these works, the navigation problem is treated as a purely geometric problem. However, a map of an environment is challenging to build, and these methods are not suitable for unseen environments. Recent success of deep learning [14] and reinforcement learning [15] has made learning-based approaches more popular. Deep reinforcement learning [16] combines the perception ability of deep learning with the decision-making ability of reinforcement learning, which can build autonomous systems with a higher-level understanding of the visual world. Learning-based navigation can be distinguished along many dimensions, such as visual semantic navigation [17, 18], instruction following [19, 20], and embodied question answering [4, 21]. In our case, the target is given to the agent in terms of image [2, 22, 23]. We focus on the task that evaluation metrics only consider navigation components compared with vision and language navigation.

Early learning-based researches aim at navigating in synthetic game-like environments [24–27], such as ViZDoom [28], Minecraft [29] and DeepMind Lab [30]. Learning-based methods solve the navigation problem in an end-to-end manner and do not require a map of the environment. In these works, pixel-level inputs are directly fed into DRL algorithms for visual navigation. The artificial agent moves in an environment based only on observations to find its target for rewards. However, their navigation performance is usually evaluated in the same training scenes or same finding targets. After that, recent works begin to evaluate the navigation performance in different scenes. These scenes are not used for training [8, 31–33]. [31] uses Deep Q-Networks for transfer learning study. [32] introduces a spatially structured 2D memory image to store information about the environment. [33] presents a hierarchical architecture where a meta controller learns to use the acquired skills. [8] propose a Gated-Attention mechanism to combine the image and text representations. These methods conduct several empirical studies of various DRL algorithms, while their methods are evaluated on slightly changing environments. From their work, they show that DRL agents could perform well when tested on over-simplified tasks, or environments that are only slightly varied from visual variations.

Nowadays, some studies focus on multi-target tasks and generalization ability after more realistic 3D simulated environments are constructed [2, 17, 18, 22, 23, 34–36], such as AI2-THOR [11], House 3D [37], Matterport 3D [38]. These environments motivate generalization research in more complicated situations. Typically, [2] proposes an actor-critic model whose policy is a function of the target as well as the current state. [22] introduces a model with the object localization network and the navigation network. [23] presents inverse dynamics model and multigoal colearning to adapt to different goals. Similar to [2, 22, 23], we represent our targets as RGB images. Besides, scene priors [18] and our work both consider graph models in DRL frameworks. However, [18] constructs a knowledge graph from visual genome [7] as prior knowledge for visual semantic navigation. Their targets are represented as words, and
Fig. 3 Overview of our proposed navigation method. It incorporates 3D relations into a deep reinforcement learning framework for target-driven visual navigation. The inputs of our model are egocentric observation and target image. We learn a policy \( \pi(\mathbf{a}_t | \mathbf{s}_t, \mathbf{s}_g) \) that decides an action based on the visual and graph features.

Their graph is readily available and fixed. In our work, the graph is represented in the form of 3D spatial knowledge. Instead of using graph convolutional networks [39] to extract graph features, we further use the attention mechanism to self-infer essential navigation rules, such as the most relevant references to attend on for reasoning or avoiding obstacles.

3 Proposed Method

In this section, we present an in-depth description of our proposed model (see Fig. 3), an architecture that extends standard DRL frameworks with the representations of 3D relations and our TSE module. We firstly extract features for observations and spatial graphs. Then the two types of features are combined and passed into the DRL model to produce an action. Each module is described in the following subsections.

3.1 Task Definition

Target-driven visual navigation is to design an intelligent agent that can find the minimum length sequence of actions to reach a pre-specified target. The agent is expected to reach a target position from any possible starting positions within a scene using only visual inputs. We formulate our navigation task as a Partially Observable Markov Decision Process (POMDP) problem considering the sequential-decision-making nature of visual navigation. The agent perceives its target and surrounding environments through RGB images. At each time \( t \), the agent receives an observation \( o_t \) and a target image \( o_g \), then produces an action \( \mathbf{a}_t \). Once the action is taken, the agent receives a scalar reward \( r_t \) and a new observation \( o_{t+1} \) from the environment. The agent’s state \( s_t \) is a function of its observation at time \( t \), \( s_t = f(o_t) \).

The main task of the agent is to find a policy \( \pi \) that maximizes the expected sum of future discounted rewards:

\[
V_\pi(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]
\]  (1)
where $\gamma \in [0, 1)$ is the discount factor, $r_t = r(s_t, a_t)$ is the received reward at time $t$, $s$ is the initial state, $s_t$ is the state at time $t$, and $a_t \sim \pi(\cdot|s_t)$ is the taken action generated by following the policy $\pi$ in state $s_t$.

### 3.2 Policy Learning

For robots that perform sequences of actions, a suitable framework of this learning problem is reinforcement learning [15], in which an agent learns the optimal action policy $\pi$ through trial and error interactions with its environments. In our work, the policy $\pi$ takes the current observation $o_t$ and the pre-specified target image $o_g$ as inputs as shown in Fig. 3. The agent’s action $a_t$ at each time step $t$ is determined by a parametrized policy function $\pi(a_t|s_t, s_g; \theta)$. Introducing targets into the policy $\pi$ is quite important, it allows an agent to generalize across multiple targets. We adopt popular asynchronous advantage actor-critic (A3C) [40] algorithm to train agents, which relies on learning both a policy $\pi(a_t|s_t, s_g; \theta)$ and value function $V(s_t, s_g; \theta_v)$ given the current state $s_t$ and target state $s_g$. A3C optimizes the policy by minimizing the loss function

$$L_{\pi}(\theta) = -\mathbb{E}\left[\sum_{t=1}^{T} (R_t - V(s_t, s_g; \theta_v)) \log \pi(a_t|s_t, s_g; \theta)\right],$$

where $\theta$ and $\theta_v$ are the parameters of the actor network and critic network, respectively. $T$ is the length of the explored trajectory. $R_t$ represents $k$-step return and is the discounted accumulative reward defined by

$$R_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}, s_g; \theta_v),$$

where $\gamma \in (0, 1)$ is the discounted factor that reflects the significance of future rewards. The value function is updated by minimizing the loss as follows:

$$L_v(\theta_v) = \mathbb{E}[(R_t - V(s_t, s_g; \theta_v))^2].$$

Finally, the overall loss function for A3C is $L_{A3C}(\theta, \theta_v) = L_{\pi}(\theta) + \alpha L_v(\theta_v)$, where $\alpha$ is a constant coefficient. In A3C, many instances of the agent interact in parallel with various environments, which both accelerates and stabilizes learning. As shown in Fig. 3, the A3C architecture we build on uses an LSTM [41] and two Multi-Layer Perceptrons (MLPs) to jointly approximate both policy $\pi(a_t|s_t, s_g; \theta)$ and value function $V(s_t, s_g; \theta_v)$. In our settings, the LSTM outputs a 256-d vector, and the hidden units of MLPs are 10 and 1 respectively.

The final gradient update rules of the actor network and critic network are shown as follows:

$$\theta \leftarrow \theta + \eta \nabla_{\theta} L_{\pi}(\theta) + \beta \nabla_{\theta} \mathcal{H}(\pi(a_t|s_t, g_t; \theta))$$

$$\theta_v \leftarrow \theta_v + \eta \nabla_{\theta_v} L_v(\theta_v)$$

where $\eta$ denotes the learning rate, $\mathcal{H}$ is the entropy of policy $\pi$. The hyperparameter $\beta$ controls the strength of the entropy regularization term [40].
3.3 Visual Features

Deep learning utilizing neural networks has achieved outstanding performance in various application domains, such as computer vision, speech recognition, and natural language processing [14]. The advancements in deep learning have been primarily perfected with convolutional neural networks (CNN). It is capable of learning representations of data with multiple levels of abstraction through numerous processing layers. At present, deep learning has facilitated reinforcement learning to build autonomous systems with a higher-level understanding of the visual world.

Our work is consistent with [2] to extract robust visual features. As shown in Fig. 3, we firstly use ResNet50 [42], which is trained with ImageNet [5] database, to extract visual features \( X_t \) for each input observation \( o_t \). Besides, we apply the same module to derive features for the target image and obtain \( X_g \). In detail, the ResNet50 network takes one \( 224 \times 224 \) RGB image as input, and outputs one 2048-d feature vector. After that, we concatenate the visual features from the current observation and the target image. The way we use is a deep siamese network [43] with 512 hidden units and ReLU activation. Then the concatenated information is fused through one fully connected (FC) neural network to form a joint representation \( I_t \). The FC network is with 256 hidden units and ReLU activation. Our final visual features are obtained by

\[
X_t = f(o_t; \theta_{\text{ResNet}}) \quad (7) \\
X_g = f(o_g; \theta_{\text{ResNet}}) \quad (8) \\
I_t = \sigma(W_f[\sigma(W_s X_t), \sigma(W_s X_g)]) \quad (9)
\]

where \( \theta_{\text{ResNet}} \) is the parameters of the ResNet50 network, \( W_s \) is the parameters of the siamese network, and \( W_f \) is the parameters of the fusion network. \( \sigma \) denotes the ReLU activation function and \([\cdot, \cdot]\) represents concatenation.

3.4 3D Relations

Except for visual features, we incorporate the representations of 3D spatial relationships into the navigation framework. The whole process is shown in the bottom part of Fig. 3, more details are shown in Fig. 4.

We denote the 3D relations by a graph \( G = (V, E) \). Each node \( v \in V \) represents an object. Each edge \( e \in E \) denotes a type of relationship between a pair of objects, such as next to, in, and below. To construct the 3D graph, the agent has to equip several powerful abilities. The first one is object recognition. The agent needs to know what objects are within its observations. In addition, the agent requires to understand the spatial relationships with multiple objects it views. Significantly, the work [21] has demonstrated that fine-tuned YOLOv3 [44] could be a good object detector. Since object detection is not our main task, we directly obtain object information from AI2-THOR to solve the first issue. The information we obtain from AI2-THOR is the same as object detection. For solving the second problem, as shown in Fig. 4, our Relation Extraction Module (REM) represents relations in the form of vectors, each of which has \( |V| \) elements. Each vector denotes one particular kind of spatial relationship. Each position of the vector becomes value one (yellow color) if the observed objects have this type of relation. In this work, our REM currently only considers the “next to” relationship between objects. We leave distinctive relations for future study, such as on, in, and below. Our agent is expected to reason about intelligent behaviors from the constructed graph.
Fig. 4 Illustration of our 3D graph framework. We build the 3D graph from REM while the agent explores its environments. Then the recently proposed GCN is used to encode the relationships between objects. Finally, we use an attention mechanism to self-infer the most relative references for guiding policy search.

Our 3D graph includes all objects that appear in AI2-THOR. Two nodes are connected with an edge if they are both visible. An object is considered visible if its distance is within 1.5 m from the agent’s camera. As illustrated in Fig. 4, we denote visible objects by a binary vector $R$ from REM. The binary value in each position of $R$ indicates whether the corresponding object is visible. We denote the total visible objects by $R_z$. It consists of visible objects $R_t$ from the observation and visible objects $R_g$ from the target image. Additionally, we build an adjacency matrix $A$ based on visible information $R$. In graph theory, an adjacency matrix is often used to represent a graph. The elements of the adjacency matrix $A$ become one when the pairs of nodes are both visible in $R$. We initialize each node feature $H_0$ as a one-hot vector. $\hat{A}$ is obtained by performing normalization on $A$ to make each node contain its own node features. In order to learn a representation of graphs in a low-dimensional latent space, we apply graph convolutional networks (GCN) [39]. GCN has achieved a superior performance in a wide range of tasks and applications [18]. The core idea behind GCN is to use edge information to aggregate node information to generate new node features (see top of Fig. 4). We further perform spatial information propagation to compute the node feature vector $H$ using GCN:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

(10)

where $\hat{A}$ is the normalized adjacency matrix, $H^{(0)}$ is the initial feature matrix, and $W^{(l)}$ is the parameter for the $l$th GCN layer. $\sigma$ denotes the ReLU activation function. In our settings, $H^{(0)} = I$, $I$ is corresponding to an identity matrix. We apply four GCN layers to extract the spatial knowledge, and the last layer of GCN generates a single value for each node. After several GCN layers, the final feature vector $H$ encodes the high-level knowledge about spatial relationships within various environments.
3.5 Attention Mechanism

For humans, our visual system usually pays more attention to the information that assists judgment, and ignores irrelevant information. Therefore, we further adopt the attention mechanism popular in recent years to the relation module. Attention mechanism in deep learning has achieved many state-of-the-art results in various applications [45–47].

As shown in Fig. 4, \( H_t \) and \( H_g \) indicates the spatial features from GCN for the current observation and the target image, respectively. Let the final attention vector \( H_a \) be the dynamic probability representation of the most relevant objects of the image at the time step \( t \) to find the current target. For each object \( i \), the attention mechanism module generates a positive weight \( \alpha_i \), which represents the probability that the object \( i \) is the right object to focus on for producing the next action. Since the next action of the agent is jointly determined by the target and the current observation, the weight \( \alpha_i \) of each object naturally depends on spatial features \( H_t \) and \( H_g \). Similar to visual features, we use the same fusion ways to get the fused spatial features \( Q \), which encodes the final spatial knowledge. Furthermore, we constrain the choices of the attention module to \( R_z \). Since there are multiple objects in \( o_g \), our agent also needs to learn what are the most appropriate references in the target image. Finally, we use two FC layers to compute the spatial representations \( H_a(i) \), \( i = 1, \ldots, |V| \), which corresponds to the probability extracted for different objects. The final attention vector \( H_a \) is described as follows:

\[
Q = W_f' [\sigma(W_s'H_t), \sigma(W_s'H_g)]
\]

\[
Q' = (QW_{fc1}W_{fc2}R_z)Q
\]

\[
H_a(i) = \frac{\exp(q'_i)}{\sum_{i=1}^{\mid V \mid} \exp(q'_i)}
\]

where \( W_s' \) and \( W_f' \) are the parameters of siamese and fusion networks, respectively. \( W_{fc1} \) and \( W_{fc2} \) are the parameters of two FC layers. \( \sigma \) denotes the ReLU activation function. \( Q = [q_1, \ldots, q_{|V|}] \) is the fusion spatial vector. The final output \( H_a \) is a probability vector of \( |V| \) elements as one input to the navigation network.

Finally, the joint representation of visual features and relational graph features, \([I_t, H_a]\), is fed into the DRL module to produce an action output \( a_t \).

3.6 Sub-targets Extraction

Sparse reward is a classic problem in deep reinforcement learning. An agent can not be rewarded frequently due to the huge searching space. HER [48] allows sample-efficient learning from sparse rewards. Consistent with the principle of HER, we propose a Target Skill Extension (TSE) module to solve this problem. Since one trajectory explored by the agent contains not only information from the starting point to the destination, but also information on how to reach the intermediate points of its trip. So if switching targets, then a failed experience can become a successful experience to reach other targets. Via this way, our agent can still learn TDVN skills from its failed tasks.

Suppose \(<s_1, a_1, r_1, \ldots, s_T, a_T, r_T>\>\) is a trajectory obtained by the agent after it explores the environment, where \( s_1 \) is a random start state and \( s_T \) is a terminal state. For each thread, in addition to training \( \pi(a_t|s_t, s_T; \theta) \), our TSE module also trains the policy \( \pi(a_t|s_t, s_j; \theta) \) (\( i \leq j \)) if \( s_j \) is a reasonable sub-target. For each sampled trajectory, as illustrated in Fig. 5, we randomly select some objects in observations as sub-targets. Such as novel
Fig. 5 Our TSE module decomposes an explored trajectory into some reasonable sub-trajectories by sub-targets. Sub-targets can be training targets (e.g., \( s_3 \)) trained by other threads or novel targets (e.g., \( s_5 \)) that are not used during training. Novel objects are the objects that can be picked up and not used for training. In this way, we divide an explored trajectory into multiple sub-trajectories for exploring different targets. Finally, our A3C model trains all these divided trajectories. Every time after training a sub-trajectory, we always assign the global network parameters to local networks. Our TSE module is demonstrated to be a superior sample efficient way for the agent to learn faster.

4 Experiment

This section presents the navigation results compared to several baselines without 3D relations and the TSE module. We evaluate the methods by testing their generalization performance against unseen targets and unseen scenes.

4.1 3D Environment

We conduct our experiments in the AI2-THOR [11] to evaluate the methods. It provides 3D virtual environments similar to real-world scenes, and includes four room categories: kitchens, living rooms, bedrooms, and bathrooms. There are 30 rooms in each room category. Each room has a set of objects and rich styles. Compared with other 3D environments, AI2-THOR allows the agent to perform several actions to interact with its scenes, such as open, pick up, and push. Some objects in the AI2-THOR are not directly visible if there are no interactions. For instance, cups are not visible since they are always hidden in closed cabinets at the beginning of random scene initialization. Considering the real-world settings, our agent
can perform two interaction actions (open, close) to manipulate objects within its rooms. A wide variety of receptacles (e.g., fridges, cabinets, and drawers) can be interacted with for our agent. Receptacles are openable objects that allow some special objects to be placed inside them. There are a total of 108 different objects in the AI2-THOR, so our 3D graph consists of $|V| = 108$ nodes.

In the AI2-THOR, the field of view of the camera is 90 degrees. The Move action moves the agent by 0.25 meters, while the Rotate action rotates it left/right by 90 degrees. Besides, the Look action tilts the camera up/down by 30 degrees. Due to the particularity of the AI2-THOR, our navigation task involves two types of targets: actionable targets and static targets (See Fig. 1). Static targets refer to objects that can be directly found through random walking. For example, the apple or bowl on the table. Actionable targets are usually hidden in some receptacles and cannot be directly found unless an open operation is taken. For example, the tomato or lettuce in the fridge.

4.2 Setup

We select the objects that can be picked up (e.g., apple) or belong to common household items (e.g., microwave) as navigation targets. We choose one nearest location as the destination for our static targets and actionable targets. Our agent is expected to take intelligent actions to find these two types of targets. To successfully navigate to actionable targets, our agent must learn which objects can be opened and infer which receptacles the target is hidden in.

Our navigation task is considered successful if the agent goes to the destination and performs the stop action. We consider the action space $A = \{move \ forward, \ move \ back, \ move \ right, \ move \ left, \ rotate \ right, \ rotate \ left, \ look \ up, \ look \ down, \ open, \ stop\}$ with ten different actions. We do not include the close action because our agent automatically applies the close operation if it does not plan to apply stop action at the next time step, when it has opened a receptacle. At the beginning of each episode, our agent is given a target image and starts from a random location. The agent requires to find the target from any possible location within its environment. In our model, we provide a target-reaching reward (10.0) upon task completion, and a terminal reward (0.01) when our agent arrives at the destination. To encourage shorter trajectories, we add a time penalty (-0.01) as an immediate reward.

We evaluate our methods in all 120 rooms of the AI2-THOR. Each room has an average of about 13 targets. Since there are few actionable targets in each room, sufficient datasets are needed to train the deep learning model. For each room category, we choose 20 rooms with the maximum number of actionable targets as training rooms. The rest of the rooms are testing rooms. Since each training room has few actionable targets, all actionable targets are used for training. The static targets left are divided into two sets for training and testing. Specifically, we select 7 targets in each training room as seen targets. The remaining static targets are as unseen targets. Moreover, in each testing room, all the static targets and actionable targets are used for evaluation. The rooms without training are called unseen scenes; otherwise, named seen scenes. In practice, our TSE module randomly samples 5 sub-targets from each explored trajectory. All sub-trajectories along with the explored trajectory are used for training, which allows the agent to learn from its failures and eventually improves sample efficiency.

4.3 Evaluation Metrics

We evaluate our methods based on two main metrics in TDVN: Success Rate (SR) and Success weighted by Path Length (SPL). SR is defined as $\frac{1}{N} \sum_{i=1}^{N} S_i$, which is the ratio of
the number of times the agent successfully navigates to the target and the total number of episodes. \( N \) is the number of episodes, and \( S_i \) is a binary indicator of success in episode \( i \). SPL is defined as \( \frac{1}{N} \sum_{i=1}^{N} S_i \frac{L_i}{\max(P_i, L_i)} \), which is recently proposed by [49] and considers both the success rate and the optimal path length. \( P_i \) represents the length of the path actually taken by the agent, and \( L_i \) is the shortest path distance from the agent’s starting position to the target in episode \( i \).

4.4 Baselines

This work compares the navigation performance with the following models:

1. **Random Walk**, the agent randomly picks an action from its action space at each time step.

2. **Imitation Learning** (IL) [50], its policy is only trained with standard behavior cloning. The demonstrations are generated by the shortest path algorithm to conduct supervised learning with the cross-entropy loss.

3. **FF A3C** [2], which uses feed-forward (FF) networks that take as input concatenated visual features from last-n frames and the target image to predict the next action. Their model consists of scene-specific layers.

4. **LSTM A3C**, which utilizes the same siamese fusion methods in [2] to get the visual features. Then the visual features are passed through a shared LSTM [41] to predict the next action.

5. **LSTM A3C+TSE**, which incorporates our TSE module into LSTM A3C. In addition to training the specified target, this model is also trained with sub-targets extracted from the exploring trajectories.

6. **Scene Priors** [18], which also utilizes a fixed graph as scene priors to produce actions, in addition to visual features.

7. **RNVN**, which is our model. It combines 3D relations into LSTM A3C+TSE. As described in Eq. 12, it learns the navigation policy according to the visual features as well as the fused graph features \( Q \) obtained from GCN.

8. **RNV A**, which is our final proposed method. It adds the attention mechanism into RNVN and produces actions based on visual features and the attention on final spatial representations.

4.5 Implementation Details

For all learning models, we report their performance after being trained with 160M frames (across all threads). We train our model with 560 threads, and each thread learns for a different target. All episodes have a maximum length of 5000 time steps for each training thread. We implement our models in Tensorflow [51] and train them on an Nvidia GeForce GTX Titan RTX GPU. We train the models with a shared RMSProp optimizer of learning rate \( 7 \times 10^{-4} \). The RMSProp decay factor \( \alpha = 0.99 \), and stability constant \( \epsilon = 0.1 \). The entropy regularization term \( \beta = 0.01 \). For evaluation, we run 100 different episodes for each target. To be fair, the initial locations of the agent are randomly chosen, and all models are evaluated using the same set. Besides, the initial location is at least 10 steps away from the target location. A testing episode ends when the agent either reaches the target location or takes the maximum number of steps. The maximum step is set to 100 for seen targets within seen scenes, and to 1000 for unseen targets within seen and unseen scenes. Since FF A3C [2] uses different policy networks for different scenes, their model lacks the generalization ability to unseen scenes unless fine-tuned. In contrast, our models and other baselines use a
Table 1 The number of parameters for each method, and processing time to pass through one million training frames

| Methods            | Parameters     | Training time |
|--------------------|----------------|---------------|
| FF A3C [2]         | $2.628 \times 10^7$ | 1.25h         |
| LSTM A3C           | $2.367 \times 10^6$ | 0.58h         |
| Scene Priors [18]  | $2.486 \times 10^6$ | 0.63h         |
| RNVN/RNVA          | $2.497 \times 10^6$ | 0.68h         |

single policy network for different scene examples. The number of parameters for the main baselines and our proposed model are given in Table 1. Consistent with [2], we also show the processing time to pass through one million training frames.

4.6 Results

Table 2 presents the results of several baselines and our methods. In order to examine if the agent can transfer its learned knowledge to unknown domains, we report three types of performance: (1) the performance of seen targets within seen scenes; (2) the performance of unseen targets within seen scenes; (3) the generalization ability of all targets in unseen scenes. Since all the actionable targets are used for training, there is no performance about unseen targets in seen scenes. Although we provide the performance of static targets and actionable targets separately, we train only one model for these two types of targets over all scenes.

Our methods outperform the baselines significantly in terms of the SR and SPL metrics from the table. We observe that poor performance is achieved when the agent applies the random walk or standard imitation learning. It indicates that the random walk is not sufficient to find our targets. In addition, it is not very useful to only learn from experiences sampled from the expert. FF A3C using scene-specific layers can learn an acceptable policy, but it is slower due to more extensive learning parameters. FF A3C seems to not converge after 160M training frames. LSTM A3C yields slightly better performance than FF A3C. The results of LSTM A3C+TSE indicate that sub-targets have a significant impact on accelerating the agent’s learning. Compared with scene priors, RNVN acquires significantly better navigation performance. It may be because of the inconsistent 3D graph built from visual gnome [7]. Furthermore, our final model, RNVA, achieves the best results, demonstrating the effectiveness of the attention mechanism for navigation. Moreover, we analyze the results in detail from the following three parts.

**Seen vs. Unseen.** We observe that all models obtain relatively high performance when tested on seen targets in seen scenes. For unseen targets in seen scenes, the models with 3D relations can achieve decent performance. The scenario, in which both targets and scenes are unseen, is more challenging. The performance degrades drastically for both baselines and our methods. However, we find that our model improves the performance on the unseen scenes by nearly 20% (static targets) and 15% (actionable targets) compared to models trained without the 3D relations.

**Static vs. Actionable.** Compared to static and actionable targets, the performance of the actionable targets is significantly lower than the static targets. There are perhaps two reasons for this situation. One is that the agent needs to infer which receptacles the finding target is possibly located. The other one is that there are many same kinds of receptacles in the
Table 2  SPL/SR (%) results on the AI2-THOR

|                     | Static targets | Actionable targets |
|---------------------|----------------|-------------------|
|                     | Seen scenes,   | Unseen targets    | Seen scenes,   |
|                     | Seen targets   |                   | Seen targets   |
|                     |                |                   | Scenes         |
| Random walk         | 2.11/3.56      | 1.29/3.84         | 1.02/2.23      |
| Imitation learning  | 3.62/8.24      | 2.17/4.39         | 1.34/4.54      |
| FF A3C [2]          | 31.16/43.33    | --/--             | 16.92/35.13    |
| LSTM A3C            | 35.04/47.29    | 5.31/9.39         | 22.82/38.6     |
| LSTM A3C+TSE        | 42.58/60.13    | 7.15/12.63        | 29.51/42.23    |
| Scene Priors [18]   | 47.25/64.53    | 8.29/18.43        | 36.25/52.75    |
| RNVN (ours)         | 51.32/75.47    | 12.18/25.56       | 37.77/67.55    |
| RNVA (ours)         | **53.58/86.44** | **16.20/34.09**  | **40.51/78.32** |

Bold numbers show the best results compared with the baselines. The number before “/” represents results on the SPL metric, otherwise on the SR metric. All the values are averaged over 100 episodes for each target.
Fig. 6 Comparison of the average number of collisions at a trajectory (blue bar), and the success rate of the target objects that is visible in the last frame (orange bar). Fewer collisions and a higher success rate indicate better performance.

Fig. 7 Example results of the attended objects predicted by our model at each time step \( t \), and the corresponding action to be taken next. The trained agent self-infers what objects it should attend on to produce its next action. We visualize top-3 objects with the highest probability that the agent reasoned at each time step.

Environment, such as many drawers and cabinets. The agent needs to search one by one, and remember which one it has explored before.

**SR vs. SPL.** We observe that the agent achieves atrocious performance when evaluated with the SPL metric. It is challenging for the agent to find the targets with the shortest path. The DRL agent needs more explorations to integrate information and make its decisions. As mentioned in [18], it is worth noting that the GCN module does not increase much GFLOPs computation compared with the baseline A3C.

### 4.7 Case Analysis

Performed in unseen scenes, Fig. 6 shows the average number of collisions, and the success rate of the last frame that contains the target objects. Most baselines collide a lot of times in the trajectory, and have a lower success rate. However, our models obtain better results. It indicates that the navigation system, established by 3D relations and attention mechanism, helps to reduce collisions and conducts target-induced exploration.
Fig. 8 The t-SNE embeddings of the total objects appeared in all kitchen rooms. We show the node features extracted by the third GCN layer and project them into 3D space.

As illustrated in Fig. 7, it shows an example of the spatial knowledge of our model in a kitchen room. At each time step $t$, our agent selects some most relative objects for producing its next action. Since our target image contains multiple objects, the agent also needs to select the most informative objects from it. We observe that our agent can adequately attend to appropriate objects, which lead to its next action just like humans. For example, at time step 1, our agent attends most on the toaster when it plans to take a look-up action. Besides, we notice that the agent tends to remind themselves what they are looking for at almost every time step.

To understand what our spatial model learns, we examine the node features learned by our GCN layers. Figure 8 shows the t-SNE [52] visualization of the node features obtained from the third GCN layer. We observe that the spatial arrangement of these node features is commonly consistent with their corresponding 3D projections. For example, vases and statues are usually placed on a shelf. Consequently, we can see that the distances between their feature spaces are also close together from the picture.

Finally, we show the impact of sub-targets on data efficiency. Our experiment was conducted with 4M training frames and 13 targets (7 static targets and 6 actionable targets) in a kitchen room. As shown in Fig. 9, LSTM A3C+TSE converges more quickly than FF A3C and LSTM A3C. It proves that our TSE module indeed makes full use of the experienced data by learning from failures, and can ameliorate the problem of sparse reward in traditional reinforcement learning. We also show RNVN, which augments with spatial information, converges faster as well.

As shown in Fig. 10, we provide more training details. We show the training curves of our proposed model about four metrics: (1) Episode Length, which shows the number of steps to find the target in each episode. (2) Episode Collision, which shows the number of collisions in each episode. (3) Episode eLook, which shows the number of errors taking the wrong $look$ action in each episode, because the allowed $look$ field is $[-60, 60]$. (4) Episode
Fig. 9  Training curves for our proposed TSE module and two baselines. LSTM A3C with sub-targets converges faster than two A3C baselines after 4M training frames.

![Graph showing training curves](image)

Fig. 10  Training curves of length, collisions, look errors, and open errors in each episode.

![Graph showing training curves](image)

eOpen, which shows the number of errors taking the wrong open action in each episode, because some objects can not be opened. There are three rows in Fig. 10, which represent three different targets. Each row shows the results of the agent finding a specific target.
5 Conclusion

This paper proposes a model to solve the problem of target-driven visual navigation. It integrates 3D relations and TSE module into a standard deep reinforcement learning framework to boost navigation performance. Our experiments, evaluated in AI2-THOR, show that spatial knowledge improves generalization ability across targets and scenes. However, despite the encouraging results, many problems remain to be improved. The main limitation of our method is that the number of nodes in the graph is fixed, we plan to denote some nodes by object categories instead of objects. In our future research, we will also conduct experiments with more complex types of spatial relationships between objects, such as on, in, above. Besides, we are interested in applying global graph, which is constructed by aggregating spatial knowledge from observations over time. Therefore, explicit way from local graph (obtained from observation) to global graph, the probabilistic model of the global graph, and the representation of node features are also our next research directions to obtain promising performance and improve generalization.

Funding The fund was provided by National Outstanding Youth Science Fund Project of National Natural Science Foundation of China.

References

1. Savva M, Kadian A, Maksymets O, Zhao Y, Wijmans E, Jain B, Straub J, Liu J, Koltun V, Malik J et al (2019) Habitat: a platform for embodied ai research. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 9339–9347
2. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: Proceedings - IEEE international conference on robotics and automation (ICRA), pp 3357–3364 (2017)
3. Anderson P, Wu Q, Teney D, Bruce J, Johnson M, Sünderhauf N, Reid I, Gould S, van den Hengel A (2018) Vision-and-language navigation: interpreting visually-grounded navigation instructions in real environments. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 3674–3683
4. Das A, Dutta S, Gkioxari G, Lee S, Parikh D, Batra D (2018) Embodied question answering. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 2054–2063
5. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 248–255
6. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: Proceedings of european conference on computer vision (ECCV), pp 740–755
7. Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li L-J, Shamma DA et al (2017) Visual genome: connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vis. 123(1):32–73
8. Chaplot DS, Sathyendra KM, Pasumarthi RK, Rajagopal D, Salakhutdinov R (2018) Gated-attention architectures for task-oriented language grounding. In: Proceedings of AAAI conference on artificial intelligence (AAAI), pp 2819–2826
9. Wu Y, Wu Y, Gkioxari G, Tian Y (2018) Building generalizable agents with a realistic and rich 3d environment.
10. Dhiman V, Banerjee S, Griffin B, Siskind JM, Corso JJ (2018) A critical investigation of deep reinforcement learning for navigation
11. Kolbe E, Mottaghi R, Han W, VanderBilt E, Weihs L, Herrasti A, Gordon D, Zhu Y, Gupta A, Farhadi A (2017) A12-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474
12. Bonin-Font F, Ortiz A, Oliver G (2008) Visual navigation for mobile robots: a survey. J. Intell. Robotic Syst. 53(3):263–296
13. Fuentes-Pacheco J, Ascencio JR, Rendón-Mancha JM (2015) Visual simultaneous localization and mapping: a survey. Artif. Intell. Rev. 43(1):55–81
14. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
15. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction. MIT Press, Cambridge
16. Mousavi SS, Schukat M, Howley E (2016) Deep reinforcement learning: an overview. In: Proceedings of the SAI intelligent systems conference, pp. 426–440. Springer
17. Zhu Y, Gordon D, Kolve E, Fox D, Fei-Fei L, Gupta A, Mottaghi R, Farhadi A (2017) Visual semantic planning using deep successor representations. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 483–492
18. Yang W, Wang X, Farhadi A, Gupta A, Mottaghi R (2019) Visual semantic navigation using scene priors. In: Proceedings of international conference on learning representations (ICLR)
19. Mei H, Bansal M, Walter MR (2016) Listen, attend, and walk: neural mapping of navigational instructions to action sequences. In: Proceedings of AAAI conference on artificial intelligence (AAAI), pp 2772–2778
20. Fried D, Hu R, Cirik V, Rohrbach A, Andreas J, Morency L-P, Berg-Kirkpatrick T, Saenko K, Klein D, Darrell T (2018) Speaker-follower models for vision-and-language navigation. In: Proceedings of the neural information processing systems (NIPS), pp 3314–3325
21. Gordon D, Kembhavi A, Rastegari M, Redmon J, Fox D, Farhadi A (2018) Iqa: visual question answering in interactive environments. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 4089–4098
22. Devo A, Mezzetti G, Costante G, Fravolini ML, Valigi P (2020) Towards generalization in target-driven visual navigation by using deep reinforcement learning. IEEE Trans. Robot. 36(5):1546–1561
23. Rao Z, Wu Y, Yang Z, Zhang W, Lu S, Lu W, Zha Z (2021) Visual navigation with multiple goals based on deep reinforcement learning. IEEE Trans Neural Netw Learn Syst 32(12):5445–5455
24. Kulkarni TD, Saeedi A, Gautham S, Gersham SJ (2016) Deep successor reinforcement learning. arXiv preprint arXiv:1606.02396
25. Tessler C, Givony S, Zahavy T, Mankowitz DJ, Mannor S (2017) A deep hierarchical approach to lifelong learning in minecraft. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 1553–1561
26. Mirowski P, Pascanu R, Viola F, Soyer H, Ballard AJ, Banino A, Denil M, Goroshin R, Sifre L, Kavukcuoglu K, et al (2017) Learning to navigate in complex environments. In: Proceedings of the international conference on learning representations (ICLR)
27. Jaderberg M, Mnih V, Czarnecki WM, Schaul T, Leibo JZ, Silver D, Kavukcuoglu K (2017) Reinforcement learning with unsupervised auxiliary tasks. In: Proceedings of international conference on learning representations (ICLR)
28. Kempka M, Wydmuch M, Runc G, Toczek J, Jaśkowski W (2016) Vizdoom: a doom-based ai research platform for visual reinforcement learning. In: Proceedings of IEEE international conference on computational intelligence and games (CIG), pp. 1–8
29. Oh J, Chockalingam V, Lee H, et al (2016) Control of memory, active perception, and action in minecraft. In: Proceedings of the international conference on machine learning (ICML), pp 2790–2799
30. Beattie C, Leibo JZ, Teplyashin D, Ward T, Wainwright M, Küttler H, Lefrancq A, Green S, Valdés V, Sadik A, et al (2016) Deepmind lab
31. Chaplot DS, Lample G, Sathyendra KM, Salakhutdinov R (2016) Transfer deep reinforcement learning in 3D environments: An empirical study. In: Proceedings of the international conference on neural information processing systems (NIPS)
32. Pariset C, Salakhutdinov R (2018) Neural map: structured memory for deep reinforcement learning. In: Proceedings of international conference on learning representations (ICLR)
33. Oh J, Singh S, Lee H, Kohli P (2017) Zero-shot task generalization with multi-task deep reinforcement learning. In: Proceedings of the international conference on machine learning (ICML), pp 2661–2670
34. Pathak D, Mahmoodieh P, Luo G, Agrawal P, Chen D, Shentu Y, Shellhammer E, Malik J, Efros AA, Darrell T (2018) Zero-shot visual imitation. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 2050–2053
35. Wortsman M, Ehsani K, Rastegari M, Farhadi A, Mottaghi R (2019) Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 6750–6759
36. Fang Q, Xu X, Wang X, Zeng Y (2021) Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning. CAAI T. Intell, Technol
37. Song S, Lu F, Zeng A, Chang AX, Savma M, Funkhouser T (2017) Semantic scene completion from a single depth image. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 1746–1754
38. Chang A, Dai A, Funkhouser TA, Halber M, Niebner M, Savma M, Song S, Zeng A, Zhang Y (2018) Matterport3d: Learning from rgb-d data in indoor environments. In: Proceedings of international conference on 3D vision (3DV), pp 667–676
39. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of international conference on learning representations (ICLR)

40. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of international conference on machine learning (ICML), pp 1928–1937

41. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput. 9(8):1735–1780

42. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 770–778

43. Chopra S, Hadsell R, LeCun Y, et al (2005) Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 539–546

44. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement

45. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Proceedings of the neural information processing systems (NIPS), pp 5998–6008

46. Zhang H, Xiao Z, Wang J, Li F, Szczerbicki E (2019) A novel iot-perceptive human activity recognition (har) approach using multihead convolutional attention. IEEE Internet Things J. 7(2):1072–1080

47. Xiao Z, Xu X, Xing H, Luo S, Dai P, Zhan D (2021) Rtfn: a robust temporal feature network for time series classification. Inf. Sci. 571:65–86

48. Andrychowicz M, Wolski F, Ray A, Schneider J, Fong R, Welinder P, McGrew B, Tobin J, Abbeel OP, Zaremba W (2017) Hindsight experience replay. In: Proceedings of the neural information processing systems (NIPS), pp 5048–5058

49. Anderson P, Chang A, Chaplot DS, Dosovitskiy A, Gupta S, Koltun V, Kosecka J, Malik J, Mottaghi R, Savva M, et al (2018) On evaluation of embodied navigation agents

50. Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, et al (2016) End to end learning for self-driving cars

51. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, et al (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems

52. Lvd Maaten, Hinton G (2008) Visualizing data using t-sne. J. Mach. Learn. Res. 9(Nov):2579–2605

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.