Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning

Yuke Zhu[1] Roozbeh Mottaghi[2] Eric Kolve[2] Joseph J. Lim[1,5]
Abhinav Gupta[2,3] Li Fei-Fei[1] Ali Farhadi[2,4]

***Abstract*— Two less addressed issues of deep reinforcement learning are (1) lack of generalization capability to new goals, and (2) data inefficiency, i.e., the model requires several (and often costly) episodes of trial and error to converge, which makes it impractical to be applied to real-world scenarios. In this paper, we address these two issues and apply our model to *target-driven visual navigation*. To address the first issue, we propose an actor-critic model whose policy is a function of the goal as well as the current state, which allows better generalization. To address the second issue, we propose the AI2-THOR framework, which provides an environment with high-quality 3D scenes and a physics engine. Our framework enables agents to take actions and interact with objects. Hence, we can collect a huge number of training samples efficiently.**

We show that our proposed method (1) converges faster than the state-of-the-art deep reinforcement learning methods, (2) generalizes across targets and scenes, (3) generalizes to a real robot scenario with a small amount of fine-tuning (although the model is trained in simulation), (4) is end-to-end trainable and does not need feature engineering, feature matching between frames or 3D reconstruction of the environment.

I. Introduction

Many tasks in robotics involve interactions with physical environments and objects. One of the fundamental components of such interactions is understanding the correlation and causality between actions of an agent and the changes of the environment as a result of the action. Since the 1970s, there have been various attempts to build a system that can understand such relationships. Recently, with the rise of deep learning models, learning-based approaches have gained wide popularity [1], [2].

In this paper, we focus on the problem of navigating a space to find a given target using only visual input. Successful navigation requires learning relationships between actions and the environment. This makes the task well suited to a Deep Reinforcement Learning (DRL) approach. However, general DRL approaches (e.g., [2], [3]) are designed to learn a policy that depends only on the current state, and the goal is implicitly embedded in the model parameters. Hence, it is necessary to learn new model parameters for a new target. This is problematic since training DRL agents is computationally expensive.

In order to achieve higher *adaptability* and *flexibility*, we introduce a *target-driven model*. Our model takes the visual task objective as an input. Thus we can avoid re-training for every new target. Our model learns a policy that jointly embeds the target and the current state. Essentially, an agent learns to take its next action conditioned on both its current state and target, rather than its current state only. Hence, there is no need to re-train the model for new targets. A key intuition that we rely on is that different training episodes share information. For example, agents explore common routes during the training stage while being trained for finding different targets. Various scenes also share similar structures and statistics (e.g., a fridge is likely to be near a microwave). In short, we exploit the fact that learning for new targets will be easier with the models that have been trained for other targets.

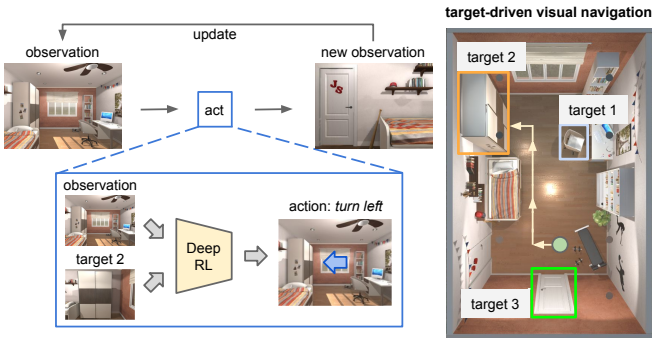

Fig. 1. The goal of our deep reinforcement learning model is to navigate towards a visual target with a minimum number of steps. Our model takes the current observation and the image of the target as input and generates an action in the 3D environment as the output. Our model learns to navigate to different targets in a scene without re-training.

Unfortunately, training and quantitatively evaluating DRL algorithms in real environments is often tedious. One reason is that running systems in a physical space is time consuming. Furthermore, acquiring large-scale action and interaction data in real environments is not trivial via the common image dataset collection techniques. To this end, we developed one of the first simulation frameworks with high-quality 3D scenes, called The House Of inteRactions (AI2-THOR). Our simulation framework enables us to collect a large number of visual observations for action and reaction in different environments. For example, an agent can freely navigate (i.e. move and rotate) in various realistic indoor scenes, and is able to have low- and high-level interactions with the objects (e.g., applying a force or opening/closing a microwave).

We evaluate our method for the following tasks: (1) *Target generalization*, where the goal is to navigate to targets that have not been used during training within a scene. (2) *Scene generalization*, where the goal is to navigate to targets in

*This work was done while the first author was an intern at AI2. [1]Stanford University, [2]Allen Institute for AI, [3]Carnegie Mellon University, [4]University of Washington, [5]University of Southern California.

scenes not used for training. (3) *Real-world generalization*, where we demonstrate navigation to targets using a real robot. Our experiments show that we outperform the state-of-the-art DRL methods in terms of data efficiency for training. We also demonstrate the generalization aspects of our model.

In summary, we introduce a novel reinforcement learning model that generalizes across targets and scenes. To learn and evaluate reinforcement learning models, we create a simulation framework with high-quality rendering that enables visual interactions for agents. We also demonstrate real robot navigation using our model generalized to the real world with a small amount of fine-tuning.

II. RELATED WORK

There is a large body of work on *visual* navigation. We provide a brief overview of some of the relevant work. The map-based navigation methods require a global map of the environment to make decisions for navigation (e.g., [4], [5], [6], [7]). One of the main advantages of our method over these approaches is that it does not need a prior map of the environment. Another class of navigation methods reconstruct a map on the fly and use it for navigation [8], [9], [10], [11], [12], or go through a training phase guided by humans to build the map [13], [14]. In contrast, our method does not require a map of the environment, as it does not have any assumption on the landmarks of the environment, nor does it require a human-guided training phase. Map-less navigation methods are common as well [15], [16], [17], [18], [19]. These methods mainly focus on obstacle avoidance given the input image. Our method is considered *map-less*. However, it possesses implicit knowledge of the environment. A survey of visual navigation methods can be found in [20].

Note that our approach is not based on feature matching or 3D reconstruction, unlike, e.g., [21], [22]. Besides, our method does not require supervised training for recognizing distinctive landmarks, unlike, e.g., [23], [24].

Reinforcement Learning (RL) has been used in a variety of applications. [25] propose a policy gradient RL approach for locomotion of a four-legged robot. [26] discuss policy gradient methods for learning motor primitives. [27] propose an RL-based method for obstacle detection using a monocular camera. [28] apply reinforcement learning to autonomous helicopter flight. [29] use RL to automate data collection process for mapping. [30] propose a kernel-based reinforcement learning algorithm for large-scale settings. [31] use RL for making decisions in ATARI games. In contrast to these approaches, our models use deep reinforcement learning to handle high-dimensional sensory inputs.

Recently, methods that integrate deep learning methods with RL have shown promising results. [2] propose deep Q-networks to play ATARI games. [32] propose a new search algorithm based on the integration of Monte-Carlo tree search with deep RL that beats the world champion in the game of Go. [3] propose a deep RL approach, where the parameters of the deep network are updated by multiple asynchronous copies of the agent in the environment. [1] use a deep RL approach to directly map the raw images into torques at robot motors. Our work deals with much more complex inputs than ATARI games, or images taken in a lab setting with a constrained background. Additionally, our method is generalizable to new scenes and new targets, while the mentioned methods should be re-trained for a new game, or in case of a change in the game rules.

Fig. 2. Screenshots of our framework and other simulated learning frameworks: ALE [39], ViZDoom [40], UETorch [41], Project Malmo [42], SceneNet [43], TORCS [44], SYNTHIA [45], Virtual KITTI [46].

There have been some effort to develop learning methods that can generalize to different target tasks [33], [34]. Similar to UVFA [35], our model takes the target goal directly as an input without the need of re-training.

Recently, physics engines have been used to learn the dynamics of real-world scenes from images [36], [37], [38]. In this work, we show that a model that is trained in simulation can be generalized to real-world scenarios.

III. THE AI2-THOR FRAMEWORK

To train and evaluate our model, we require a framework for performing actions and perceiving their outcomes in a 3D environment. Integrating our model with different types of environments is a main requirement for generalization of our model. Hence, the framework should have a plug-n-play architecture such that different types of scenes can be easily incorporated. Additionally, the framework should have a detailed model of the physics of the scene so the movements and object interactions are properly represented.

For this purpose, we propose The House Of inteRactions (AI2-THOR) framework, which is designed by integrating a physics engine (Unity 3D) with a deep learning framework (Tensorflow [47]). The general idea is that the rendered images of the physics engine are streamed to the deep learning framework, and the deep learning framework issues a control command based on the visual input and sends it back to the agent in the physics engine. Similar frameworks have been proposed by [39], [40], [44], [42], [41], but the

main advantages of our framework are as follows: (1) The physics engine and the deep learning framework directly communicate (in contrast to separating the physics engine from the controller as in [38]). Direct communication is important since the feedback from the environment can be immediately used for online decision making. (2) We tried to mimic the appearance distribution of the real-world images as closely as possible. For example, [39] work on Atari games, which are 2D environments and limited in terms of appearance. [43] is a collection of synthetic scenes that are non-photo-realistic and do not follow the distribution of real-world scenes in terms of lighting, object appearance, textures, and background clutter, etc. This is important for enabling us to generalize to real-world images.

To create indoor scenes for our framework, we provided reference images to artists to create a 3D scene with the texture and lighting similar to the image. So far we have 32 scenes that belong to 4 common scene types in a household environment: kitchen, living room, bedroom, and bathroom. On average, each scene contains 68 object instances.

The advantage of using a physics engine for modeling the world is that it is highly scalable (training a robot in real houses is not easily scalable). Furthermore, training the models can be performed cheaper and safer (e.g., the actions of the robot might damage objects). One main drawback of using synthetic scenes is that the details of the real world are under-modeled. However, recent advances in the graphics community make it possible to have a rich representation of the real-world appearance and physics, narrowing the discrepancy between real world and simulation. Fig. 2 provides a qualitative comparison between a scene in our framework and example scenes in other frameworks and datasets. As shown, our scenes better mimic the appearance properties of real world scenes. In this work, we focus on navigation, but the framework can be used for more fine-grained physical interactions, such as applying a force, grasping, or object manipulations such as opening and closing a microwave. Fig. 3 shows a few examples of high-level interactions. We will provide Python APIs with our framework for an AI agent to interact with the 3D scenes.

IV. TARGET-DRIVEN NAVIGATION MODEL

In this section, we first define our formulation for target-driven visual navigation. Then we describe our deep siamese actor-critic network for this task.

A. Problem Statement

Our goal is to find the minimum length sequence of actions that move an agent from its current location to a target that is specified by an RGB image. We develop a deep reinforcement learning model that takes as input an RGB image of the current observation and another RGB image of the target. The output of the model is an action in 3D such as move forward or turn right. Note that the model learns a mapping from the 2D image to an action in the 3D space.

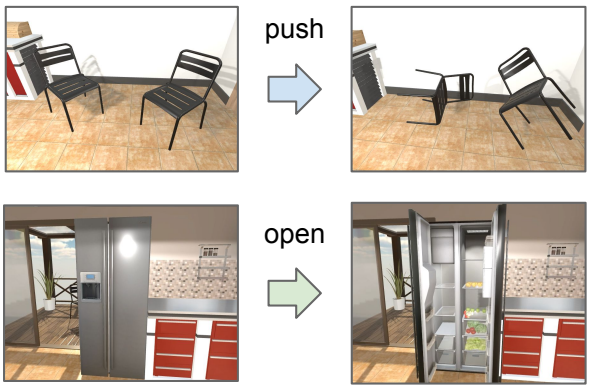

Fig. 3. Our framework provides a rich interaction platform for AI agents. It enables physical interactions, such as pushing or moving objects (the first row), as well as object interactions, such as changing the state of objects (the second row).

B. Problem Formulation

Vision-based robot navigation requires a mapping from sensory signals to motion commands. Previous work on Reinforcement Learning typically do not consider high-dimensional perceptual inputs [48]. Recent deep reinforcement learning (DRL) models [2] provide an end-to-end learning framework for transforming pixel information into actions. However, DRL has largely focused on learning goal-specific models that tackle individual tasks in isolation. This training setup is rather inflexible to changes in task goals. For instance, as pointed out by Lake et al. [49], changing the rule of the game would have devastating performance impact on DRL-based Go-playing systems [32]. Such limitation roots from the fact that standard DRL models [2], [3] aim at finding a direct mapping (represented by a deep neural network π) from state representations s to policy $\pi(s)$. In such cases, the goal is hardcoded in neural network parameters. Thus, changes in goals would require to update the network parameters in accordance.

Such limitation is especially problematic for mobile robot navigation. When applying DRL to the multiple navigation targets, the network should be re-trained for each target. In practice, it is prohibitive to exhaust every target in a scene. This is the problem caused by a lack of generalization – i.e., we would have to re-train a new model when incorporating new targets. Therefore, it is preferable to have a single navigation model, which learns to navigate to new targets without re-training. To achieve this, we specify the task objective (i.e., navigation destination) as inputs to the model, instead of implanting the target in the model parameters. We refer to this problem as *target-driven visual navigation*. Formally, the learning objective of a target-driven model is to learn a stochastic policy function π which takes two inputs, a representation of current state s_t and a representation of target g and produces a probability distribution over the action space $\pi(s_t, g)$. For testing, a mobile robot keeps taking actions drawn from the policy distribution until reaching the destination. This way, actions are conditioned on both states and targets. Hence, no re-training for new targets is required.

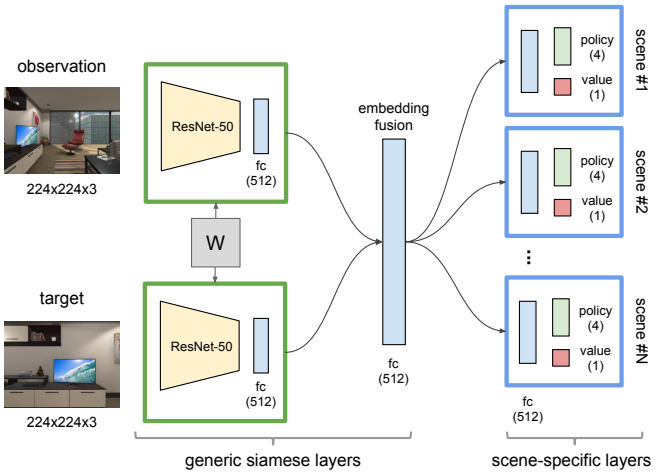

Fig. 4. Network architecture of our deep siamese actor-critic model. The numbers in parentheses show the output dimensions. Layer parameters in the green squares are shared. The ResNet-50 layers (yellow) are pre-trained on ImageNet and fixed during training.

C. Learning Setup

Before introducing our model, we first describe the key ingredients of the reinforcement learning setup: action space, observations and goals, and reward design.

1) Action space: Real-world mobile robots have to deal with low-level mechanics. However, such mechanical details make the learning significantly more challenging. A common approach is to learn at a certain level of abstraction, where the underlying physics is handled by a lower-level controller (e.g., 3D physical engine). We train our model with command-level actions. For our visual navigation tasks, we consider four actions: moving forward, moving backward, turning left, and turning right. We use a constant step length (0.5 meters) and turning angle (90 degree). This essentially discretizes the scene space into a *grid-world* representation. To model uncertainty in real-world system dynamics, we add a Gaussian noise to steps $\mathcal{N}(0, 0.01)$ and turns $\mathcal{N}(0, 1.0)$ at each location.

2) Observations and Goals: Both observations and goals are images taken by the agent's RGB camera in its first-person view. The benefit of using images as goal descriptions is the flexibility for specifying new targets. Given a target image, the task objective is to navigate to the location and viewpoint where the target image is taken.

3) Reward design: We focus on minimizing the trajectory length to the navigation targets, although other factors such as energy efficiency could be considered. We only provide a goal-reaching reward (10.0) upon task completion. To encourage shorter trajectories, we add a small time penalty (-0.01) as immediate reward.

D. Model

We focus on learning the target-driven policy function π via deep reinforcement learning. We design a new deep neural network as a non-linear function approximator for π, where action a at time t can be drawn by:

$$a \sim \pi(s_t, g \,|\, \mathbf{u}) \tag{1}$$

where $\mathbf{u}$ are the model parameters, s_t is the image of the current observation, and g is the image of the navigation target. When target g belongs to a finite discrete set, π can be seen as a mixture model, where g indexes the right set of parameters for each goal. However, the number of real-world goals is often countless (due to many different locations or highly variable object appearances). Thus, it is preferable to learn a projection that transforms the goals into an embedding space. Such projection enables knowledge transfer across this embedding space, and therefore allows the model to generalize to new targets.

Navigational decisions demand an understanding of the relative spatial positions between the current locations and the target locations, as well as a holistic sense of scene layout. We develop a new deep siamese actor-critic network to capture such intuitions. Fig. 4 illustrates our model for the target-driven navigation tasks. Overall, the inputs to the network are two images that represent the agent's current observation and the target. Our approach to reasoning about the spatial arrangement between the current location and the target is to project them into the same embedding space, where their geometric relations are preserved. Deep siamese networks are a type of two-stream neural network models for discriminative embedding learning [50]. We use two streams of weight-shared siamese layers to transform the current state and the target into the same embedding space. Information from both embeddings is fused to form a joint representation. This joint representation is passed through *scene-specific* layers (refer to Fig. 4). The intention to have scene-specific layers is to capture the special characteristics (e.g., room layouts and object arrangements) of a scene that are crucial for the navigation tasks. Finally, the model generates policy and value outputs similar to the advantage actor-critic models [3]. In this model, targets across all scenes share the same generic siamese layers, and all targets within a scene share the same scene-specific layer. This makes the model better generalize across targets and across scenes.

E. Training Protocol

Traditional RL models learn for individual tasks in separation, resulting in the inflexibility with respect to goal changes. As our deep siamese actor-critic network shares parameters across different tasks, it can benefit from learning with multiple goals simultaneously. A3C [3] is a type of reinforcement learning model that learns by running multiple copies of training threads in parallel and updates a shared set of model parameters in an asynchronous manner. It has been shown that these parallel training threads stabilize each other, achieving the state-of-the-art performance in the video-game domain. We use a similar training protocol as A3C. However, rather than running copies of a single game, each thread runs with a different navigation target. Thus, gradients are backpropagated from the actor-critic outputs back to the lower-level layers. The scene-specific layers are updated by

gradients from the navigation tasks within the scene, and the generic siamese layers are updated by all targets.

F. Network Architectures

The bottom part of the siamese layers are ImageNet-pretrained ResNet-50 [51] layers (truncated the softmax layer) that produce 2048-d features on a $224 \times 224 \times 3$ RGB image. We freeze these ResNet parameters during training. We concatenate features of 4 history frames to account for the agent's past motions. These 8192-d output vectors from both streams are projected into the 512-d embedding space. The fusion layer takes a 1024-d concatenated embedding of the state and the target, generating a 512-d joint representation. This vector is further passed through two fully-connected scene-specific layers, producing 4 policy outputs (i.e., probability over actions) and a single value output. We train this network with a shared RMSProp optimizer of learning rate 7×10^{-4}.

V. EXPERIMENTS

Our main objective for target-driven navigation is to find the shortest trajectories from the current location to the target. We first evaluate our model with baseline navigation models that are based on heuristics and standard deep RL models. One major advantage of our proposed model is the ability to generalize to new scenes and new targets. We conduct two additional experiments to evaluate the ability of our model to transfer knowledge across targets and across scenes. Also, we show an extension of our model to continuous space. Lastly, we demonstrate the performance of our model in a complex real setting using a real robot.

A. Navigation Results

We implement our models in Tensorflow [47] and train them on an Nvidia GeForce GTX Titan X GPU. We follow the training protocol described in Sec. IV-E to train our deep siamese actor-critic model (see Fig. 4) with 100 threads, each thread learns for a different target. It takes around 1.25 hours to pass through one million training frames across all threads. We report the performance as the average number of steps (i.e., average trajectory length) it takes to reach a target from a random starting point. The navigation performance is reported on 100 different goals randomly sampled from 20 indoor scenes in our dataset. We compare our final model with heuristic strategies, standard deep RL models, and variations of our model. The models we compare are:

1) **Random walk** is the simplest heuristic for navigation. In this baseline model, the agent randomly draws one out of four actions at each step.
2) **Shortest Path** provides an upper-bound performance for our navigation model. As we discretize the walking space by a constant step length (see Sec. IV-C), we can compute the shortest paths from the starting locations to the target locations. Note that for computing the shortest path, we have access to the full map of the environment, while the input to our system is just an RGB image.

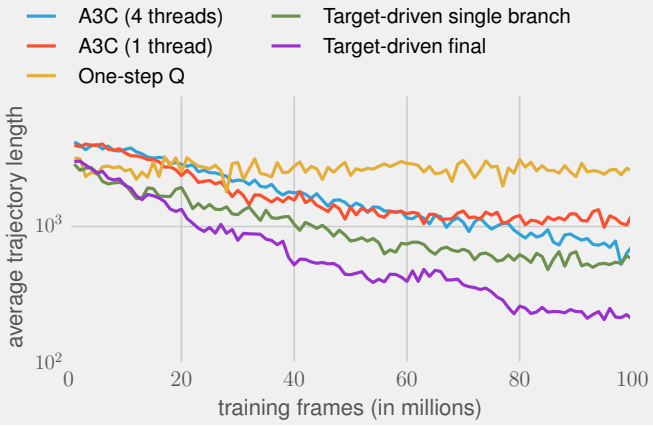

Fig. 5. **Data efficiency of training.** Our model learns better navigation policies compared to the state-of-the-art A3C methods [3] after 100M training frames.

3) **A3C** [3] is an asynchronous advantage actor-critic model that achieves the state-of-the-art results in Atari games. Empirical results show that using more threads improves the data efficiency during training. We thus evaluate A3C model in two setups, where we use 1 thread and 4 threads to train for each target.
4) **One-step Q** [3] is an asynchronous variant of deep Q-network [2].
5) **Target-driven single branch** is a variation of our deep siamese model that does not have scene-specific branches. In this case, all targets will use and update the same scene-specific parameters, including two FC layers and the policy/value output layers.
6) **Target-driven final** is our deep siamese actor-critic model introduced in Sec. IV-D.

For all learning models, we report their performances after being trained with 100M frames (across all threads). The performance is measured by the average trajectory length (i.e., number of steps taken) over all targets. An episode ends when either the agent reaches the target, or after it takes 10,000 steps. For each target, we randomly initialize the agent's starting locations, and evaluate 10 episodes. The results are listed in Table I.

TABLE I
PERFORMANCE OF TARGET-DRIVEN METHODS AND BASELINES

Type	Method	Avg. Trajectory Length
Heuristic	Random walk	2744.3
	Shortest path	17.6
Purpose-built RL	One-step Q	2539.2
	A3C (1 thread)	1241.3
	A3C (4 threads)	723.5
Target-driven RL (Ours)	Single branch	581.6
	Final	**210.7**

We analyze the data efficiency of several deep RL models with the learning curves in Fig. 5. Q-learning suffers from slow convergence. A3C performs better than Q-learning; plus, increasing the number of actor-learning threads per

target from 1 to 4 improves learning efficiency. Our proposed target-driven navigation model significantly outperforms standard deep RL models with 100M frames for training. We hypothesize that this is because both the weight sharing scheme across targets and the asynchronous training protocol facilitate learning generalizable knowledge. In contrast, purpose-built RL models are less data-efficient, as there is no straightforward mechanism to share information across different scenes or targets. The average trajectory length of the final model is three times shorter than the one of the *single branch* model. It justifies the use of scene-specific layers, as it captures particular characteristics of a scene that may vary across scene instances.

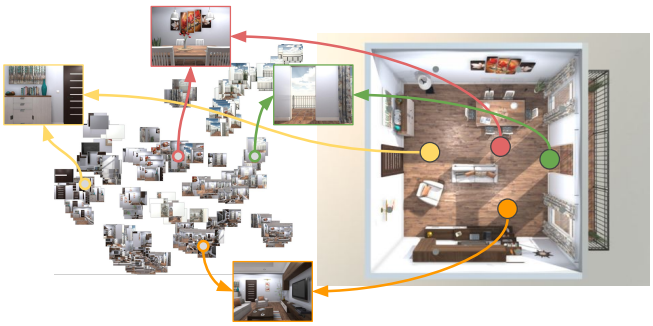

Fig. 6. **t-SNE embeddings of observations in a living room scene.** We highlight four observation examples in the projected 2D space and their corresponding locations in the scene (bird's-eye view on the right). This figure shows that our model has learned observation embeddings while preserving their relative spatial layout.

To understand what the model learns, we examine the embeddings learned by generic siamese layers. Fig. 6 shows t-SNE visualization [52] of embedding vectors computed from observations at different locations at four different orientations. We observe notable spatial correspondence between the spatial arrangement of these embedding vectors and their corresponding t-SNE projections. We therefore hypothesize that the model learns to project observation images into the embedding space while preserving their spatial configuration. To validate this hypothesis, we compare the distance of pairwise projected embeddings and the distance of their corresponding scene coordinates. The Pearson correlation coefficient is 0.62 with p-value less than 0.001, indicating that the embedding space preserves information of the original locations of observations. This means that the model learns a rough map of the environment and has the capability of localization with respect to this map.

B. Generalization Across Targets

In addition to the data efficiency of our target-driven RL models, it has the built-in ability to generalize, which is a significant advantage over the purpose-built baseline models. We evaluate its generalization ability in two aspects: 1. generalizing to new targets within a scene, and 2. generalizing to new scenes. We focus on generalization across targets in this section, and explain scene generalization in the next section.

We test the model to navigate to new targets. These targets are not trained, but they might share common routes with

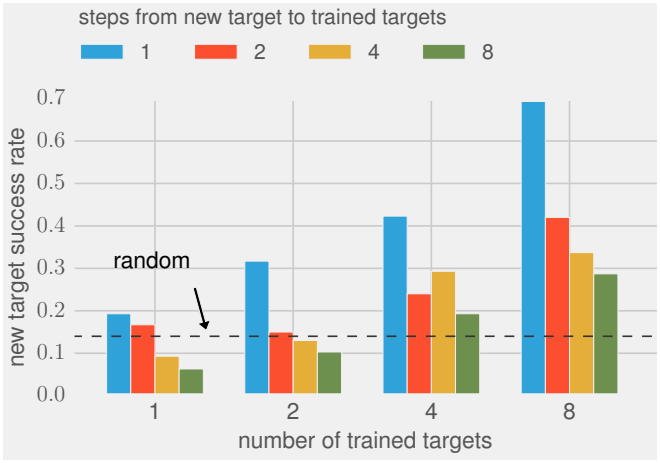

Fig. 7. **Target generalization.** Each histogram group reports the success rate of navigation to new targets with certain number of trained targets. The four bars in each group indicate the impact of adjacency between the trained and new targets on generalization performance.

trained targets, thus allowing knowledge to transfer. We take 10 of the largest scenes in our dataset, each having around 15 targets. We gradually increase the number of trained targets (from 1, 2, 4 to 8) using our target-driven model. All models are trained with 20M frames. During testing, we run 100 episodes for each of 10 new targets. These new targets are randomly chosen from a set of locations that have a constant distance (1, 2, 4 and 8 steps) from the nearest trained targets. The results are illustrated in Fig. 7. We use *success rate* (percentage of trajectories shorter than 500 steps) to measure the performance. We choose this metric due to the bipolar behavior of our model on new targets – it either reaches the new targets quickly, or fails completely. Thus, this metric is more effective than average trajectory lengths. In Fig. 7, we observe a consistent trend of increasing success rate, as we increase the number of trained targets (x-axis). Inside each histogram group, the success rate positively correlates with adjacency between trained and new targets. It indicates that the model has a clearer understanding of nearby regions around the trained targets than distant locations.

C. Generalization Across Scenes

We further evaluate our model's ability to generalize across scenes. As the generic siamese layers are shared over all scenes, we examine the possibility of transferring knowledge from these layers to new scenes. Furthermore, we study how the number of trained scenes would influence the transferability of generic layer parameters. We gradually increase the number of trained scenes from 1 to 16, and test on 4 unseen scenes. We select 5 random targets from each scene for training and testing. To adapt to unseen scenes, we train the scene-specific layers while fixing generic siamese layers. Fig. 8 shows the results. We observe faster convergence as the number of trained scenes grows. Compared to training from scratch, transferring generic layers significantly improves data efficiency for learning in new environments. We also evaluate the *single branch* model in the same setup.

As the *single branch* model includes a single scene-specific layer, we can apply a trained model (trained on 16 scenes) to new scenes without extra training. However, it results in worse performance than chance, indicating the importance of adapting scene-specific layers. The *single branch* model leads to slightly faster convergence than training from scratch, yet far slower than our final model.

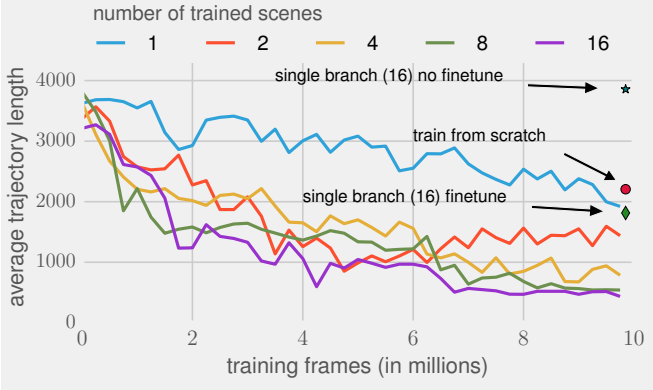

Fig. 8. **Scene generalization.** We compare the data efficiency for adapting trained navigation models to unseen scenes. As the number of trained scene instances increases, fine-tuning the scene-specific layers becomes faster.

D. Continuous Space

The space discretization eliminates the need for handling complex system dynamics, such as noise in motor control. In this section, we show empirical results that the same model is capable of coping with more challenging continuous space.

To illustrate this, we train the same target-driven model for a door-finding task in a large living room scene, where the goal is to arrive at the balcony through a door. We use the same 4 actions as before (see Sec. IV-C); however, the agent's moves and turns are controlled by the physics engine. In this case, the method should explicitly handle forces and collisions, as the agent may be stopped by obstacles or slide along heavy objects. Although this setting requires significantly more training frames (around 50M) to train for a single target, the same model learns to reach the door in average 15 steps, whereas random agents take 719 steps on average. We provide sample test episodes in the video.[1]

E. Robot Experiment

To validate the generalization of our method to real world settings, we perform an experiment by using a SCITOS mobile robot modified by [53] (see Fig. 9). We train our model in three different settings: 1) training on real images from scratch; 2) training only scene-specific layers while freezing generic layer parameters trained on 20 simulated scenes; and 3) training scene-specific layers and fine-tuning generic layer parameters.

We train our model (with backward action disabled) on 28 discrete locations in the scene, which are roughly 30 inches apart from each other in each dimension. At each location,

[1]Video link: `https://youtu.be/SmBxMDiOrvs`

Fig. 9. **Robot experiment setup.** Our experiments are conducted on a SCITOS mobile robot. On the left, we show a picture of the SCITOS robot. On the right, we show the test environment and one target (microwave) that we have used for evaluation.

the robot takes 4 RGB images (90 degrees apart) using its head camera. During testing, the robot moves and turns based on the model's predictions. We evaluate the robot with two targets in the room: *door* and *microwave*. Although the model is trained on a discretized space, it exhibits robustness towards random starting points, noisy dynamics, varying step lengths, changes in illumination and object layouts, etc. Example test episodes are provided in the video. All three setups converge to nearly-optimal policy due to the small scale of the real scene. However, we find that transferring and fine-tuning parameters from simulation to real data offers the fastest convergence out of these three setups (44% faster than training from scratch). This provides supportive evidence on the value of simulations in learning real-world interactions and shows the possibility of generalization from simulation to real images using a small amount of fine-tuning.

VI. CONCLUSIONS

We proposed a deep reinforcement learning (DRL) framework for target-driven visual navigation. The state-of-the-art DRL methods are typically applied to video games and environments that do not mimic the distribution of natural images. This work is a step towards more realistic settings.

The state-of-the-art DRL methods have some limitations that prevent them from being applied to realistic settings. In this work we have addressed some of these limitations. We addressed generalization across targets and across scenes, improved data efficiency compared to the state-of-the-art DRL methods, and provided the AI2-THOR framework that enables inexpensive and efficient collection of action and interaction data.

Our experiments showed that our method generalizes to new targets and scenes that are not used during the end-to-end training of the model. We also showed our method converges with much fewer training samples compared to the state-of-the-art DRL methods. Furthermore, we showed that the method works in both discrete and continuous domains. We also showed that a model that is trained on simulation can be adapted to a real robot with a small amount of fine-tuning. We provided visualizations that show that our DRL method implicitly performs localization and mapping. Finally, our

method is end-to-end trainable. Unlike the common visual navigation methods, it does not require explicit feature matching or 3D reconstruction of the environment.

Our future work includes increasing the number of high-quality 3D scenes in our framework. We also plan to evaluate our model in environments with longer distances and dynamic changes, and to build models that learn the physical interactions and object manipulations in the framework.

ACKNOWLEDGEMENTS

We would like to thank Dieter Fox for his helpful comments, Andrzej Pronobis and Yu Xiang for helping us with the robot experiments, and Noah Siegel for his help with creating the video.

REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *JMLR*, 2016.
[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
[3] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.
[4] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. on Systems, Man and Cybernetics*, 1989.
[5] ——, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. on Robotics and Automation*, 1991.
[6] D. Kim and R. Nevatia, "Simbolic navigation with a generic map," in *IEEE Workshop on Vision for Robots*, 1995.
[7] G. Oriolo, G. Ulivi, and M. Vendittelli, "On-line map building and navigation for autonomous mobile robots," in *ICRA*, 1995.
[8] F. Dayoub, T. Morris, B. Upcroft, and P. Corke, "Vision-only autonomous navigation using topometric maps," in *IROS*, 2013.
[9] R. Sim and J. J. Little, "Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters," in *IROS*, 2006.
[10] D. Wooden, "A guide to vision-based map building," *IEEE Robotics and Automation Magazine*, 2006.
[11] A. J. Davison, "Real time simultaneous localisation and mapping with a single camera," in *ICCV*, 2003.
[12] M. Tomono, "3-d object map building using dense object models with sift-based recognition features," in *IROS*, 2006.
[13] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human guided experience," *Robotics and Autonomous Systems*, 2002.
[14] E. Royer, J. Bom, M. Dhome, B. Thuillot, M. Lhuillier, and F. Marmoiton, "Outdoor autonomous navigation using monocular vision," in *IROS*, 2005.
[15] A. De, K. S. Bayer, and D. E. Koditschek, "Active sensing for dynamic, non-holonomic, robust visual servoing," in *ICRA*, 2014.
[16] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila, "Reactive navigation in outdoor environments using potential fields," in *ICRA*, 1998.
[17] S. Lenser and M. Veloso, "Visual sonar: Fast obstacle avoidance using monocular vision," in *IROS*, 2003.
[18] A. Remazeilles, F. Chaumette, and P. Gros, "Robot motion control from a visual memory," in *ICRA*, 2004.
[19] P. Saeedi, P. D. Lawrence, and D. G. Lowe, "Vision-based 3-d trajectory tracking for unknown environments," *T-RO*, 2006.
[20] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *J. of Intelligent and Robotic Systems*, 2008.
[21] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *Intl. J. of Robotics Research*, 2010.
[22] S. Phillips, A. Jaegle, and K. Daniilidis, "Fast, robust, continuous monocular egomotion computation," in *ICRA*, 2016.
[23] C. McManus, B. Upcroft, and P. Newman, "Scene signatures: Localised and point-less features for localisation," in *RSS*, 2014.
[24] C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *ICRA*, 2016.
[25] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *ICRA*, 2004.
[26] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, 2008.
[27] J. Michels, A. Saxena, and A. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *ICML*, 2005.
[28] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, "Autonomous helicopter flight via reinforcement learning," in *NIPS*, 2004.
[29] T. Kollar and N. Roy, "Trajectory optimization using reinforcement learning for map exploration," *Intl. J. of Robotics Research*, 2008.
[30] A. M. S. Barreto, D. Precup, and J. Pineau, "Practical kernel-based reinforcement learning," *JMLR*, 2016.
[31] Y. Liang, M. C. Machado, E. Talvitie, and M. Bowling, "State of the art control of atari games using shallow reinforcement learning," in *AAMAS*, 2016.
[32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.
[33] A. A. Rusu, S. G. Colmenarejo, Ç. Gülçehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," in *ICLR*, 2016.
[34] E. Parisotto, L. J. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," in *ICLR*, 2016.
[35] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *ICML*, 2015.
[36] J. Wu, I. Yildirim, J. J. Lim, W. T. Freeman, and J. B. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *NIPS*, 2015.
[37] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian image understanding: Unfolding the dynamics of objects in static images," in *CVPR*, 2016.
[38] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi, ""What happens if..." learning to predict the effect of forces in images," in *ECCV*, 2016.
[39] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
[40] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jakowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," in *CIG*, 2016.
[41] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example," in *ICML*, 2016.
[42] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The Malmo platform for artificial intelligence experimentation," in *IJCAI*, 2016.
[43] A. Handa, V. Patraucean, S. Stent, and R. Cipolla, "Scenenet: An annotated model generator for indoor scene understanding," in *ICRA*, 2016.
[44] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, the open racing car simulator, v1.3.5," http://www.torcs.org, 2013.
[45] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *CVPR*, 2016.
[46] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.
[47] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[48] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The Intl. J. of Robotics Research*, 2013.
[49] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, 2016.
[50] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.
[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[52] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *JMLR*, 2008.
[53] M. J.-Y. Chung, A. Pronobis, M. Cakmak, D. Fox, and R. P. N. Rao, "Autonomous question answering with mobile robots in human-populated environments," in *IROS*, 2016.