# Unsupervised Domain Adaptation for Visual Navigation

Shangda (Harry) Li

CMU-CS-20-110
May 2020

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Louis-Philippe Morency (Chair)
Matthew R. Gormley

*Submitted in partial fulfillment of the requirements for the degree of Master of Science.*

**Keywords:** Machine Learning, Reinforcement Learning, Unsupervised Learning, Visual Navigation, Multimodal Adaptation, Computer Vision, Vision for Robotics, Generative Adversarial Network

## Abstract

Recent advances in artificial intelligence, especially in the fields of computer vision and reinforcement learning, have made it possible to train visual navigation agents with great performance in a wide variety of navigation tasks. For example, in computer photo-realistic simulation of real-world apartments, the trained agent can reliably navigate to a specified coordinate, or a room of a specified type such as kitchen and bathroom. When asked to explore as much area as possible under a fixed time budget, the trained agent exhibit great memory of where it has been to and strategic planning. All these tasks require the agent to process raw first-person images to construct a meaningful understanding and representation of the room such as where the walls and obstacles are located, and conduct structural and semantic reasoning to determine its path, the room type, or the floor plan.

However, for most learning-based navigation agents, the training and testing are done in the same simulation environment. In order for these methods to be practical in the real world, they need to be transferable to unseen environments and non-simulated environments.

We propose an unsupervised domain adaptation method for visual navigation, which trains an image translation model that translates the images of the evaluation environment that the agent is never trained on, into images of the training environment where the agent learns to perform the task, so that the agent can recognize the translated images and achieve good performance in the evaluation environment. The image translation model is trained given an already trained agent, so that it could take advantage of the task-relevant representations learned by the agent to ensure those representations are preserved during translation.

We conduct both simulation-to-simulation and simulation-to-real-world experiments to demonstrate the effectiveness of our method in helping the trained agents adapt to unseen environments. In the simulation-to-simulation environment, the proposed method outperforms several baselines including direct transfer and popular generic image translation methods such as CycleGAN, across two different visual navigation tasks. In the simulation-to-real-world experiment, the agent enhanced by our method achieves significantly better performance than those without the enhancement.

## Acknowledgments

I would like to thank my advisor Professor Louis-Philippe Morency for giving me countless valuable guidance and absolute support for this research project.

I would like to thank Professor Matthew R. Gormley for agreeing to be on my thesis committee, and providing insightful feedback during the oral presentation.

I would like to thank my research mentor Yao-Hung Hubert Tsai for promoting, supporting, inspiring, and preparing me for all kinds of research ideas and opportunities.

I would like to thank all my collaborators on this project: Devendra Singh Chaplot, Yao-Hung Hubert Tsai and Yue Wu, for huge dedications and contributions to this research project, and encouraging and inspiring each other through all the hardships. I would also like to thank Professor Ruslan Salakhutdinov for providing advice for this research project.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation and Challenges

Recent advancements in the field of deep reinforcement learning have enabled training of intelligent agents in complex environments to perform tasks requiring understanding and reasoning of visual input, such as the traditional board game Go, RTS (Real-Time-Strategy) video games like Dota and StarCraft, and visual navigation tasks given target GPS coordinate or even language descriptions like kitchen or bathroom. Many of these agents can achieve very impressive performance. For example, Alpha Zero [37] mastered Go and defeated top human professional players. OpenAI Five [31] and AlphaStar [42] managed to reach the performance level of top professional human teams and players in Dota and StarCraft, and demonstrated deep understanding and complex reasoning of the game mechanics.

However, all these experiments are carried out in games and computer simulations, because training these agents require a large amount of data and simulation, which is impossible in the real world. According to OpenAI [31] and DeepMind [42], the training of OpenAI Five and AlphaStar would require tens of thousands of years if the training were to take place in the real world instead of in the computer simulations. Even for visual navigation tasks that are simple to humans, like walking to a specific location in an apartment, it would take the state-of-the-art reinforcement learning method DD-PPO [45] hundreds of years to achieve near-human-level performance if directly trained in the real world.

Beyond sample efficiency, there are more obstacles to make real-world training of these agents feasible. Reinforcement learning essentially trains the agent by placing it in an environment where the agent can freely explore and learn from experiences. However, we cannot allow a reinforcement learning agent to freely explore the real world. For example, if we want to train an autonomous driving car in the real world through learning-based methods, we need humans to supervise the entire training process and restrict the agents' behaviours when necessary, to prevent them from crashing or running over people. Moreover, humans have to provide proper feedback to make the agents learn from their experiences. In the autonomous driving car example, humans need to explicitly tell the agents that they are wrong whenever the agents run the red light. All these requirements of a lot of human intervention, supervision and domain knowledge are really costly to implement in the real world.

Since it is very challenging to train reinforcement learning agents in the real world, then how can we make reinforcement learning adaptable to the real world settings? How can we take advantage of the agents with potentially super-human performance trained in the simulation, and apply them to the real world?

This thesis aims to explore methods transferring reinforcement learning agents that are trained in a computer simulation to be used in the real world and other simulations. We focus on visual navigation tasks for experiments, because Facebook recently released a high-quality photo-realistic simulation environment called Habitat [33] for indoor visual exploration tasks. Habitat supports multiple simulators which is ideal for experiments on transfer between different simulations. We also have access to real-world robots equipped with similar sensors to those used in Habitat, which can be deployed in buildings to conduct simulation-to-real-world experiments.

We call the environment where reinforcement learning agents are trained (a computer simulation) as the source environment or source domain, and we call the environment where the agents are adapted to, tested and evaluated (the real world or a different simulation) as the target environment or target domain. The most significant technical challenge is that we want to make the transfer process require only limited knowledge and access to the target environment, and as little human supervision, intervention and domain knowledge as possible. Theoretically, the more information and domain knowledge we have about the target environment, the better performance we should expect from the transferred policy, but at the same time, it would require more human intervention and supervision, making the method less general, less practically useful, and applicable to less environments and tasks. It remains debatable what might be the best trade-off.

In this work, we focus on the extreme setting that requires no human intervention or supervision, and as little knowledge about the target environment as possible. Moreover, to make our approach more general, we want to separate the "policy component" and the "adaptation component". The "policy component" denotes the trained agent in the source environment (the training can be done through reinforcement learning or other learning-based methods), while the "adaptation component" denotes some model that helps the "policy component" to perform well in the target environment. Ideally, the "adaptation component" should have no or little assumption and knowledge about the "policy component", and our approach only trains an "adaptation component" without modifying the "policy component". To realize this goal, we explore the potential of unsupervised learning that learns a task-specific mapping between the source environment visual input and the target environment visual input.

More specifically, as we focus on visual navigation tasks, our ideal is to be able to train autonomous driving cars in realistic computer simulations of the read world (the popular open-world video game Grand Theft Auto is an example of a relatively realistic simulation of the real world environment), and adapt them to the real world through random real world images, which we believe to be readily available. For example, we may use the countless Google Street View photos. To take the zero-to-one step toward this idea, the problem setup in this project uses only random observations in the target environment as the "minimal" knowledge about the target environment. We use photo-realistic 3D simulations and real world robots to demonstrate the possibility of such transfer, where the tasks concerned are visual navigation and exploration, which is the foundation for effective autonomous driving agents.

## 1.2 Problem Statement

In the past few years, a lot of progress has been made in learning to navigate from first-person RGB images. Reinforcement learning have been applied to train navigation policies to navigate to goals according to coordinates [7, 17, 46], images [56], object labels [17, 52], room labels [48, 49] and language instructions [3, 5, 8, 14, 19, 44]. However, such navigation policies are predominantly trained and tested in simulation environments. Our goal is to have such navigation capabilities in the real-world. While some progress has been made towards moving from game-like simulation environments to more realistic simulation environments based on reconstructions [4, 38, 51] or 3D modeling [25], there is still a significant visual domain gap between simulation environments and real-world.

Training navigation policies (machine learning models of navigation agents) in the real-world has not been possible as current reinforcement learning methods typically require tens of millions of samples for training. Even if we parallelize the training across multiple robots, it will still require multiple weeks on training with constant human supervision due to safety concerns and battery limitations. This makes real-world training practically infeasible and leaves us with the other option of transferring models trained in simulation to the real-world, which highlights the importance of domain adaptation methods.

Among domain adaptation techniques, unsupervised methods are most favourable because it is extremely expensive to collect parallel data for the purpose of visual navigation: It essentially requires reconstructing real-world scenes in the simulator separately for all possible scenarios one might deploy the navigation model in such as different lightning conditions, time of day, indoor vs outdoor, weather conditions and so on. Undoubtedly, reconstructing real-world scenes from aligned simulation-reality scene pairs is a tedious job requiring specialized cameras and significant human effort. Unsupervised learning method has the potential to overcome this difficulty since it considers only a few real-world images taken by regular cameras.

One possible solution involves using unsupervised image translation techniques to translate visual perception from simulation to reality and adapt the navigation policy learned in simulation to the real-world. Although there already exists a rich amount of prior works in unsupervised image translation techniques that transfers images from one domain to another [21, 28, 54], prior techniques are not well suited for navigation since the image translations are agnostic of the navigation policy and instead focus on photo-realisticity and clarity.

In this thesis, we propose a unique unsupervised domain adaptation method for transferring navigation policies from simulation to the real-world, by unsupervised image translation subject to the constraint that the image translation respects agent's policy. In order to learn policy-based image translation (PBIT) in an unsupervised fashion, we devise a disentanglement of content and style in images such that the representations learnt by the navigation policy are consistent for images with the same content with different styles. Our experiments show that the proposed method outperforms the baselines in transferring navigation policies for different tasks between two simulation domains and from simulation to the real-world.

See Figure 1.1 for the illustration of PBIT. We model the observation (image) of a particular domain as generated from a content code and a style code. The content code should contain all the information required to perform the task (i.e. construct the policy), which should be the embedding space shared by different domains and makes the transfer between the two domains

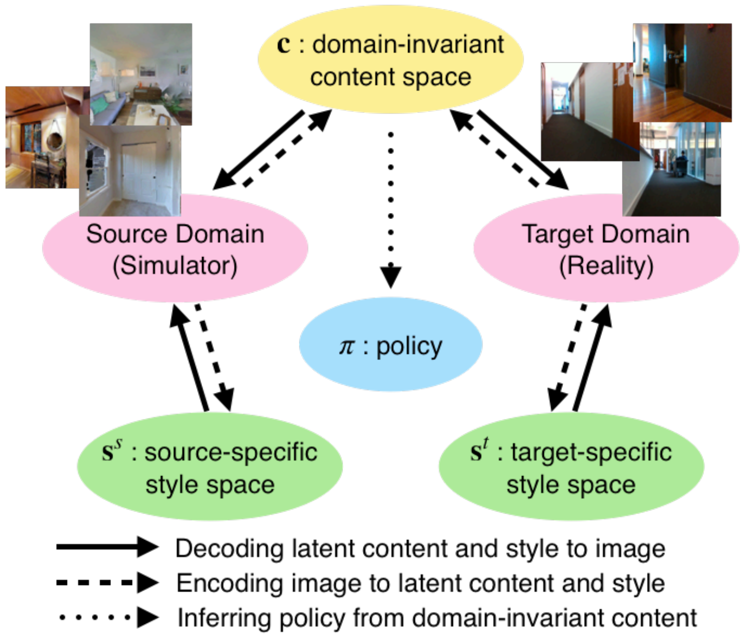

c : domain-invariant content space
Source Domain (Simulator)
Target Domain (Reality)
π : policy
s^s : source-specific style space
s^t : target-specific style space
Decoding latent content and style to image
Encoding image to latent content and style
Inferring policy from domain-invariant content

Figure 1.1: **PBIT.** The proposed policy-based image translation for unsupervised visual navigation adaptation.

possible. The style code is domain-specific, and should be irrelevant to the task. In visual navigation tasks, for example, the content code should encode information about where the walls and obstacles are located, whereas the style code should be about the particular colors, textures, shades and lighting of the walls and the obstacles.

## 1.3 Contributions

- We propose a general framework and setting for adapting reinforcement learning agents trained in one environment to other environments where knowledge and access are very limited. The framework involves the separation of "navigation component" and "adaptation component".
- Under the framework, we propose an unsupervised domain adaptation method for visual navigation. Our method translates the images in the target domain to the source domain such that the translation is consistent with the representations learned by the navigation policy.
- We conduct both simulation-to-simulation and simulation-to-real-world experiments to investigate the intrinsic adaptation capabilities of learning-based visual navigation methods, and demonstrate the effectiveness of our domain adaptation method. The proposed method outperforms several baselines (including Direct Transfer, CycleGAN, and ablation study of our method) across two different navigation tasks (PointGoal and Exploration).
- We conduct both quantitative and qualitative analysis of the experiment results. We explore different implementations of our method and present the best solutions we find.

- We build a open-source platform packaging code, data, simulators and environment, for future researches on the topic of domain adaptation for visual navigation.

## 1.4 Thesis Organization

In Chapter 2, we a survey of prior works on visual navigation and visual domain adaptation. Chapter 3 describes our proposed unsupervised domain adaptation method for translating target environment observations to source environment observations. Chapter 4 presents our experiments procedures and results. Chapter 5 reflects on our findings and discusses limitations of our methods as well as future directions.

# Chapter 2

# Related Work

Simulation to Reality (Sim2Real) visual navigation requires the adaptation from simulation to reality for both visual perception and agent's policy. Among its wide range of relevant literature, we focus on discussing related work on *visual navigation* and *visual domain adaptation*.

**Visual Navigation.** Prior work on learning-based visual navigation can broadly be categorized into two classes based on whether the location of the goal is known or unknown. Navigation scenarios where the location of the goal is known includes the most common *pointgoal* task where the coordinate to the goal is given [17, 32]. Another example of a task in this category is vision and language navigation [3] where the path to the goal is described in natural language. Navigation scenarios where the location of the goal is not known include a wide variety of tasks. These include navigating to a fixed set of objects [12, 17, 26, 29, 50], navigating to an object specified by language [5, 19] or by an image [56], and navigating to a set of objects in order to answer a question [11, 16]. Tasks in this category essentially involve efficiently and exhaustively exploring the environment to search the desired object. Some recent works explicitly tackle the problem of exploration by training end-to-end RL policies maximizing the explored area [7, 10, 13]. In this project, we tackle one task in each category, PointGoal and Exploration.

Most of the above works train navigation policies using reinforcement or imitation learning and test in simulation and test on different scenes in the same domain in the simulator. Some prior works which tackle sim2real transfer for navigation policies directly transfer the policy trained in simulation to the real-world without any domain adaptation technique [7, 17]. We show that the proposed domain adaptation method can lead to large improvements over direct policy transfer.

**Visual Domain Adaptation.** Simulation and reality can be viewed as two distinct visual domains, and adapting their visual perceptions can be regarded as an image-to-image translation task. Thanks to the success of Generative Adversarial Networks (GANs) [15] for matching cross-domain distribution, we are able to adapt an image across domains without changing its context. For example, pix2pix [22] changes only the style of an image (e.g., photograph $\rightarrow$ portrait) while preserving its context (e.g., the same face of a person). We note that, for Sim2Real navigation, some amount of context should be preserved across domains, such as the barriers and the walls, to prevent collisions of our agent.

If we have access to the paired cross-domain images, then pix2pix [22] and BicycleGAN [55] serve as good candidates to model the context-preserving adaptation. However, the paired data between simulation and reality is notoriously hard to collect [40] or even do not exist (e.g., we cannot always build simulators for new environments). To tackle this challenge, numerous visual domain adaptation approaches [23, 28, 36, 39, 53, 54] have been proposed to relax the constraint of requiring paired data during training time. Nevertheless, the above methods still assume one-to-one correspondence across domains (i.e. there exists a deterministic mapping between images of the two domains). As an example, these models can only generate the same target-domain image given a source-domain image. We argue that it is more realistic to assume many-to-many mappings between simulation and reality.

To achieve both multimodal mappings and training without paired data, MUNIT [21] and DRIT [27] propose to disentangle the context and style of an image. Precisely, they assume the context is shared across domains and the styles are specific to each domain. Note that these models focus on realistic image generation, and hence it remains unclear on how image translation benefits cross-domain visual navigation. To further bridge the gap between navigation and image translation, our key idea is to ensure the agent's navigation policy be consistent under domain translation. As a consequence, we propose to enforce constraints such that the agent's policy is only inferred from the shared context across simulation and reality.

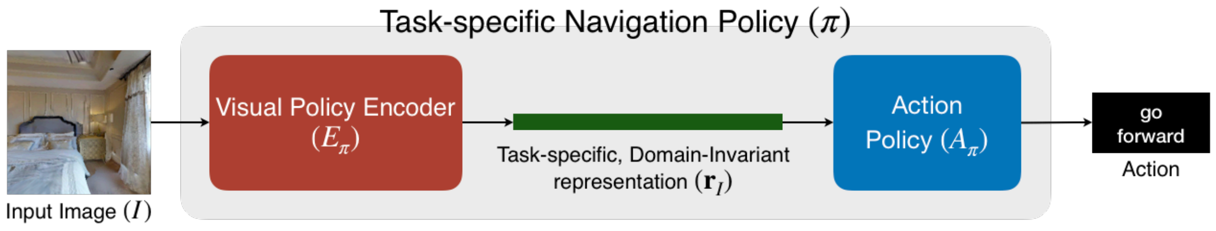
# Chapter 3

# An Unsupervised Domain Adaptation Method for Visual Navigation

Denote the source domain as $(\mathcal{S}^s, \mathcal{A}, P^s)$ and the target domain as $(\mathcal{S}^t, \mathcal{A}, P^t)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition probability distribution. Note that we assume action spaces $\mathcal{A}$ are shared across domains. Let $\pi^s : \mathcal{S}^s \rightarrow \mathcal{A}$ be a navigation policy in the source domain $s$ (the navigation policy is given). For our task setup, we have access to some target-domain images $I^t \in \mathcal{S}^t$ during training, but we cannot perform target-domain policy $(\pi^t)$ training. Our objective is to learn a many-to-many mapping $F : \mathcal{S}^t \rightarrow \mathcal{S}^s$ such that the navigation policy under the source-to-target mapping, $\pi^t(I^t) = \pi^s(F(I^t))$, is effective in the target domain $t$. Under Sim2Real setting, the source domain refers to simulator and the target domain refers to reality. Unless specified, we abbreviate $\pi^s$ as $\pi$ for the rest of the paper. The remaining part of this section shall describe our proposed 'Policy-Based Image Translation' (PBIT).

## 3.1 Policy Decomposition

As our objective is to transfer the task-specific navigation policy across domains, we assume that the task itself is domain-invariant. As a consequence, given a policy $\pi$ (for the navigation task in the source domain), we assume that some intermediate task-specific representation inferred by the policy is invariant from the source to the target domain. For example, a simple obstacle avoidance navigation policy would extract task-specific and domain-invariant features



Task-specific Navigation Policy ($\pi$)
Visual Policy Encoder ($E_\pi$)
Task-specific, Domain-Invariant representation ($\mathbf{r}_I$)
Action Policy ($A_\pi$)
go forward
Action
Input Image ($I$)

Figure 3.1: **Policy Decomposition.** The Task-specific Navigation Policy ($\pi$) can be sequentially decomposed into a Visual Policy Encoder ($E_\pi$) and a Action Policy ($A_\pi$) such that $E_\pi$ extracts all task-specific features ($\mathbf{r}_I$) and throws away all domain-specific features from the input image ($I$) and $A_\pi$ learns an action distribution function over the task-specific features.

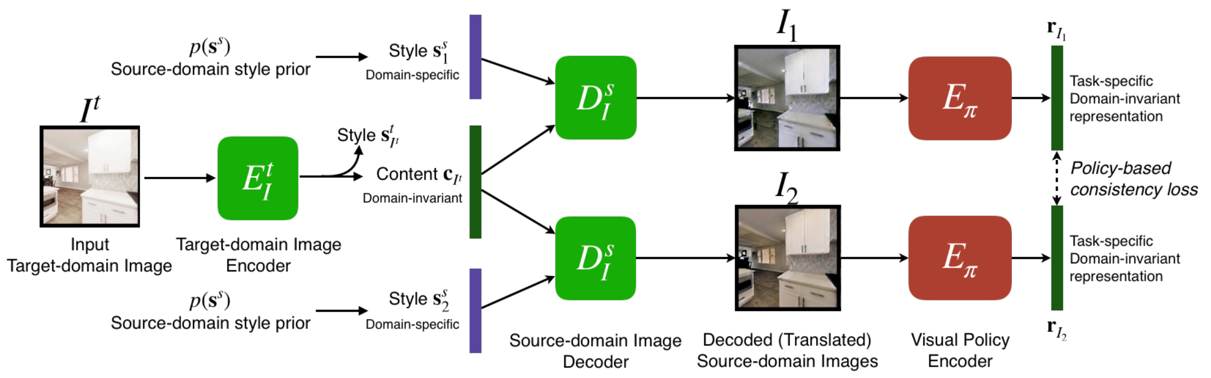


Figure 3.2: **Policy-based Consistency Loss.** Since the task is domain-invariant, task-specific representations obtained from different domain-specific styles but the same domain-invariant content should be similar.

such as distance to obstacles at various angles and then learn a policy over these features. Let $\pi$ be sequentially decomposed into a Visual Policy Encoder ($E_\pi$) and a Action Policy ($A_\pi$). $E_\pi$ extracts all task-specific features ($\mathbf{r}_I$ with $I$ indicating the input image) and throws away all domain-specific features in the input image. $A_\pi$ learns an action distribution function over the task-specific features. We illustrate the policy decomposition in Figure 3.1.

## 3.2 Policy-based Consistency Loss

Recall that our objective is to learn an image translation model $F : \mathcal{S}^t \rightarrow \mathcal{S}^s$, such that $\pi^t(I^t) = \pi^s(F(I^t))$. By policy decomposition, given a target-domain image, different translated images to the source domain would have similar task-specific features. Precisely, if $I_1^s$ and $I_2^s$ are the translated images to the source domain from the same target-domain image $I^t$. In other words, if $I_1^s, I_2^s \sim F(I^t)$, then $E_\pi(I_1^s) \approx E_\pi(I_2^s)$.

To achieve the above policy consistency in an unsupervised fashion, we take inspiration from style and content-based unsupervised methods designed for image translation [21]. We assume that each image can be decomposed into a domain-invariant content representation ($\mathbf{c}$) and a domain-specific style representation ($\mathbf{s}$). Let $E_I^s$ be an Image Encoder for domain $s$ which encodes an image ($I$) to domain-invariant content ($\mathbf{c}$) and domain-specific style ($\mathbf{s}^s$): $E_I^s(I) = (\mathbf{c}_I, \mathbf{s}_I^s)$. On the contrary, let $D_I^s$ be an Image Decoder which is the inverse of the Image Encoder: $D_I^s(\mathbf{c}_I, \mathbf{s}_I^s) = I$.

Since we assume the navigation task is domain-invariant, all the the task-specific features are a subset of content representation $\mathbf{r}_I \in \mathbf{c}_I$. Therefore, images generated from different styles but same content should lead to the same task-specific features as shown in Figure 3.2. We operationalize this idea using the following policy-based consistency loss:

$$\mathcal{L}_{pol} = \mathbb{E}_{\mathbf{c}_{I^t}:(\mathbf{c}_{I^t},\text{-})\in E_I^t(I^t), I^t\sim\mathcal{S}^t, \mathbf{s}_1^s\sim p(\mathbf{s}^s), \mathbf{s}_2^s\sim p(\mathbf{s}^s)}[||E_\pi(D_I^s(\mathbf{c}_{I^t}, \mathbf{s}_1^s)) - E_\pi(D_I^s(\mathbf{c}_{I^t}, \mathbf{s}_2^s))||_1] \tag{3.1}$$

with $\mathbf{s}_1^s$ and $\mathbf{s}_2^s$ being two distinct styles sampled from the the prior distribution $p(\mathbf{s}^s) := \mathcal{N}(\mathbf{0}, \mathbf{I})$ being a multivariate Gaussian with zero mean and diagonal unit covariance.

Note that in the above equation, $E_\pi(\cdot)$ is part of the given navigation policy. We assume the navigation policy is only trained before deciding the target domain; hence, $E_\pi(\cdot)$ is fixed during the domain adaptation phase. This adoption ensures that PBIT can be used for transferring a policy across domains (potentially not anticipated during policy training) without re-training the the navigation policy.

## 3.3 Reconstruction and Adversarial Loss

Using just policy-based consistency loss would make decoder $D_I$ ignore the style and decode based only on the content. Inspired by prior work [21, 54], to encourage the content to be domain-invariant and style representations to be domain-specific, we adopt the following image and latent representation reconstruction losses, and use $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for the prior distributions of styles $p(\mathbf{s}^s)$ and $p(\mathbf{s}^t)$:

$$\begin{aligned}\mathcal{L}_{im\_rec} =& \mathbb{E}_{I^t \sim \mathcal{S}^t}[||D_I^t(E_I^t(I^t)) - I^t||_1] + \mathbb{E}_{I^s \sim \mathcal{S}^s}[||D_I^s(E_I^s(I^s)) - I^s||_1], \\ \mathcal{L}_{lat\_rec} =& \mathbb{E}_{\mathbf{c}_{I^t}:(\mathbf{c}_{I^t},\_) \in E_I^t(I^t), I^t \sim \mathcal{S}^t, \mathbf{s}^s \sim p(\mathbf{s}^s)}[||E_I^s(D_I^s(\mathbf{c}_{I^t}, \mathbf{s}^s)) - (\mathbf{c}_{I^t}, \mathbf{s}^s)||_1] \\ &+ \mathbb{E}_{\mathbf{c}_{I^s}:(\mathbf{c}_{I^s},\_) \in E_I^s(I^s), I^s \sim \mathcal{S}^s, \mathbf{s}^t \sim p(\mathbf{s}^t)}[||E_I^t(D_I^t(\mathbf{c}_{I^s}, \mathbf{s}^t)) - (\mathbf{c}_{I^s}, \mathbf{s}^t)||_1].\end{aligned} \tag{3.2}$$

We also use adversarial losses to match the distribution of images to their respective domains. Let $\mathrm{Dis}^t$ be the discriminator for the target domain $t$ and $\mathrm{Dis}^s$ be the discriminator for the source domain $s$:

$$\begin{aligned}\mathcal{L}_{adv} =& \mathbb{E}_{\mathbf{c}_{I^t}:(\mathbf{c}_{I^t},\_) \in E_I^t(I^t), I^t \sim \mathcal{S}^t, \mathbf{s}^s \sim p(\mathbf{s}^s)}[\log \mathrm{Dis}^s(D_I^s(\mathbf{c}_{I^t}, \mathbf{s}^s))] \\ &+ \mathbb{E}_{\mathbf{c}_{I^s}:(\mathbf{c}_{I^s},\_) \in E_I^s(I^s), I^s \sim \mathcal{S}^s, \mathbf{s}^t \sim p(\mathbf{s}^t)}[\log \mathrm{Dis}^t(D_I^t(\mathbf{c}_{I^s}, \mathbf{s}^t))] \\ &+ \mathbb{E}_{I^s \sim \mathcal{S}^s}[\log(1 - \mathrm{Dis}^s(I^s))] + \mathbb{E}_{I^t \sim \mathcal{S}^t}[\log(1 - \mathrm{Dis}^t(I^t))].\end{aligned} \tag{3.3}$$

Putting everything together, our overall objective is

$$\mathcal{L}_{full} := \lambda_{pol}\mathcal{L}_{pol} + \lambda_{im\_rec}\mathcal{L}_{im\_rec} + \lambda_{lat\_rec}\mathcal{L}_{lat\_rec} + \lambda_{adv}\mathcal{L}_{adv}, \tag{3.4}$$

where $\lambda$.s are hyper-parameters controlling the weight of each loss during training. The cross-domain image translation model consists of $D_I^t, E_I^t, D_I^s, E_I^s$, and the optimization admits a mix-max objective:

$$D_I^t, E_I^t, D_I^s, E_I^s = \arg \min_{D_I^t, E_I^t, D_I^s, E_I^s} \max_{\mathrm{Dis}^t, \mathrm{Dis}^s} \mathcal{L}_{full}. \tag{3.5}$$

# Chapter 4

# Experiments

## 4.1 Experimental Setup

We conduct experiments in both the simulator and the real world. For the simulator setting, we use the Habitat simulator [33] with the Gibson [51] and Replica [38] datasets, and consider two visual navigation tasks for domain adaptation: PointGoal and Exploration. We firstly train an RL policy on Gibson for each task, and create a dataset with unlabeled and unpaired images from Gibson and Replica scenes to train PBIT and a CycleGAN baseline. We then benchmark PBIT against direct transfer policy, CycleGAN transfer, and PBIT without policy consistency constraint on both tasks in the Replica dataset.

For the real world, we use LoCoBot [1] as our real-world agent, train PBIT model with 1125 random real-world images from 3 different indoor scenes and 7200 random Gibson images. We then benchmark PBIT against the direct transfer policy baseline.

### 4.1.1 Navigation Tasks Definitions

**PointGoal Task.**

The PointGoal Task is natively implemented in the Habitat simulator [33]. An agent is positioned at a random starting location and orientation in each episode, and is supposed to navigate to a target location. The agent has two sensors: RGB camera and GPS+Compass. The observation space consists of RGB images of shape $3 \times 256 \times 256$ and GPS+Compass input of shape $2 \times 1$. The action space consists of four actions: STOP (indicating the agent has reached the target location), MOVE-FORWARD ($0.25m$), TURN-LEFT ($10°$), TURN-RIGHT ($10°$). The episode ends immediately after the agent takes the STOP action. Otherwise, the episode automatically ends after 500 steps. As [2] suggested, we used two evaluation metrics: Success and Success weighted by Path Length (SPL). The episode is considered successful if the agent is within $0.2m$ of the target location when the episode ends. SPL measures also measures the efficiency of the policy in addition to the success, i.e. shorter trajectories lead to higher SPL: SPL $= \frac{l}{\max(l,p)}S$ where $l$ is the length of the shortest path possible between the starting location and the target location, $p$ is the length of the agent's path and $S$ denotes success. The reward for training RL policies on this task is the decrease in geodesic distance to the point goal.

**Exploration Task.**

We follow the Exploartion task setup used in [9] and [6], where an agent is positioned at a random starting location and orientation in each episode, and is supposed to maximize coverage given a fixed time budget of 500 steps. Coverage is defined to be the total area of explored traversable points from the agent's starting location. A traversable point is explored by the agent if it is in the field-of-view of the agent and less than $3m$ away from the agent. The agent is equipped with two sensors: RGB camera and base odometry sensor. The spec of the RGB camera is the same as in PointGoal task. The base odometry sensor provides the agent with readings that denote the change in the agent's x-y coordinates and orientation. Thus the observation space consists of RGB images of shape $3 \times 256 \times 256$ and base odometry sensor input of shape $3 \times 1$. The action space consists of three actions: MOVE-FORWARD ($0.25m$), TURN-LEFT ($10°$), TURN-RIGHT ($10°$). Each episode ends after 500 steps. As [6] suggested, we use two evaluation metrics, the absolute coverage area in $m^2$ (Explored Area) and proportion of area explored in the scene (Explored Ratio). Explored Ratio is defined as ratio of coverage to maximum possible coverage in the corresponding scene. During training, the reward received by the agent at each step is equal to the amount of new area explored by that step.

### 4.1.2 RL Training Details

**Agent Architecture.**

Our agent architecture consists of two neural networks: a visual encoder $E_\pi$ and a policy encoder $A_\pi$. The visual encoder $E_\pi$ is based on the 18-layer ResNet [18], as illustrated in Figure 4.1. $E_\pi$ outputs 128-dimensional policy-related representations $\mathbf{r}_I$ given RGB images of shape $3 \times 256 \times 256$. The policy encoder $A_\pi$ is based on a 2-layer GRU, which takes $\mathbf{r}_I$ together with readings from either the GPS+Compass sensor in PointGoal task or the base odometry sensor in Exploration task.

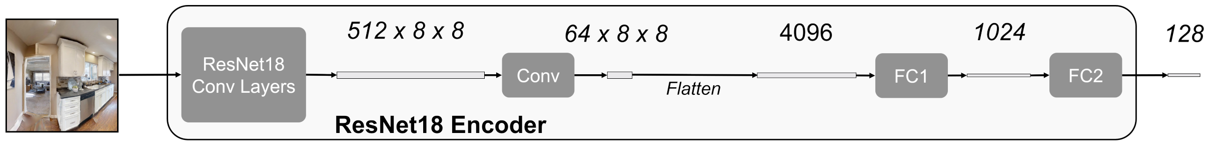


Figure 4.1: An illustration of the network architecture of the Visual Policy Encoder.

**Training.**

We train three RL agents in Gibson using PPO [35] with Generalized Advantage Estimation [34]. The first agent is for PointGoal task with $1.25m$ camera height, which is tested to transfer to Replica. The second is for PointGoal task with $60cm$ camera height, which is tested to transfer to Real World. The last agent is for Exploration task, which is tested to transfer to Replica. All three agents are trained on the train split of the *pointnav_gibson_v1* dataset provided by [33]. The dataset contains episode definitions for all 72 scenes in Gibson. The two PointGoal agents are trained with 8 concurrent workers for around 30 million frames, and achieve SPL=$0.80$ on the val split of the dataset. The Exploration agent is trained with 24 concurrent workers for around 3 million frames. At each update, each work collects 128 steps of experience and perform 2 PPO epocks with minibatch size $128 \times 2$ and clipping parameter 0.2. We use discount factor 0.99, GAE parameter 0.95, and the Adam optimizer [24] with learning rate $2.5 \times 10^{-4}$.

### 4.1.3 Baselines

We transfer the trained RL policies to the target domain using the proposed model, Policy-Based Image Translation (PBIT) against 3 baselines:

1. **Direct Transfer:** This is the most common method of transferring navigation policies across domains, which involves directly testing the policy in the target domain without any fine-tuning.
2. **CycleGAN:** CycleGAN is a competitive and popular unsupervised image translation method. This method is designed for static image translation and is agnostic to the navigation policy.
3. **Policy-Based Image Translation w.o. Policy Loss:** This is an ablation of the proposed method without the policy-based consistency loss.

### 4.1.4 PBIT & Baselines Training/Testing Details

**Domain Translation Dataset.**

For Gibson to Replica experiment, we build a dataset of unlabeled and unpaired images from Gibson and Replica scenes. Gibson environment contains 72 scenes and Replica environment contains 18 scenes. For each scene in Gibson, using the function provided by the Habitat simulator [33], we sample 100 random *navigable location* $\times$ *orientation* pairs and save the first-person view RGB images obtained from those locations and orientations. Similarly, we sample 400 for each scene in Replica. Thus, our Gibson-Replica dataset contains 7200 images from Gibson and 7200 images from Replica. For Gibson to Real World experiment, we use the robot to create a similar unpaired image dataset of 7200 images from Gibson and 1125 images from Real World. Details about how the real-world images are collected is described in Section 4.1.5.

**Model Architecture.**

For PBIT, we follow the setup suggested by [21]. We use several convolutional layers and residual blocks to construct the image encoders $E_I^s, E_I^t$ and image decoders $D_I^s, D_I^t$. We use Instance Normalization [41] in $E_I^s, E_I^t$ and Adaptive Instance Normalization [20] in $D_I^s, D_I^t$. For the discriminators $\text{Dis}^s, \text{Dis}^{st}$, we adopt the multi-scale discriminator architecture proposed by [43]. Detailed descriptions of the architecture are given in Figure 4.2. For the CycleGAN baseline, we use the architecture proposed in the CycleGAN paper [54].

Style Encoder

| |
|---|
| $(256 \times 256)$ RGB |
| Conv(32,(7x7),1) |
| Conv(64,(4x4),2) |
| Conv(128,(4x4),2) |
| Conv(128,(4x4),2) |
| Conv(128,(4x4),2) |
| global_avg_pool |
| dense(8) |
| $\rightarrow s \in \mathbb{R}^8$ |

Content Encoder

| |
|---|
| $(256 \times 256)$ RGB |
| Conv(32,(7x7),1) |
| Conv(64,(4x4),2) |
| Conv(128,(4x4),2) |
| Residual(128) |
| Residual(128) |
| Residual(128) |
| Residual(128) |
| $\rightarrow c \in \mathbb{R}^{128\times64\times64}$ |

Decoder

| |
|---|
| $c$ and AdaIN($s$) |
| Residual(128) |
| Residual(128) |
| Residual(128) |
| Residual(128) |
| UpSampling(2x2) |
| Conv(64,(5x5),1) |
| UpSampling(2x2) |
| Conv(32,(5x5),1) |
| UpSampling(2x2) |
| Conv(3,(7x7),1) |
| $(256 \times 256)$ RGB |

Discriminator

| |
|---|
| Multi-scale Input |
| Conv(64,(4x4),2) |
| Conv(128,(4x4),2) |
| Conv(256,(4x4),2) |
| Conv(512,(4x4),2) |

Figure 4.2: An illustration of the network architecture of the Visual Policy Encoder.

## Hyperparameters for PBIT.

PBIT is trained using the full objective defined in Equation (3.4) and Equation (3.5). For each task (PointGoal/Exploration) and scenario (Replica/Real-World), we use the same set of hyperparameters $\lambda_{im\_rec} = 10, \lambda_{lat\_rec} = 1, \lambda_{adv} = 1$. $\lambda_{pol}$ is policy-dependent, because the scale of the loss $\mathcal{L}_{pol}$ defined in Equation (3.1) depends on the scale of task-specific features $\mathbf{r}_I$ outputted by $E_\pi$. In our case $\mathbf{r}_I$ is a 128-dimensional vector after ReLU activation layer. For different tasks and scenarios, we use the same formula to calculate the proper $\lambda_{pol} = \frac{1.0}{\mathbb{E}_{I^s \sim \mathcal{S}^s}[||E_\pi(I^s)||_1]}$, where $\mathbb{E}_{I^s \sim \mathcal{S}^s}[||E_\pi(I^s)||_1]$ is estimated using the Gibson images from the domain translation dataset.

## Training.

For PBIT, PBIT without Policy Loss, and CycleGAN, we use the Adam optimizer [24] with $0.0001$ initial learning rate, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is halved every 100k iterations. We train all the models for 500k iterations with batch size 1, on the domain translation dataset for each task (PointGoal/Exploration) and scenario (Replica/Real-World).

## Testing.

When the RL agent trained in Gibson is tested in Replica or Real-World, the Direct Transfer baseline directly performs on the raw input images from Replica or Real-World. For our proposed model, at each step, PBIT translates the input Replica/Real-World image to Gibson image, and then the Gibson RL agent takes in the translated image as input and output the action for the task. The testing procedure for the CycleGAN baseline is the same as PBIT.

### 4.1.5 Real World Experiment

**Robot configuration.**

We present results of unsupervised zero-shot simulation to real-world indoor navigation on a LoCoBot [1]. The robot has an RGB camera 60cm from the ground, and is programmed to take action space: [stop, forward $0.25m$, left $10°$, right $10°$].

**Data collection and training.**

By taking pictures at random locations with the robot, we collect a total of 1125 images from three indoor locations: 1) 397 images from a meeting room with chairs and tables. 2) 728 images from the corridor of a building. 3) 151 images from a large study place chairs and tables.

We train an agent of camera height $60cm$ in Gibson with the same architecture as in section 4.1.2, and a policy based transfer model from 7500 random Gibson images and the 1125 real-world images.

## 4.2 Result

We transfer the trained RL policies to the target domain using the proposed model, Policy-Based Image Translation (PBIT) against 3 baselines defined in Section 4.1.3. We first present domain adaptation results from Gibson to Replica domains within the Habitat simulator and then present results of sim-to-real transfer from Gibson to real-world office scenes.

Table 4.1: **PointGoal Results.** The performance of the proposed method Policy-Based Image Translation (PBIT) as compared to the baselines on the PointGoal task when transferred from Gibson domain to the Replica domain.

| | SPL | Success Rate | Collisions |
|---|---|---|---|
| Direct Transfer | 0.505 | 0.688 | **38.7** |
| CycleGAN | 0.605 | 0.803 | 50.6 |
| PBIT w.o. Policy Loss | 0.669 | 0.852 | 40.6 |
| PBIT | **0.712** | **0.881** | 39.5 |

Table 4.2: **Exploration Results.** The performance of the proposed method Policy-Based Image Translation (PBIT) as compared to the baselines on the Exploration task when transferred from Gibson domain to the Replica domain.

| | Explored Ratio | Explored Area ($m^2$) | Collisions |
|---|---|---|---|
| Direct Transfer | 0.832 | 22.9 | **59.4** |
| CycleGAN | 0.885 | 24.7 | 84.2 |
| PBIT w.o. Policy Loss | 0.879 | 24.6 | 70.1 |
| PBIT | **0.897** | **25.3** | 73.6 |

### 4.2.1 Gibson to Replica

We use the script provided by Habitat [33] to generate 50 test episodes for each of the 18 scenes in Replica (900 episodes in total). The script makes sure that the each test episode is reasonably difficult (an episode is trivial if there is a obstacle-free straight line between the starting and target locations). We use the same 900 episodes across 18 scenes to evaluate the performance of each model in Replica.

For the **PointGoal** navigation task, we compare all the methods across Success Weighted by Path Length (SPL), Success Rate and number of collisions, which are the standard metrics for evaluation of navigation models used in most prior works. For the **Exploration** task, we compare all the methods using Explored Area in $m^2$, ratio of the environment explored and number of collisions. The performances of our method and all the baselines for the PointGoal task are presented in Table 4.1 and for the Exploration task are presented in Table 4.2.

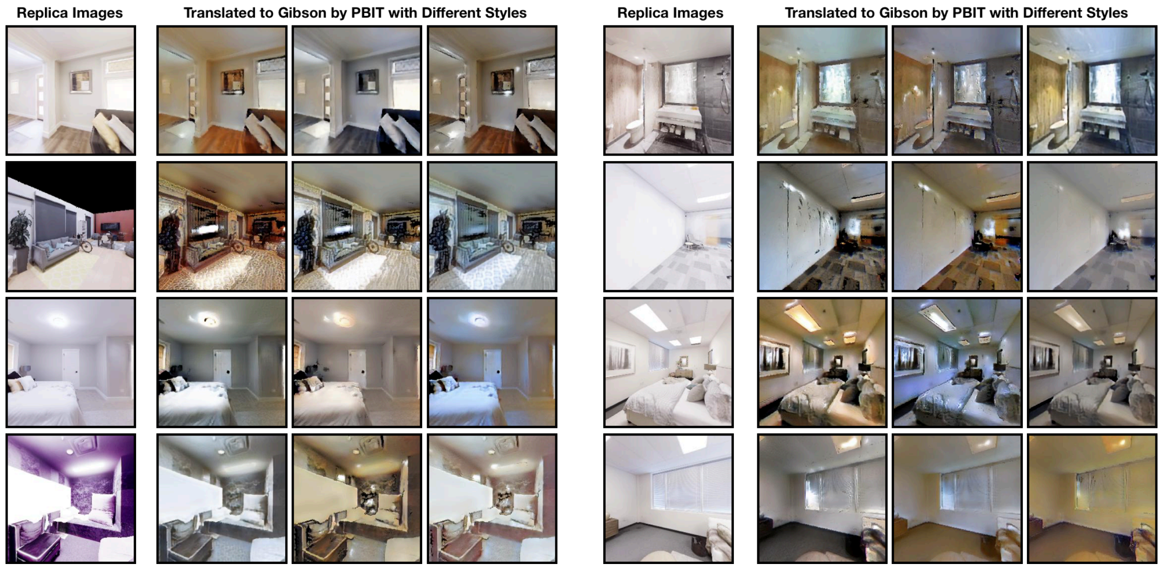

Replica Images
Translated to Gibson by PBIT with Different Styles
Replica Images
Translated to Gibson by PBIT with Different Styles

Figure 4.3: **Replica Images Translated to Gibson.** The first and fifth columns are the input images from Replica during test time. The other columns are images translated to Gibson domain by PBIT. The Gibson RL agent relies on the translated images (which preserve the policy-relevant features) to perform well in Replica.

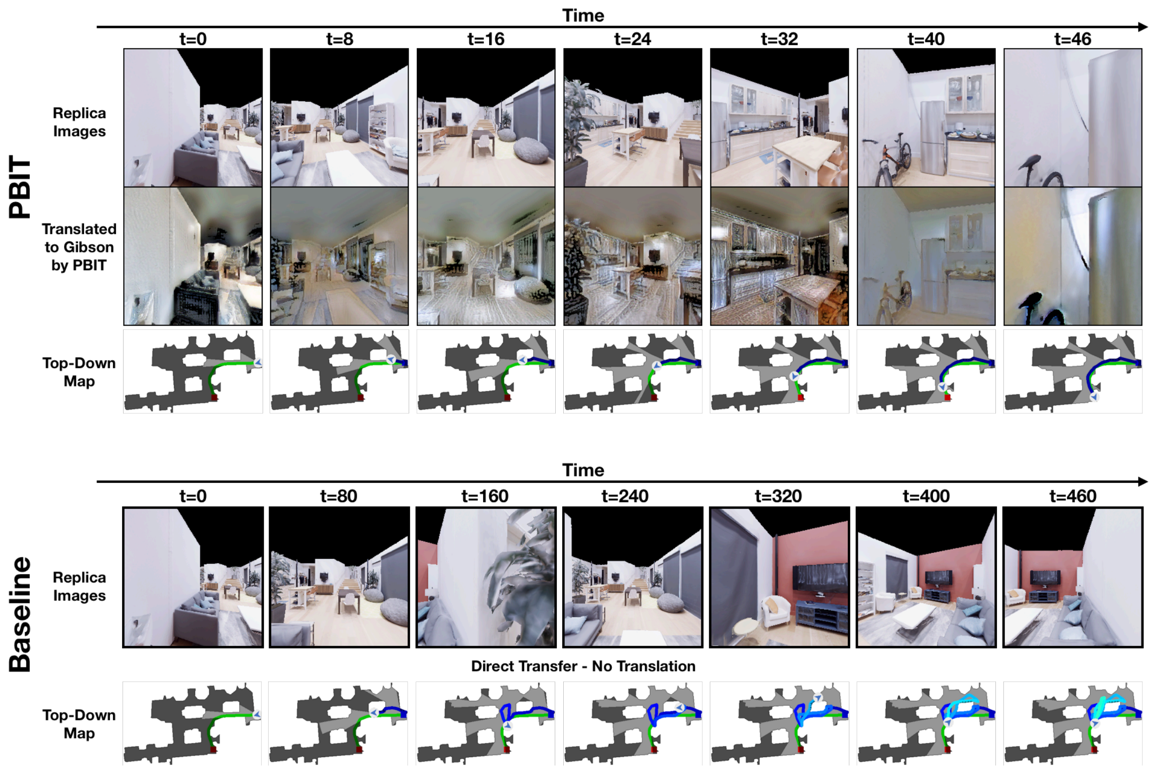
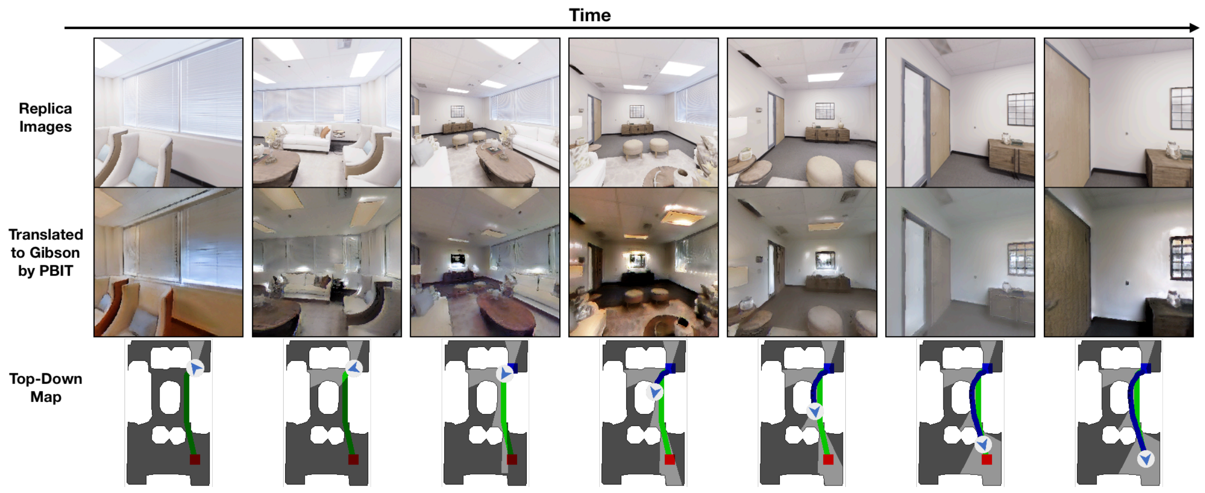


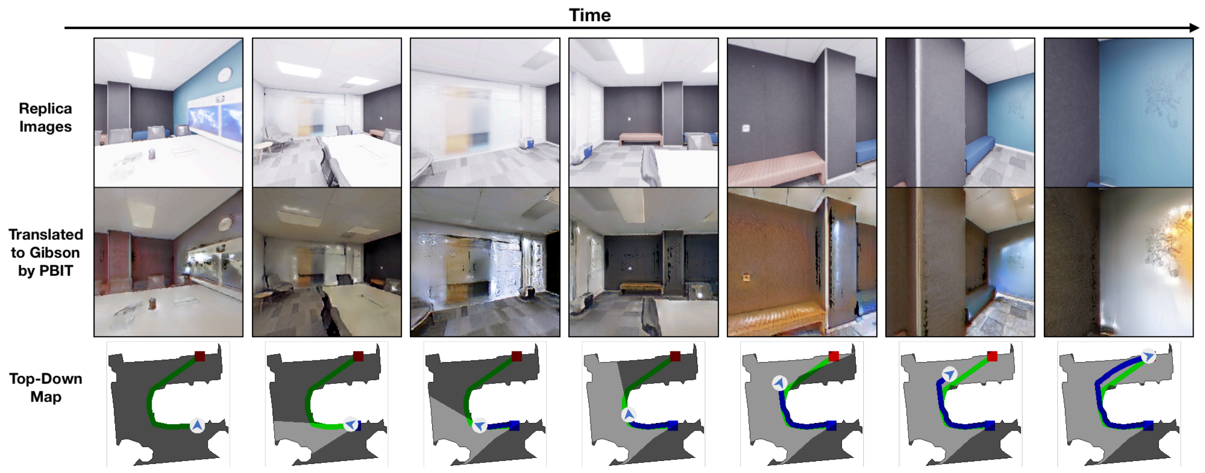
Figure 4.4: **Trajectory Comparison between PBIT and Baseline (Direct Transfer) on PointGoal Task in Replica.** The upper half of the figure is the trajectory of PBIT: the agent successfully navigates from a corner of the apartment to the fridge in 46 steps, by seeing the translated images by PBIT. The agent takes almost the shortest path possible, as shown in the Top-Down Map (not visible to the agent). The lower half of the figure is the trajectory of the Direct Transfer baseline on the same test episode. The Direct Transfer agent fails to navigate to the target location and gets lost, even after 460 steps.

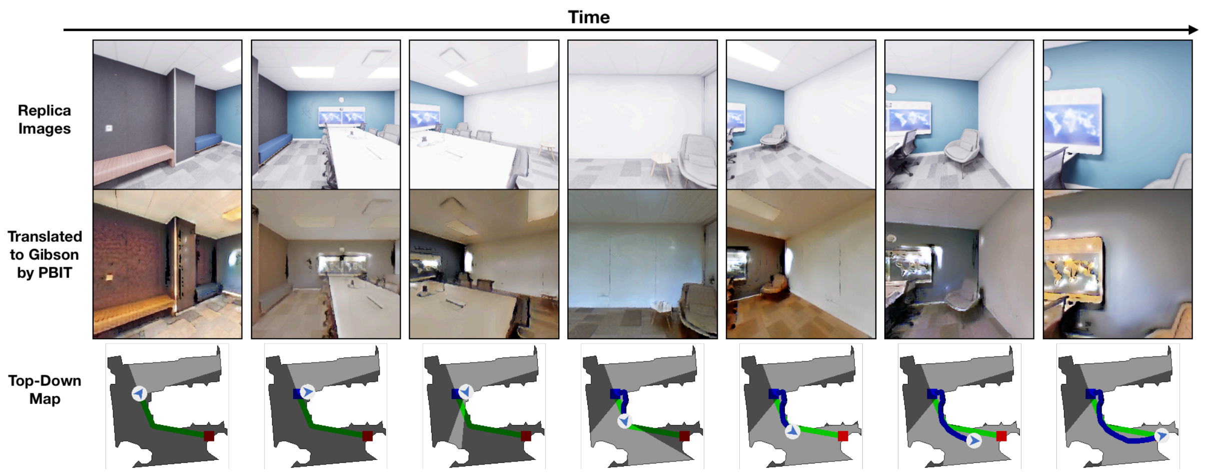
PBIT outperforms all the baselines on both the tasks. It improves the SPL from 0.605 to 0.712 for PointGoal and Explored Ratio from 0.885 to 0.897 for Exploration as compared to the CycleGAN baseline. Note that the number of collisions for the Direct Transfer baseline are low because in many episodes this baseline does not move and just turns around on the spot. This highlights the visual domain gap between the two domains. Lower performance of 'PBIT w.o. Policy Loss' ablation highlights the importance of policy-based consistency loss. The exploration ratios of all the methods in Table 4.2 are high on a absolute level because the Replica scenes are small usually having one or two rooms. Just turning on the spot leads to an exploration ratio of 0.75.

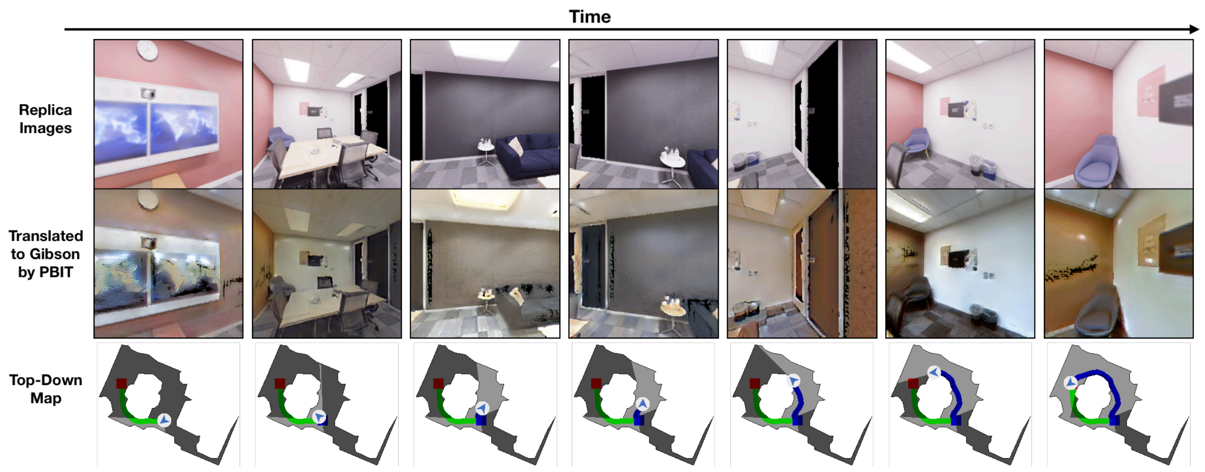
In Figure 4.3, we visualize some examples of images in the target Replica domain translated to the source Gibson domain using PBIT using different styles. The examples indicate that policy-relevant characteristics of the image such as corners of obstacles, walls and free space are preserved during the translation. In Figure 4.4, we visualize an example trajectory for the PointGoal task using the proposed method  PBIT (Fig 4.4 above) and the Direct Transfer baseline (Fig 4.4 below). The figure shows the images observed by the agent in the target domain, the translated images and a top-down map (not visible to the agent) showing the point goal coordinates and the agent's path.

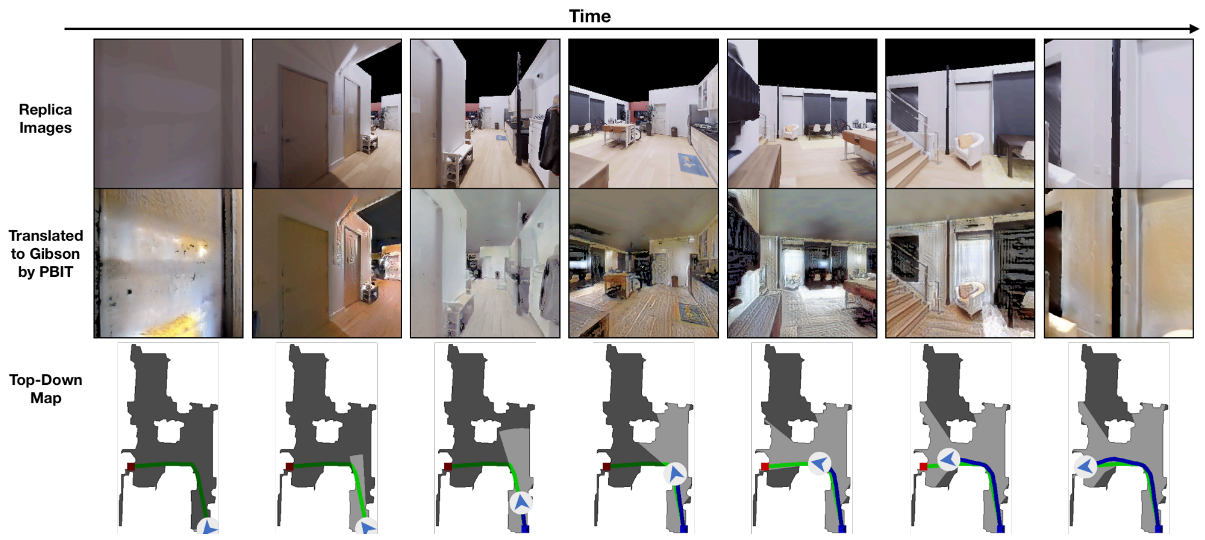
## Additional Gibson to Replica Trajectory Visualizations

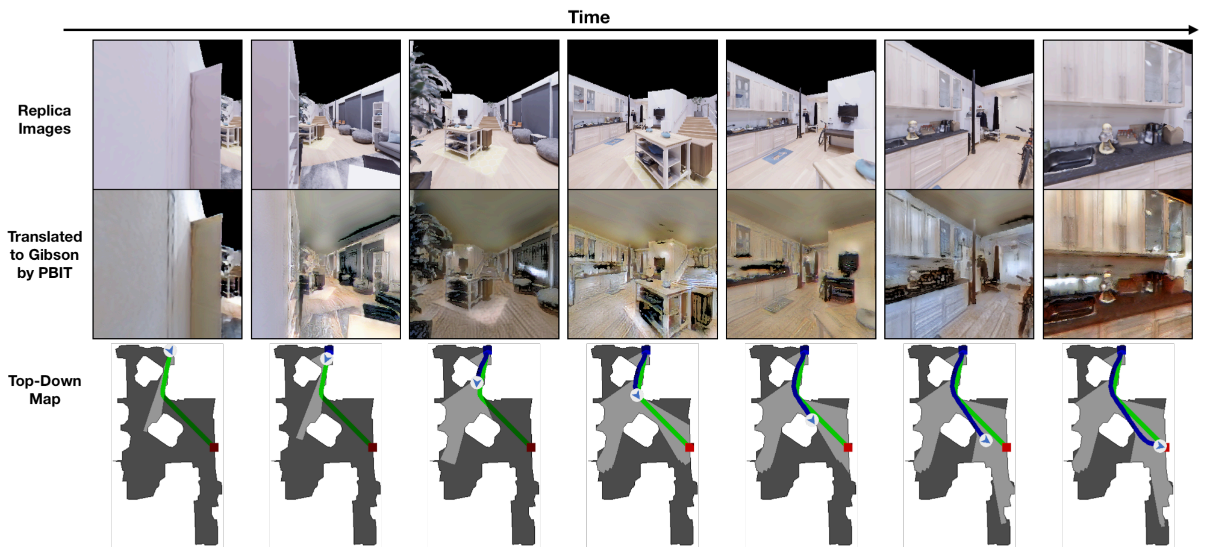


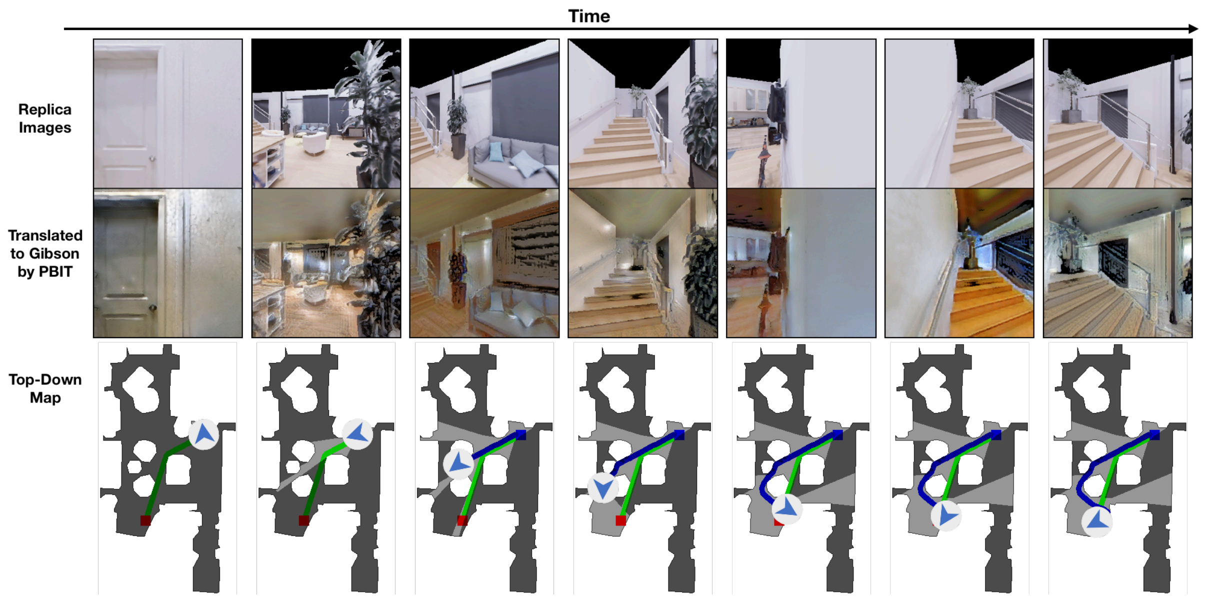



Time
Replica Images
Translated to Gibson by PBIT
Top-Down Map


Time
Replica Images
Translated to Gibson by PBIT
Top-Down Map


Time
Replica Images
Translated to Gibson by PBIT
Top-Down Map


Time
Replica Images
Translated to Gibson by PBIT
Top-Down Map


Time
Replica Images
Translated to Gibson by PBIT
Top-Down Map

### 4.2.2 Gibson to Real-world

We now transfer the PointGoal navigation policy to the real-world using the proposed method PBIT and compare it to the Direct Transfer baseline. We transfer the navigation policy to a LoCoBot [1] using the PyRobot API [30] for both the methods. We conduct 20 trials across 3 different scenes in the real-world. For each trial, we set the target point for the robot through PointGoal, and update the PointGoal each step according to the relative position calculated through the internal odometry sensors, and stop the trial either when the robot have reached the goal (distance less than $20cm$) or after 99 steps. Each trial specification and the corresponding results are presented in Table 4.3. PBIT achieves an overall $55\%$ improvement in success rate over the Direct Transfer baseline across all the trials. PBIT also has a much lower collision rate as compared to the baseline.

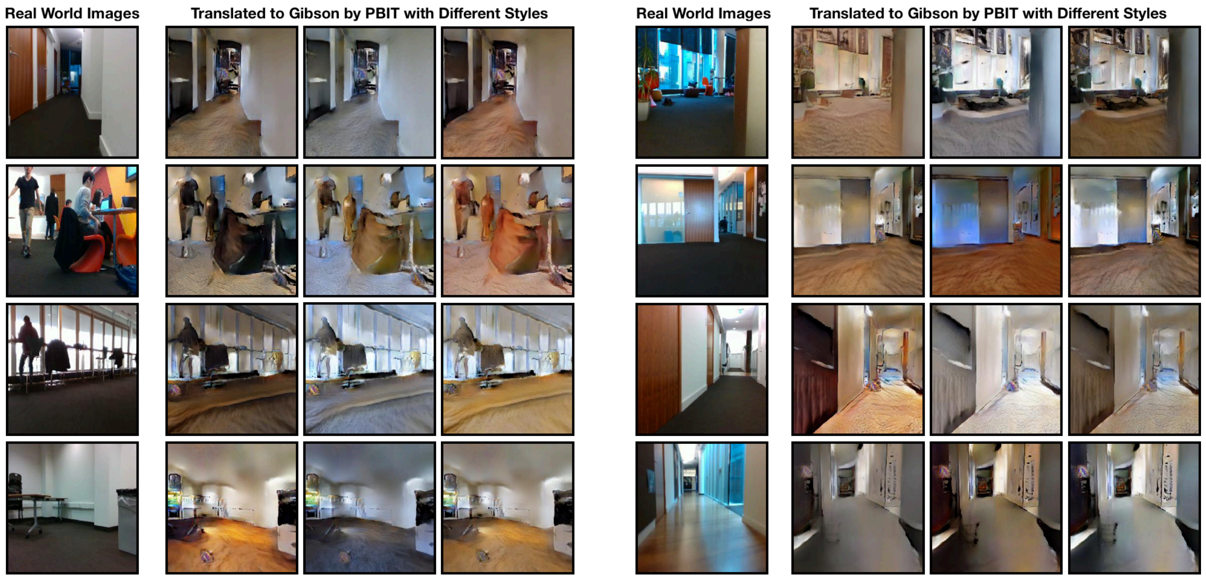

Real World Images
Translated to Gibson by PBIT with Different Styles
Real World Images
Translated to Gibson by PBIT with Different Styles

Figure 4.5: **Real World Images Translated to Gibson.** The first and fifth columns are the input images from Real World during test time. The other columns are images translated to Gibson domain by PBIT. Although the agent has never been trained to navigate with Real World images, it can recognize the translated images (which preserve the policy-relevant features) and perform well on PointGoal in Real World.

In Figure 4.5, we visualize some examples of images seen by the agent in the real-world and their translation to the source Gibson domain using our PBITmodel. The examples indicate that the model generates good translations similar to images in the Gibson domain. For example, the dark grey carpet floors in the office space scenes in the real-world are successfully translated to brown floors, representative of wooden floors of apartment scenes in the Gibson domain. At the same time, navigation relevant details such as the boundary between the floor and walls and free space area, are preserved during translation. In Figure 4.6, we show an example of a successful trajectory in the real-world using PBIT. It shows some of the images seen by the agent during the trajectory, the corresponding translations and a third-person view of the robot. The trajectory shows the PBIT is able to successfully navigate around the blue chair obstacle to reach the pointgoal.

Table 4.3: **Real-world results.** Table comparing the performance of baseline and our method under 20 goals within 3 scenes. We the average the collision rate, the success rate, the final stopping distance from the PointGoal in meters, and the average number of steps the robot takes to finish a trial excluding trials with collisions. Although both agent fail in first unseen scene where there is high ground reflection, our agent demonstrates better performance in terms of completion rate and average steps taken in the second unseen location. Our PBIT agent achieves overwhelmingly better performance in all metrics in the third scene since 151 sample images have been included in the transfer training set.

| Episode Specification | | | | Baseline: Direct Transfer | | | | PBIT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ep No | Dist | Angle | Obstacle in way | Steps | Collision | Stopping Distance | Success | Steps | Collision | Stopping Distance | Success |
| **Scene 1: Wooden corridor with intense ground reflection (Not in training set of PBIT)** | | | | | | | | | | | |
| 1 | 4.00 | 0.00 | FALSE | 99 | FALSE | 5.62 | FALSE | 99 | FALSE | 3.48 | FALSE |
| 2 | 2.83 | 45.00 | FALSE | 68 | TRUE | 4.12 | FALSE | 99 | FALSE | 1.80 | FALSE |
| SCENE AVG | - | - | - | 99 | 50% | 4.87 | 0% | **99** | **0%** | **2.64** | 0% |
| **Scene 2: Public kitchen area with high traffic (Not in training set of PBIT)** | | | | | | | | | | | |
| 3 | 2.00 | 0.00 | FALSE | 32 | TRUE | 0.50 | FALSE | 36 | FALSE | 0.18 | TRUE |
| 4 | 2.00 | 0.00 | FALSE | 35 | TRUE | 0.56 | FALSE | 10 | FALSE | 0.04 | TRUE |
| 5 | 2.24 | 333.43 | FALSE | 31 | FALSE | 0.02 | TRUE | 16 | FALSE | 0.07 | TRUE |
| 6 | 2.24 | 153.43 | FALSE | 38 | FALSE | 0.08 | TRUE | 43 | FALSE | 0.09 | TRUE |
| 7 | 4.12 | 345.96 | TRUE | 80 | TRUE | 4.10 | FALSE | 44 | TRUE | 1.96 | FALSE |
| 8 | 4.47 | 26.57 | TRUE | 99 | FALSE | 4.01 | FALSE | 99 | FALSE | 2.85 | FALSE |
| 9 | 4.47 | 26.57 | TRUE | 99 | FALSE | 4.49 | FALSE | 70 | FALSE | 0.14 | TRUE |
| 10 | 5.39 | 21.80 | TRUE | 99 | FALSE | 5.50 | FALSE | 99 | FALSE | 2.73 | FALSE |
| 11 | 2.83 | 45.00 | TRUE | 99 | FALSE | 3.57 | FALSE | 99 | FALSE | 2.84 | FALSE |
| SCENE AVG | - | - | - | 77.5 | 33.3% | 2.54 | 22.2% | **59** | **11.1%** | **1.21** | **55.6%** |
| **Scene 3: Large common area with few traffic (151 randomly-sampled images in training set of PBIT)** | | | | | | | | | | | |
| 12 | 2.83 | 45.00 | FALSE | 99 | FALSE | 3.41 | FALSE | 21 | FALSE | 0.15 | TRUE |
| 13 | 4.47 | 333.43 | TRUE | 99 | FALSE | 4.68 | FALSE | 41 | FALSE | 0.02 | TRUE |
| 14 | 4.00 | 0.00 | TRUE | 99 | FALSE | 4.43 | FALSE | 41 | FALSE | 0.13 | TRUE |
| 15 | 5.39 | 21.80 | TRUE | 10 | TRUE | 4.60 | FALSE | 32 | FALSE | 0.13 | TRUE |
| 16 | 4.24 | 315.00 | FALSE | 99 | FALSE | 6.20 | FALSE | 73 | FALSE | 0.14 | TRUE |
| 17 | 4.00 | 0.00 | FALSE | 36 | TRUE | 3.88 | FALSE | 63 | FALSE | 0.09 | TRUE |
| 18 | 1.41 | 45.00 | FALSE | 99 | FALSE | 1.95 | FALSE | 9 | FALSE | 0.19 | TRUE |
| 19 | 1.41 | 135.00 | FALSE | 50 | FALSE | 0.15 | TRUE | 29 | FALSE | 0.07 | TRUE |
| 20 | 4.47 | 26.57 | TRUE | 99 | FALSE | 4.73 | FALSE | 27 | FALSE | 0.19 | TRUE |
| SCENE AVG | - | - | - | 92 | 22.2% | 3.78 | 11.1% | **37.3** | **0%** | **0.13** | **100%** |
| AVG | - | - | - | 86.29 | 30% | 3.33 | 15% | **52.9** | **5%** | **0.87** | **70%** |

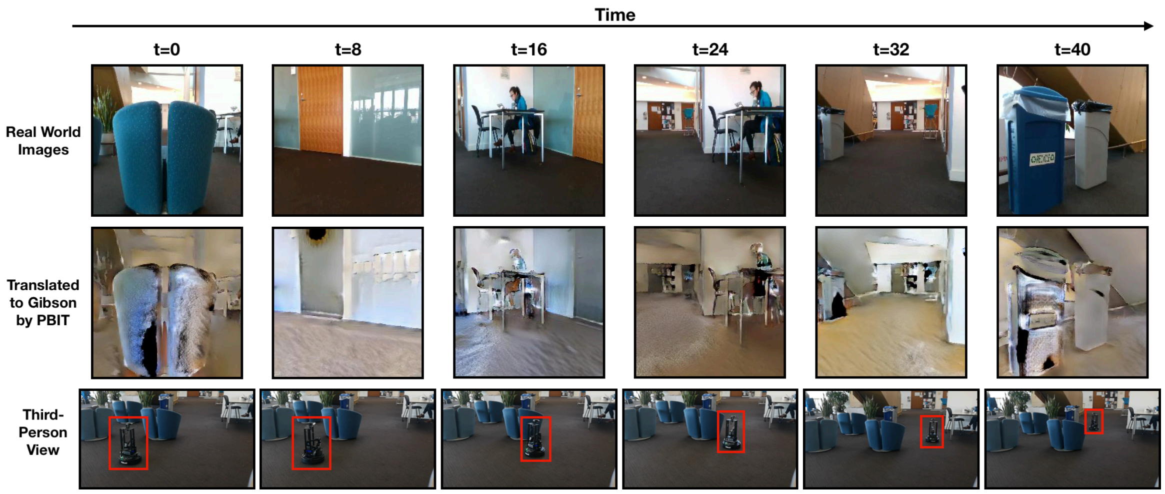
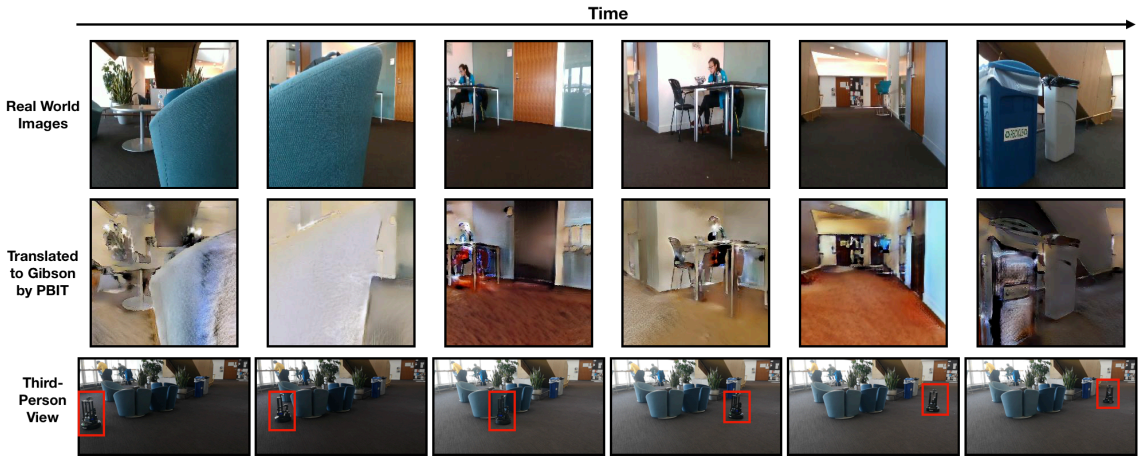
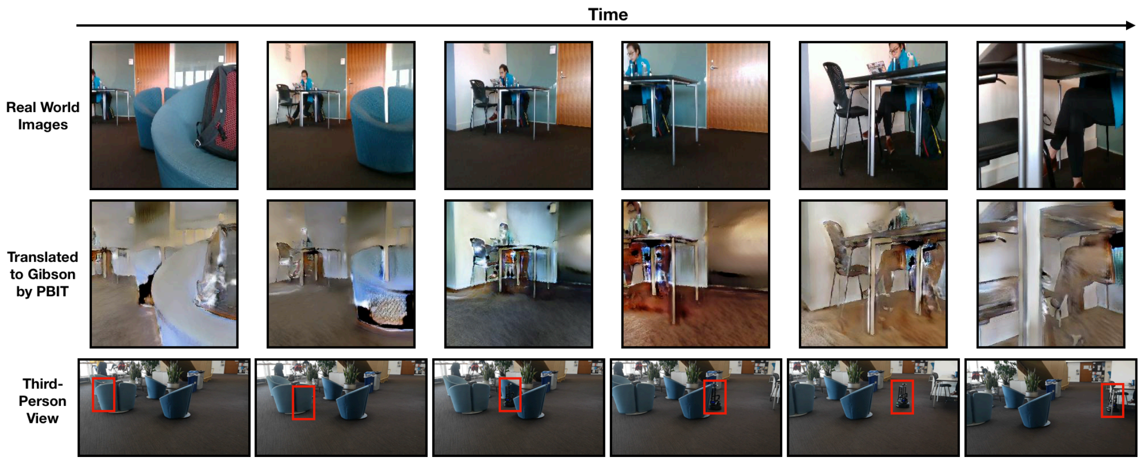


Figure 4.6: **Sample Real World Trajectory on PointGoal Task.** Figure contains raw inputs from Real World (row one), translated Gibson images by PBIT (row 2), and a third-person perspective from the back. The PBIT agent successfully reached it's destination (a trash can) by avoiding an obstacle (a chair) in its way.

## Additional Gibson to Real-world Trajectory Visualizations





### 4.2.3 Visualization of Policy Representations

We analyze the policy representation before and after translation by reducing the dimensionality of the policy representations using Principle Component Analysis (PCA) [47]. In Figures 4.7 and 4.8, we visualize the policy representations reduced to 2 dimensions using PCA in Replica and Real-World respectively. Both figures show that PBIT brings the representations of target domain Replica/Real-World images closer to the distribution of representations of Gibson images.

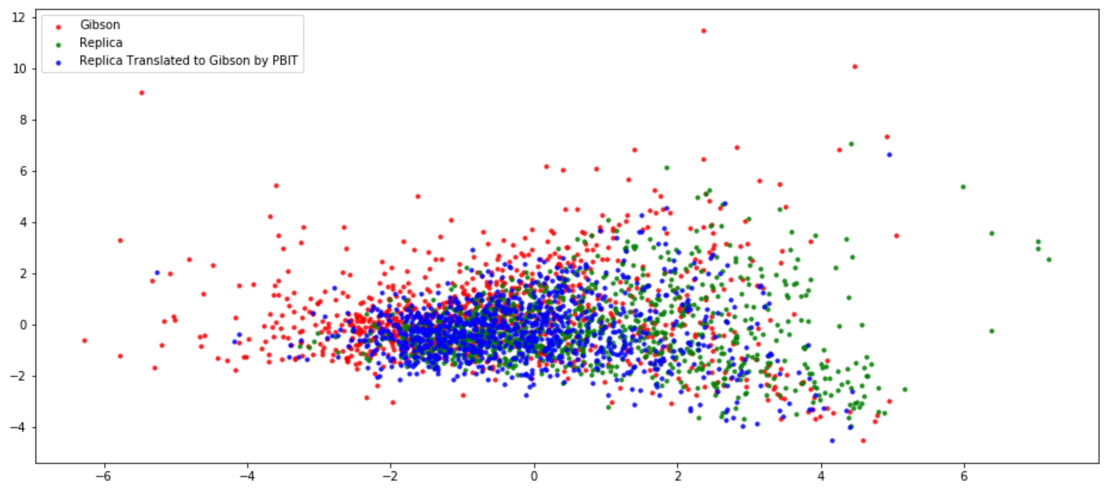


Figure 4.7: We use PCA to visualize and compare the 128-dimensional task-specific feature vectors produced by PointGoal agent trained in Gibson, when given Gibson images, Replica images, or Replica images translated to Gibson by PBIT as input. The figure shows the translated images by PBIT bridge the policy domain gap between Gibson and Replica.

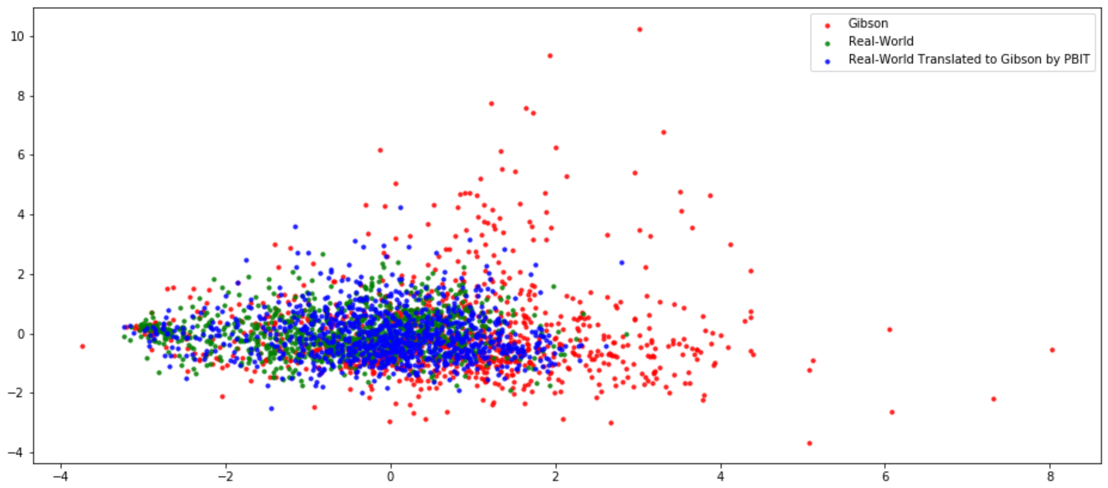


Figure 4.8: We use PCA to visualize and compare the 128-dimensional task-specific feature vectors produced by PointGoal agent trained in Gibson, when given Gibson images, Real-World images, or Real-World images translated to Gibson by PBIT as input. The figure shows the translated images by PBIT bridge the policy domain gap between Gibson and Real-World.

# Chapter 5

# Conclusion

In this thesis, we proposed domain adaptation method for transferring navigation policies from simulation to the real-world. Given a navigation policy in the source domain, our method translates images from the target domain to the source domain such that the translations are consistent with the task-specific and domain-invariant representations learnt by the given policy. Our experiments across two different tasks for domain transfer in simulation show that the proposed method can improve the performance on the transferred navigation policies over baselines. We also show strong performance of navigation policies transferred from simulation to the real-world using our method.

In fact, beyond the field of visual navigation, our proposed method is applicable to any learning-based policy parameterized by neural networks, since it does not take advantage of any knowledge specific to visual navigation tasks. Through our method, reinforcement learning agents in general can be adapted to new environments that they have never been trained on, given only a pool of unlabeled observations in those new environments.

## 5.1 Limitations and Future Directions

Our adaptation method relies on random observations in the target environment, which might be hard to gather in certain scenarios, and the quality of this dataset of random images greatly affects the quality of learned image translation. We need to assure that this dataset is not very biased. For example, in visual navigation tasks, we want this dataset to be a combination of images taken at different angles, rooms, distances to obstacles, etc. Otherwise the learned translation can only used in those scenarios that are similar to those in the dataset, and cannot to generalize well to other scenarios that the agent has to deal with when tested in the target environment. A high-quality balanced dataset might be difficult to obtain for certain tasks in the real world. Future experiments can be done to evaluate the performance of our method under different dataset scenarios (different sizes, distribution, bias, etc.).

Also, due to the intrinsic instability of current reinforcement learning algorithms, unsupervised learning, and generative adversarial networks, we simplifies our training process, where we train the reinforcement learning policy first in the source environment, and then fix the learned policy and use it to train the image translation model. However, the most ideal setting would be to train the whole thing end-to-end, which is worth experimenting with in the future.

Besides, there is more information that the image translation model can further take advantage of, but we haven't incorporated it due to the complexity of the project. For example, though the images observed in the target environment are random, we still have the full access to the source environment (the simulation), which contains valuable information like consecutive images in the same trajectory, metadata about scenes and floor-plans, etc. When training the image translation model, our method uses only random images in the source environment as well. It is interesting to see in the future how these extra information can be exploited without making our proposed method less general.

We also tried experimenting our method with adaptation between multimodal sensors. In visual navigation tasks, we consider depth camera agents as the source environment, and the RGB camera agents as the target environment. In general, depth camera agents perform and generalize to new environments much better than RGB camera agents, and thus we want to transfer a policy taking depth images as input to a policy taking RGB images as input. However, our unsupervised adaptation method fails in this setting, because the domain gap between depth and RGB is too large. Moreover, RGB images contain much more noisy and diverse information than depth images. We believe huge unbalance between the entropy of the source domain and the entropy of the target domain is very technically challenging to unsupervised image translation methods.

Finally, the performance achieved by our adaptation method still has a lot of room for improvement. Perhaps the problem setting we are considering is too restrictive, since in practice perhaps it is not very hard to incorporate a small amount of additional human supervision. For example, if our translation model can have access to some paired images between the source and the target environments, the performance might be boosted by a lot. Our proposed method works in a more model agnostic fashion, but it would be interesting to see how our proposed method can be combined with some paired data and domain knowledge.

# Bibliography

[1] Locobot · an open source low cost robot. URL `http://www.locobot.org/`. 4.1, 4.1.5, 4.2.2

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018. URL `http://arxiv.org/abs/1807.06757`. 4.1.1

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 1.2, 2

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1.2

[5] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 1.2, 2

[6] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklXn1BKDH`. 4.1.1

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 1.2, 2

[8] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547, 2019. 1.2

[9] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. 1 2019. 7th International Conference on Learning Representations, ICLR 2019 ; Conference date: 06-05-2019 Through 09-05-2019. 4.1.1

[10] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for naviga-

tion. *arXiv preprint arXiv:1903.01959*, 2019. 2

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

[12] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. In *ICLR*, 2017. 2

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

[14] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018. 1.2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[16] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098, 2018. 2

[17] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017. 1.2, 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4.1.2

[19] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017. 1.2, 2

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 4.1.4

[21] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018. 1.2, 2, 3.2, 3.3, 4.1.4

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[23] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017. 2

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 4.1.2, 4.1.4

[25] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1.2

[26] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 2

[27] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51, 2018. 2

[28] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017. 1.2, 2

[29] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *ICLR*, 2017. 2

[30] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019. 4.2.2

[31] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018. 1.1

[32] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 2

[33] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1.1, 4.1, 4.1.1, 4.1.2, 4.1.4, 4.2.1

[34] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015. 4.1.2

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4.1.2

[36] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017. 2

[37] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. URL `http://arxiv.org/abs/1712.01815`. 1.1

[38] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green,

Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 1.2, 4.1

[39] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016. 2

[40] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. Adapting deep visuomotor representations with weak pairwise constraints. *arXiv preprint arXiv:1511.07111*, 2015. 2

[41] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4.1.4

[42] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019. 1.1

[43] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4.1.4

[44] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019. 1.2

[45] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, 2019. 1.1

[46] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Decentralized distributed ppo: Solving pointgoal navigation. In *ICLR*, 2020. 1.2

[47] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 4.2.3

[48] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents

with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 1.2

[49] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2769–2779, 2019. 1.2

[50] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR*, 2017. 2

[51] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. 1.2, 4.1

[52] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 1.2

[53] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017. 2

[54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 1.2, 2, 3.3, 4.1.4

[55] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in neural information processing systems*, pages 465–476, 2017. 2

[56] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017. 1.2, 2