Sim-to-Real: Autonomous Robotic Control

Technical Report

Zhang-Wei Hong[1], Yu-Ming Chen[1], Hsuan-Kung Yang[1], Hsin-Wei Hsiao[1], and Sih-Pin Lai[1]
[1]*Elsa Lab, Department of Computer Science, National Tsing Hua University*
{williamd4112, allenchen0958, hellochick, wayne1029jihad, cindy6101}@gapp.nthu.edu.tw

Project Description

Collecting training data from the physical world is usually time-consuming and even dangerous for fragile robots, and thus, recent advances in robot learning advocate the use of simulators as the training platform. Unfortunately, the reality gap between synthetic and real visual data prohibits direct migration of the models trained in virtual worlds to the real world. This project proposes a modular architecture for tackling the virtual-to-real problem. The proposed architecture separates the learning model into a perception module and a control policy module, and uses semantic image segmentation as the meta representation for relating these two modules. The perception module translates the perceived RGB image to semantic image segmentation. The control policy module is implemented as a deep reinforcement learning agent, which performs actions based on the translated image segmentation. Our architecture is evaluated in an obstacle avoidance task and a target following task. Experimental results show that our architecture significantly outperforms all of the baseline methods in both virtual and real environments, and demonstrates a faster learning curve than them. We also present a detailed analysis for a variety of variant configurations, and validate the transferability of our modular architecture. The architecture is implemented on an NVIDIA Jetson TX2 development board, and comprehensively evaluated on real robots.

I. Introduction

Visual perception based control has been attracting attention in recent years for controlling robotic systems, as visual inputs contain rich information of the unstructured physical world. It is usually necessary for an autonomous robot to understand visual scene semantics to navigate to a specified destination. Interpreting and representing visual inputs to perform actions and interact with objects, however, are challenging for robots in unstructured environments as colored images are typically complex and noisy [1, 2]. It is especially difficult to design a rule-based robot satisfying such requirements.

Both modular and end-to-end learning-based approaches have been proven effective in a variety of vision-based robotic control tasks [3–7]. A modular cognitive mapping and planning approach has been demonstrated successful in first-person visual navigation [5]. Vision based reinforcement learning (RL) has been attempted to train an end-to-end control policy for searching specific targets [7]. Applying end-to-end supervised learning to navigate a drone along a trail with human-labeled image-action pairs is presented in [6]. A methodology of end-to-end training a robot for object manipulation tasks using unlabeled video data is described in [4]. While these learning-based approaches seem attractive, they typically require a huge amount of training data. Collecting training data for learning a control policy in the physical world is usually costly and poses a number of challenges. First, preparing large amounts of labeled data for supervised learning takes considerable time and human efforts. Second, RL relies on trial-and-error experiences, which restrict fragile robots from dangerous tasks. Online training and fine-tuning robots in the physical world also tend to be time-consuming, limiting the learning efficiency of various RL algorithms.

An alternative approach to accelerate the learning efficiency and reduce the cost is training robots in virtual worlds. Most of the recent works on robot learning collect training data from simulators [3, 7–11]. However, the discrepancies between virtual and real worlds prohibit an agent trained in a virtual world from transferring to the physical world directly [8]. Images rendered by low-fidelity simulators are unlikely to contain as much rich information as real ones. Therefore, bridging the reality gap [11] has been a challenging problem in both computer vision and robotics. Many research efforts have been devoted to tackling this problem by either domain adaption (DA) [9, 12, 13] or domain randomization (DR) [3, 10, 11, 13]. Both of these methods train agents by simulators. DA fine-tunes simulator-trained models with real world-data. DR, on the other hand, trains agents with randomized object textures, colors, light conditions, or even physical dynamics in virtual worlds. The diversified simulated data enable the agents to generalize their models to the real world. Unfortunately, collecting the real-world data required by DA for control policy learning is exceptionally time-consuming. Although DR does not require real-world data, the technique lacks a systematic way to determine which parameters are to be randomized. Furthermore, DR requires considerably large number of training samples in the training phase to achieve acceptable performance.

Instead of training a vision-based control policy in an end-to-end fashion, we propose a new modular architecture to address the reality gap in vision domain. We focus on the problem of transferring deep neural network (DNN) models trained in simulated virtual worlds to the real world for vision-based robotic control. We propose to separate the learning model into a perception module and a control policy module, and use semantic image segmentation as the meta-state for relating these two modules. These two modules are trained separately and independently. Each of them assumes no prior knowledge of the other module. The perception module translates RGB images to semantic image segmentations. This module can be any semantic image segmentation models pre-trained on either commonly available datasets [14, 15] or synthetic images generated by simulators [16]. As annotated datasets for semantic segmentation are widely available nowadays and cross dataset domain adaptation has been addressed [17], training and fine-tuning a perception module for segmenting outdoor [14] or indoor [15] scenes have become straightforward. The control policy module employs deep RL methods and takes semantic image segmentations as its inputs. In the training phase, the control policy module only receives the image segmentations rendered by low-fidelity simulators. The RL agent interacts with the simulated environments and collects training data for the control policy module. While in the execution phase, the control policy module receives image segmentations from the perception module, enabling the RL agent to interact with the real world. As the image semantics rendered by the simulators and those generated by the perception module are invariant [18], the control policy learned from the simulators can be transferred to the real world directly. The proposed architecture provides better efficiency than the conventional learning-based methods, as the simulators are able to render image segmentations at a higher frame rate, enabling the RL agent to collect training or trial-and-error experiences faster. It also allows the RL agent to learn faster, as semantic image segmentation contains less noise than raw images.

Another advantage of the proposed architectures is its modularity. Both the perception and control policy modules can be flexibly replaced in a plug-and-play fashion, as long as their meta-state representation (i.e., image segmentation) formats are aligned with each other. Replacing the perception module allows a pre-trained control policy module to be applied to different scenes. Replacing the control policy module, on the other hand, changes the policy of the robot in the same environment. Furthermore, our architecture allows another visual-guidance module to be incorporated. The visual-guidance module adjusts the meta-state representations, such that the behavior of the robot can be altered online without replacing either the perception or control policy modules.

To demonstrate the effectiveness of the proposed modular architecture, we evaluate it against a number of baseline methods on two benchmark tasks: obstacle avoidance and target following. These tasks are essential for autonomous robots, as the former requires collision-free navigation, while the latter requires understanding of high-level semantics in the environments. To validate the transferability of our architecture, we train our models and the baseline methods in our simulators, and compare their performance on the benchmark tasks in both virtual and real worlds. Please note that we focus on RL-based robotic control policy, though the control policy module can be implemented by various options (e.g., imitation learning). Our results show that the proposed architecture outperforms all the baseline methods in all the benchmark tasks. Moreover, it enables an RL agent to learn dramatically faster than all the baseline methods. By simply replacing the perception module, we also show that no additional fine-tuning is necessary for the control policy module when migrating the operating environment. We further demonstrate the versatility of the visual-guidance module in Section V. We evaluate the pre-trained perception and control policy modules on two robots. These two robots are designed and developed to navigate in different scenarios. We use robot operating system (ROS) as the interface between the proposed architecture and the robotic platforms. All models are executed on an NVIDIA Jetson TX2 development board. The contributions of this work are as follows:

- A new modular learning-based architecture which separates the vision-based robotic learning model into a perception module and a control policy module.
- A novel concept for bridging the reality gap via the use of semantic image segmentation, which serves as the meta-state for relating the two modules.
- A simple methodology for directly transferring the control policy learned in virtual environments to the real world. The methodology does not require any domain randomization or task-specific real-world data.
- A way to migrate the operational environment of a robot without further fine-tuning its control policy module.
- A visual-guidance module for altering the behavior of the robot via adjusting the meta-state representations.

- A comprehensive evaluation and analysis of the proposed architecture on an NVIDIA Jetson TX2 development board and real robots.

The remainder of this report is organized as follows. Section II introduces background material. Section III walks through the proposed modular architecture. Section IV describes the evaluation tasks and environments. Section V presents the experimental results. Section VI explains the semantic segmentation models used in this work. Section VII describes the robotic platforms and their setups. Section VIII provides the details of the training methodologies employed in this work. Section IX explains the switching-target following task in detail. Section X provides the video demonstration link. Section XI concludes.

II. Background

This section briefly reviews background material on semantic image segmentation, RL, and policy gradient methods.

A. Semantic Image Segmentation

The goal of semantic segmentation is to perform dense predictions at pixel level. It has received great attention due to its applicability to a number of research domains (e.g., visual understanding, autonomous driving, and robotic control). Fully convolutional network (FCN) [19] pioneered to replace fully-connected (FC) layers by convolutional layers. A number of successive works have further enhanced the accuracy and efficiency [20–24], making them more promising for robotic control tasks. Unfortunately, relatively less attempts have been made to incorporate them into robotic learning models.

B. RL and Policy Gradient Methods

RL trains an agent to interact with an environment $\mathcal{E}$. An RL agent observes a state s from $\mathcal{E}$, takes an action a according to its policy $\pi(a|s)$, and receives a reward $r(s, a)$. $\mathcal{E}$ then transits to a new state s'. The agent's objective is to maximize its accumulated rewards G_t with a discounted factor γ, expressed as $G_t = \sum_{\tau=t}^{T} \gamma^{\tau-t} r(s_\tau, a_\tau)$, where t is the current timestep, and T is the total number of timesteps. Policy gradient methods are RL approaches that directly optimize π in the direction: $\nabla_\pi \sum_{t=0}^{T} \log \pi(a_t|s_t)(G_t - b(s_t))$, where $b(s_t)$ is a baseline function. A common choice for $b(s_t)$ is the value function $V^\pi(s) = \mathbb{E}\left[G_t|s_t = s, \pi\right]$. This approach is known as the actor-critic algorithm [25]. An asynchronous variant of the actor-critic algorithm, namely asynchronous advantage actor-critic (A3C) [26], has been proven data-efficient and suitable for vision based control tasks [7, 18, 27, 28].

III. Proposed Methodology

In this section, we present the modular architecture of our model. We first provide an overview of the model architecture, followed by the implementation details of the modules. Fig. 1 illustrates the proposed modular architecture. It consists of a perception module, a control policy module, and a visual guidance module. Semantic image segmentation s_t serves as the meta-state for related the former two modules, as shown in Fig. 1 (a). The perception module generates s_t from an RGB input image x_t, which comes from different sources in the training (x_t^{syn}) and execution (x_t^{real}) phases. The control policy module takes s_t as its input, and reacts with a_t according to π. The visual guidance module enables high-level planning by modifying s_t, as shown in Fig. 1 (b).

A. Perception Module

The main function of the perception module is to generate s_t, and passes it to the control policy module. In the training stage, s_t is rendered by a segmentation shader from a synthetic image x_t^{syn}. While in the execution phase, the perception module could be any semantic segmentation model. Its function is expressed as $s_t = \phi^{percept}(x_t^{real}; \theta^{percept})$, where $\phi^{percept}$ represents the semantic segmentation model, x_t^{real} is the RGB image from the real world, and $\theta^{percept}$ denotes the parameters of the perception module. We directly employ existing models, such as DeepLab [20] and ICNet [21], and pre-train them on datasets ADE20K [15] and Cityscape [14] for indoor and outdoor scenes, respectively. The format of meta-state representation s_t is the same for both phases. The effect of using different semantic image segmentation models as the perception module is analyzed in Section V.

The benefit of using semantic segmentation as the meta-state is threefold. First, although the visual appearances of x_t^{real} and x_t^{syn} differ, their semantic segmentations are almost identical [18] (Fig. 1 (a)). This allows us to use semantic

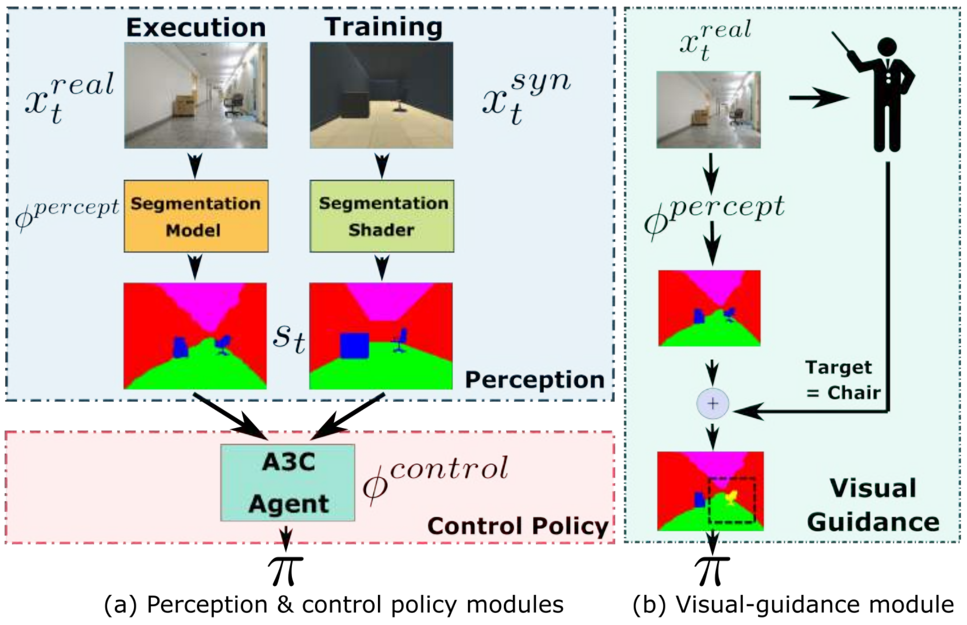


Fig. 1 Model architecture

segmentation to bridge the reality gap. Second, semantic segmentation preserves only the most important information of objects in an RGB image, resulting in a more structured representation of a scene. It also provides much richer information than other sensory modalities, as they typically fail to cover crucial high-level scene semantics. Third, the perception module in the execution phase only requires a monocular camera, which is a lot cheaper than depth cameras or laser range finders. As a result, semantic segmentation is inherently an ideal option for meta-state representation.

B. Control Policy Module

The control policy module is responsible for finding the best control policy π for a specific task. In this work, it employs the A3C algorithm, and takes s_t as its inputs (Fig. 1 (a)). In the training phase, the model $\phi^{control}$ is trained completely in simulated environments, and learns π directly from s_t rendered by low-fidelity simulators. While in the execution phase, it receives s_t from the perception module, allowing the RL agent to interact with the real world. As semantic segmentation is used as the meta state representation, it enables virtual-to-real transferring of π without fine-tuning or DR.

C. Visual Guidance Module

The use of scene semantics as the meta-state gives the proposed architecture extra flexibility. Our modular architecture allows a visual-guidance module to be augmented to guide the control policy module to perform even more complex tasks by manipulating the meta-state s_t. The manipulation can be carried out by either human or another RL agent. One can easily modify a robot's task through manipulating the class labels in s_t. This implies that a roadway navigation robot can be transformed to a sidewalk navigation robot by swapping the labels of the roadway and the sidewalk in s_t. The visual guidance module can also alter a target following robot's objective online by modifying the target label to a new one, as shown in Fig. 1 (b) (yellow chair). Note that visual guidance does not require any retraining, fine-tuning, or extra data in the above scenarios, while the previous works [5, 7] demand additional training samples to achieve the same level of flexibility.

IV. Virtual-to-Real Evaluation Setup

In this section, we present the evaluation setup in detail. We evaluate the proposed architecture both in virtual and real environments. We begin with explaining the simulator used for training and evaluation, followed by an overview of the baseline models and their settings. Finally, we discuss the tasks and scenarios for performance evaluation.

Action	(Linear (m/s), Angular (rad/s))
Forward	$(0.2, 0.0)$
Turn Left	$(0.2, -0.2)$
Turn Right	$(0.2, 0.2)$

Table 1 Mapping actions & robot velocities.

Model	Dimension	Format
Seg (Ours) Seg-S (Ours)	84×84×3 84×84×3×4	RGB Frame
DR-A3C DR-A3C-S	84×84×3 84×84×3×4	RGB Frame
Depth-A3C Depth-A3C-S	84×84×1 84×84×4	Depth Map
ResNet-A3C	224×224×3	RGB Frame

Table 2 Settings of model inputs.

Simulator. We use the Unity3D * engine to develop the virtual environments for training our RL agent. As Unity3D supports simulation of rigid body dynamics (e.g., collision, fraction, etc.), it allows virtual objects to behave in a realistic way. In the simulated environment, our RL agent receives its observations in the form of scene semantics in first-person perspective. It then determines the control actions (linear and angular velocities) to take accordingly. The virtual environment responds to the actions by transiting to the next state and giving reward signals. Please note that the geometric shape of our simulated agent is set to be similar to that of the real robot.

Tasks. The proposed model is evaluated against the baselines in virtual and real environments on two benchmark tasks: obstacle avoidance and target following. In both tasks, an RL agent starts at a random position in the environment at the beginning of each episode. At each timestep, it selects an action from the three possible options: moving forward, turning left, and turning right. Each action corresponds to the linear and angular velocities of the robot specified in Table 1.
(1) Obstacle Avoidance: The agent's goal is to navigate in a diverse set of scenes, and avoid colliding with obstacles.
(2) Target Following: The agent's objective is to follow a moving target (e.g., human) while avoiding collisions.

Scenarios. We evaluate the models with the following three scenarios. Fig. 2 illustrates a few sample scenes of them.
(1) Simple corridor: This scenario features indoor straight passages, sharp turns, static obstacles (e.g., chairs, boxes, tables, walls, etc.), and moving obstacles (e.g., human).
(2) Cluttered hallway: This scenario features a hallway crammed with static and moving obstacles for evaluating an agent's capability of avoiding collision in narrow space.
(3) Outdoor: This scenario features an outdoor roadway with sidewalks, buildings, terrain, as well as moving cars and pedestrians. This is used to evaluate how well the control policy trained for the tasks can be transfered from an indoor environment to an outdoor environment. Note that the agent is not allowed to move on the sidewalks in this scenario.

Models. For all the experiments, our model is trained with scene semantics generated from the simulator. We adopt DR, depth perception, and ResNet [29] as the baseline models. For the DR baseline, the texture of each mesh in a scene is randomly chosen from 100 textures. For the depth perception baseline, we use the depth maps generated from the simulator. We set the minimum and maximum sensible depths of the agent in the simulator to be 0.3m and 25m, respectively, which are close to those of the depth cameras (e.g., ZED) mounted on robots. We summarize the detailed settings in Table. 2. All the models in Table. 2 adopt the vanilla A3C architecture [26] except ResNet-A3C, in which the convolutional layers are replaced by ResNet-101 pre-trained on ImageNet [30]. ResNet-A3C is mainly used to justify that directly applies features extracted by ResNet as the meta-state does not lead to the same performance as ours. In Table. 2, we denote our model as Seg, and the other baselines as DR-A3C, Depth-A3C, and ResNet-A3C, respectively. The model named with "-S" (e.g., DR-A3C-S) indicate that these models concatenate the latest four frames as their inputs. Please note that we do not include DA in our experiments, as we only focus on control policies solely trained in simulated environments without further fine-tuning. We use Adam optimizer [31], and set both the learning rate and epsilon to 0.001. Each model is trained for 5M frames, and the training data are collected by 16 worker threads for all experimental settings. For the evaluation tasks in the real world, we train DeepLabv2 [20] on ADE20K [15] and

*Unity3D: https://unity3d.com/

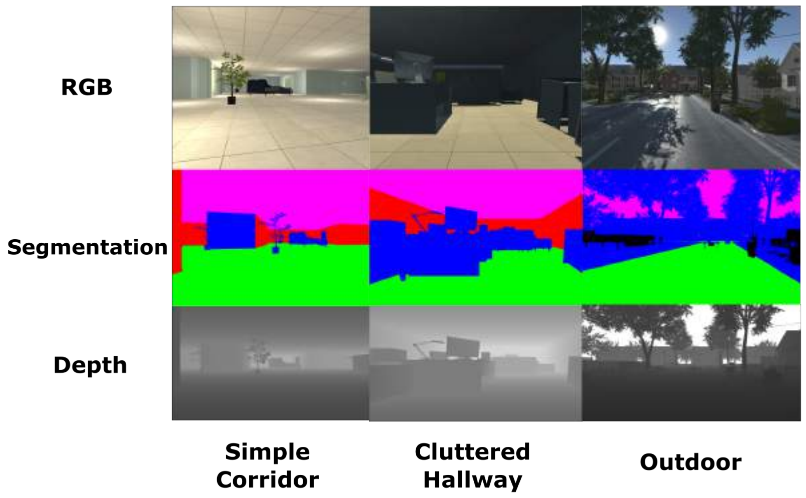
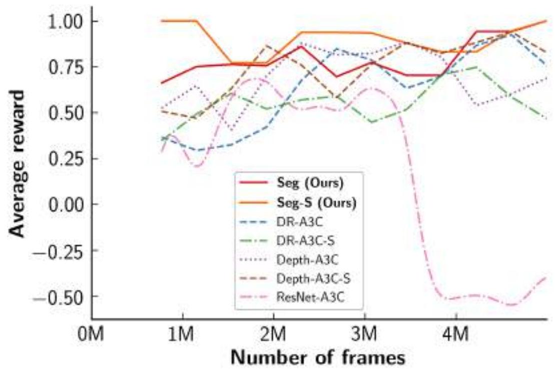
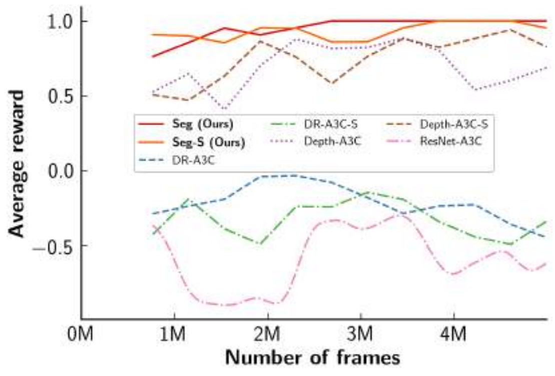


Fig. 2 Samples of evaluation scenes. From left to right: simple corridor, cluttered hallway, and outdoor. Top to bottom rows are RGB image, segmentation, and depth.



(a) Obstacle avoidance task



(b) Target following task

Fig. 3 Learning curve comparison.

ICNet [21] on Cityscapes [14] as the indoor and outdoor perception modules, respectively. Please note that the class labels in ADE20K are reduced as Table 5 for better accuracy and training efficiency.

V. Experimental Results and Analysis

In this section, we present the experimental results and their implications. We comprehensively analyze the results and perform ablative study of our methodology. In Section V.A and V.B, we compare the performance of our architecture and the baseline models in the tasks mentioned in Section IV. We demonstrate the concept of visual guidance in Section V.C. Finally, we present a series of ablative studies in Section V.D

A. Comparison in the Obstacle Avoidance Tasks

To compare the performance of our models against those of the baselines in the obstacle avoidance tasks, we designed a total of 16 indoor scenes for training the agents. Each scene features a pre-defined set of attributes (e.g., hallway, static obstacles), along with randomly located obstacles and moving objects. A scene is randomly selected for each training episode. Each episode terminates after 1,000 timesteps, or when a collision occurs. The agent receives a reward of 0.001 at each timesteps. Note that all the models are trained in simulation. Fig. 3 plots the learning curves of the models. Figs. 3 (a) and 3 (b) show the curves for the obstacle avoidance tasks and the target following tasks, respectively. While most of the models achieve nearly optimal performance at the end of the training phase, our methods learn significantly faster than the baseline models. This is due to the fact that scene semantics have simplified the

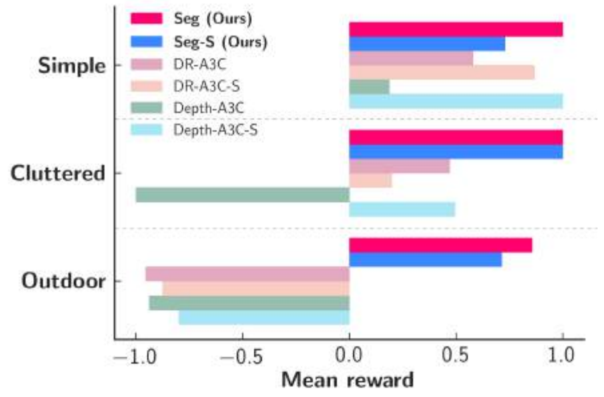


Fig. 4 Evaluation results in the obstacle avoidance tasks.

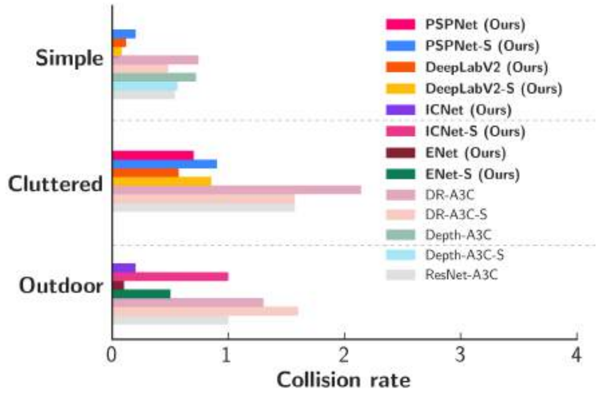


Fig. 5 Collision rate in the obstacle avoidance tasks.

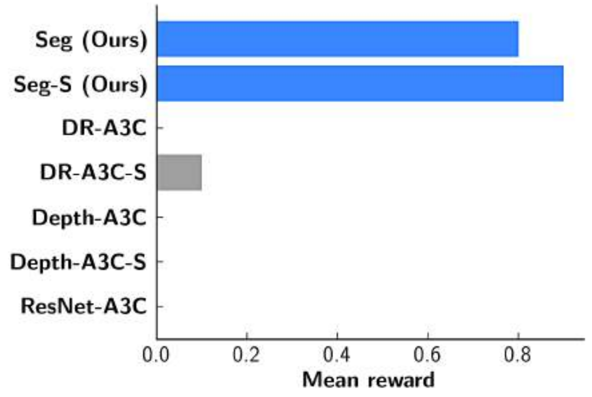


Fig. 6 Comparison of the target following tasks in simulation.

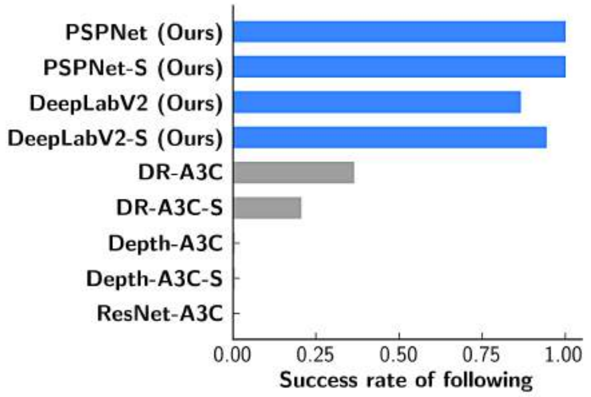


Fig. 7 Comparison of the target following tasks in real world.

original image representations to structured forms. On the contrary, the poor performance of ResNet-A3C indicates that the deep representations extracted by ResNet does not necessarily improve the performance. In addition, we observe that frame stacking has little impact on performance in the training phase. Depth-A3C shows a slightly better performance than DR-A3C and ResNet-A3C, implying the usefulness of depth information.

Evaluation in Simulation. Fig. 4 compares the mean rewards of the agents in three scenarios for seven types of models. The mean rewards are evaluated over 100 episodes, and the scenes are different from those used at the training phase. It can be observed that our methods outperform the other baseline models in all of the three scenarios. In *Simple Corridor*, noticeable performance can be observed from the other models. Most of the agents navigate well along the corridor, where there are relatively fewer obstacles compared to the other scenarios. In a more challenging scenario *Cluttered Hallway*, it can be seen that both Depth-A3C and Depth-A3C-S experience a considerable drop in performance. This is because depth map becomes too noisy in resolution to navigate in narrow space. In *Outdoor*, all models receive negative mean rewards, except for our methods. We observe that the baseline models tend to drive on the sidewalk, or rotate in place. In contrast, our models focus on driving on the roadway. As a result, we conclude that using semantic segmentations as the meta state enables a control policy to be transfered from an indoor environment to an outdoor environment.

Evaluation in the Real World. To test the virtual-to-real transferability of the models, we replicate the settings of the scenarios mentioned above in the real world. We measure the performance by recording the number of collisions per minute. We manually position the robot back to a random position when it collides with an obstacle. We report the results in Fig. 5. In *Simple Corridor*, we observe that Depth-A3C experience a huge performance drop from simulation to the real world, since the depth map estimated in the real world is too noisy for the model to adapt. The performances of the other baselines are similarly decreased, while our methods still maintain the same level of performance. We exclude Depth-A3C from the rest of the real world scenarios due to the poor quality of depth estimation of the depth cameras in complex environments. In *Cluttered Hallway*, while all of the models degrade in performance, our methods still significantly outperform the baselines. In *Outdoor*, we observe that both DR-A3C and ResNet-A3C fail to transfer their knowledge from simulation to the real world, and tend to bump into cars and walls. We also notice that frame stacking leads to lower performance, because it amplifies the errors resulting from the changes in environmental dynamics. Additionally, to demonstrate the robustness of the proposed architecture to the quality of segmentation, we compare the performance of our methods with different segmentation models. They include PSPNet [22] (high) and DeepLab [20] (low) for indoor environments, and ICNet [21] (high) and ENet [23] (low) for outdoor environments. Fig. 5 shows that our agents are not quite sensitive to the quality of image segmentations. Note that in *Outdoor* scenario, ICNet performs worse than ENet due to its inevitable computational delay. From the results in Figs. 3-5, we conclude that our methods are robust, generalizable, and transferable in the obstacle avoidance tasks.

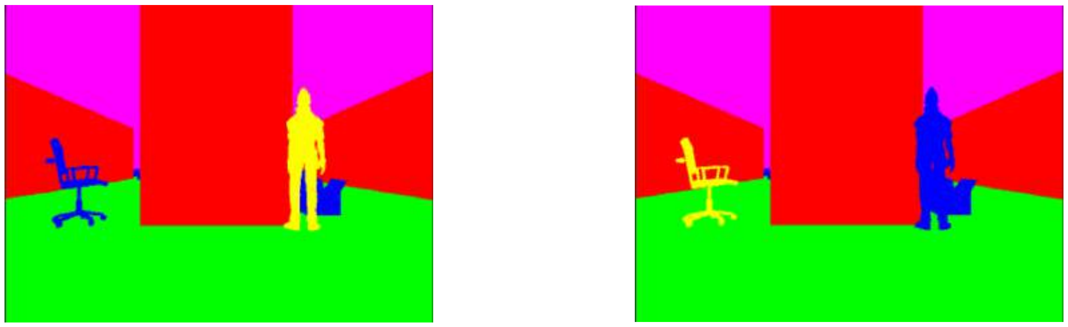
B. Comparison in the Target Following Tasks

We further compare the performance of our methods against those of the baselines in the target following tasks. For each episode, a simulated scene featuring static and moving obstacles is similarly selected at random. An episode ends immediately after 1,000 timesteps, or when a collision with the target/obstacles occurs. The agent receives a reward of 0.001 at each timestep if the target is within its sight, (1.0-"cumulative reward") if it touches the target, and 0.0 otherwise. At the beginning of each episode, the target is located in front of the agent. The target then navigates to a randomly chosen destination along the path computed by the A* algorithm. The ultimate goal of the agent is to avoid any collision on the way, and keep up with the target. Note that for a fair comparison, the geometry of the target is always fixed. Fig. 3 (b) plots the learning curves of our models and the baselines in the training phase. It can be observed that our models are much superior to the other baseline models. We also notice that our agents learn to chase the target person in the early stages of the training phase, while the baseline models never learn to follow the target. We conclude that the superiority of our models over the baselines results from semantic segmentation, which makes the target easily recognizable by the agent.

Evaluation in Simulation. To validate the generalizability, we evaluate the agent in a virtual scene, which is not included in the training sets. In each episode, the agent start from a random position in the scene. We evaluate the mean rewards over 100 episodes, and present the results in Fig. 6. Our models achieve higher mean rewards than all the baselines, which suggests the effectiveness of our models in capturing and tracking a fixed moving object. We observe that DR-A3C and ResNet-A3C always fail to follow the target at the fork of a corridor, because they are easily distracted by other objects. On the other hand, although Depth-A3C has higher mean rewards than the other baselines, it often fails when the target is far away from it. From the results in Fig. 6, we conclude that our models are generalizable and transferable to new scenes.

Evaluation in the Real World To evaluate if a learned policy can be successfully transfered to the real world, we further conduct experiments in real indoor environments, which consist of corridors, forks, sharp turns, and randomly placed static obstacles. The target is a human who moves within the environment. It is considered failed if the agent loses the target or collides with an obstacle, and successful if the agent follows the target to the destination. We report the success rate for each model in Fig. 7. The success rate is evaluated over ten episodes. The results show that all the baseline models fail in real world scenarios. Specifically, as the models trained with DR never learn a useful policy even in the simulated training environments, there is no doubt that they fail in the real world tests. As for the cases of Depth-A3C and Depth-A3C-S, despite the high mean rewards they attained in the training phase, the learned policies do not transfer successfully to the real world. During the evaluation, we observe that DR-A3C and ResNet-A3C agents treat the target person as an obstacle, and avoid the target as it comes close. We also notice that our models do not lose track of the target even at sharp turns. We deduce that this is due to the temporal correlations preserved by frame stacking.

Fig. 7 also provides evidence that our architecture is robust to the quality of image segmentations even in the target following tasks.



(a) Switching target to person (b) Switching target to chair

Fig. 8 Visual guidance for switching the following target.

	VW (mean reward)	RW (success rate)
Seg	0.824	80%
Seg-S	0.925	90%

Table 3 Performance of visual guidance in the virtual world (VW) and real world (RW) in the *Switching-Target Following* task.

C. Demonstration of Visual Guidance

In this section, we demonstrate the effectiveness of the visual guidance module of our modular architecture by switching the target in a *Switching-Target Following* task. We perform experiments in both virtual and real environments. Both environments are the same as the ones in the *Target Following* task. To demonstrate visual guidance, we randomly select an object in the environment, and re-label it as the target at the beginning of each episode. The initial location of the agent is chosen such that the selected target falls in the field of view (FOV) of the agent. The selected object is rendered to yellow, as shown in Fig. 8.

Fig. 8 illustrates the procedure of switching the target from a human to a chair. We measure the mean rewards over 1,000 episodes in the virtual environments, and the success rate over 10 episodes in the real world. Table 3 summarizes the results. The results reveal that our agents can successfully catch the randomly specified targets in both environments, no matter what target size and shape are.

D. Robustness to Redundant Labels

We further show that our control policy module can retain its performance, even if its input contains redundant labels. We conduct obstacle avoidance experiments in *Cluttered Hallway*, as it is the most complex scenario for our agents. When training in simulators, we add redundant class labels in semantic segmentations rendered by simulators. Given the same training time, the agent is able to obtain similar performance as the results shown in Fig. 3 (a). For evaluation in the real world, we compare the agent trained with redundant labels and the one used in Section V.A (i.e. trained with reduced labels). For perception in the real world, we adopt PSPNet as the perception module for both agents, and modify the output labels to be matched with the labels rendered by simulators. The experimental results in Table 4 show that the agents with redundant labels demonstrate better performance than their counterparts shown in Section V.A. We attribute this improvement to the richer context of the scenes in image segmentations with more labels.

VI. Semantic Segmentation Models

We employ two different semantic segmentation models, DeepLab-v2 [20] and ICNet [21], as the perception module to validate the proposed architecture (illustrated in Fig. 1) in the real world. The former is pre-trained on ADE20K [15], and is mainly used for indoor scenes. The latter is pre-trained on Cityscape [14], and is primarily used for outdoor scenes. The models for indoor and outdoor scenes are pre-trained separately due to the dissimilar nature of the two datasets. ADE20k covers a wide range of common indoor scenes and object categories. Cityscape, on the other hand, contains a diverse set of video frames recorded in street scenes.

To reduce the data dimension presented to the control policy module, we decrease number of output classes of the

Model	Collision rate
Reduced-Seg	0.7
Reduced-Seg-S	0.9
Redundant-Seg	0.1
Redundant-Seg-S	0.2

Table 4 The results in *Cluttered Hallway* with redundant labels

Reduced class labels	Original class labels in the dataset
Wall	Window, Door, Fence, Pillar, Sign board, Bulletin board
Floor	Road, Ground, Field, Path, Runway
Ceil (Background)	*Ceil
Obstacle	Bed, Cabinet, Sofa, Table, Curtain, Chair, Shelf, Desk, Plant
Target	(Depends on scenario)

Table 5 Class reduction from ADE20K dataset.

Obstacle avoidance	Target following
Wall	Wall
Floor	Floor
Ceil (Background)	Ceil (Background)
Obstacle	Obstacle
-	Target

Table 6 Reduced class labels

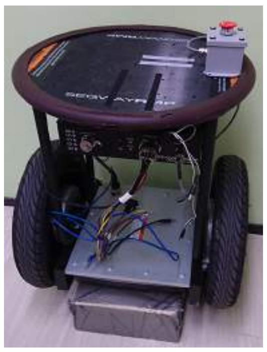
segmentation models to four for the obstacle avoidance task, and five for the target following task. Table 6 lists the class labels for the two tasks separately. Table 5 lists the mapping from the original class labels to the reduced class labels. Please note that reduction of classes is achieved by re-assigning the class labels. Neither modification nor fine-tuning is performed on the pre-trained semantic segmentation models.

VII. Robotic Platform Setup

Fig. 9 shows the robotic platforms for validating the proposed architecture in the real world. We evaluate the pre-trained perception and control policy modules on two robots. These two robots are designed and developed to navigate in different scenarios. Fig. 9(a) shows the Segway RMP 220 robotic platform used for the *Simple Corridor* and *Outdoor* scenarios in the obstacle avoidance task, as well as the target following task described in Section 5.3. Segway features a higher motor power and a self-balancing system, enabling it to handle a wider variety of road surfaces. Fig. 9(b) shows the Kobuki robotic platform. We mainly use Kobuki to perform experiments in the *Cluttered Hallway* scenario because it is smaller in size. Kobuki is especially suitable for navigation in indoor environments containing multiple obstacles. We use robot operating system (ROS) [32] as the interface between the proposed architecture and the robotic platforms. All models are executed on an NVIDIA Jetson TX2 development board. Motion commands are sent to the robots via Ethernet or USB ports. RGB images are fetched from NVIDIA Jetson TX2's onboard camera. Depth images are fetched from Intel Realsense ZR300. No other sensor is used in our experiments.

VIII. Training Methodologies

In this section, we present the training methodologies for the models listed in Section 5.1 in the simulated environments. Section VIII.A describes the domain randomization (DR) [11] method for training our DR-based agents. Section VIII.B explains the depth perception method for training the depth-based agents. Section VIII.C presents the rendering methodology for generating semantic image segmentations from the simulated environments. The generated


(a) Segway RMP 220

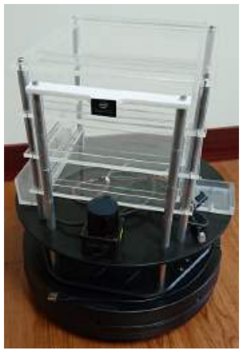

(b) Kobuki

Fig. 9 Robotic platforms

semantic image segmentations are fed to the control policy module of the proposed architecture at the training phase. Fig. 10 illustrates a few sample training scenes.

A. Domain Randomization Method

For DR-based agents (i.e., A3C, A3C-S, and ResNet-A3C), we randomly set the textures of objects and lighting conditions at the beginning of each episode. The texture of each mesh in the environment is randomly chosen from 100 textures including metal, wood, rock, etc. A single directional light is applied to the the simulated environment. where the intensity, color, and direction of the directional light are randomly determined for each episode. The intensity of the light is randomly set to a value between 15 to 60 in Unity. The color of the light is randomly selected from the color spectrum. The direction of the light is set as a unit vector from the origin to a random point on a unit sphere. For DR-based models that take as their inputs a single frame (i.e., A3C and ResNet-A3C), we directly feed a single synthetic RGB frame rendered by DR to the agents at each time step. For the DR-based model that take stacked frames as inputs at each time step (i.e., A3C-S), we concatenate the latest four rendered frames as the channels of an image, and feed it into the model.

B. Depth Perception Method

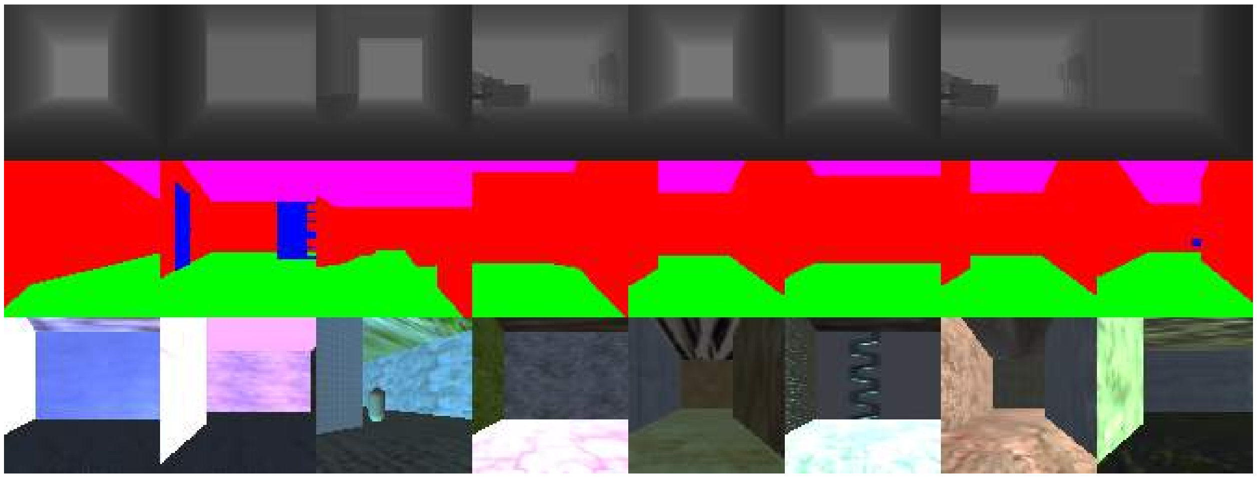
For the depth perception method, we set the minimum and maximum sensible depths of the agent in the simulator to be 0.3m and 25m, respectively, which are as same as those of the depth camera (i.e. Intel Realsense ZR300) mounted on the robots. We re-scale the measured depth values to fall in a range between 0 and 255. For the model that take a single depth map as its input (i.e., Depth-A3C), we straightly feed the depth map rendered by the simulator to the agent. For the model that take stacked depth maps as its input (i.e., Depth-A3C-S), we concatenate the latest four rendered depth maps as the channels of an image, and feed it into the model.

C. Image Segmentation Rendering Method

In the proposed architecture (Fig. 1), we feed the semantic image segmentations rendered by our simulator as the inputs to the control policy module, such that we are able to train the agents in the simulated environments. To generate semantic image segmentations from the simulated environments, we render objects of different categories with different colors. The color map used for rendering objects is carefully tuned such that it matches that of the outputs of the semantic segmentation models described in Section VI, as illustrated in Fig. 3.

IX. Switching Target Following Task

In this section, we demonstrate the flexibility of our modular architecture by switching the target in the *Target Following* task. The reward function, action space, and termination conditions are the same as those in the original setting. We perform experiments both in the simulated environments and the real world. The simulated environments are the same as those in *Target Following* task. For real-world experiments, we arrange the environments to be similar to those in the simulated environments. To demonstrate the flexibility of our architecture, we randomly select an object in the environment and re-label it as the target at the beginning of each episode. The initial location of the agent is chosen


(a) Obstacle avoidance

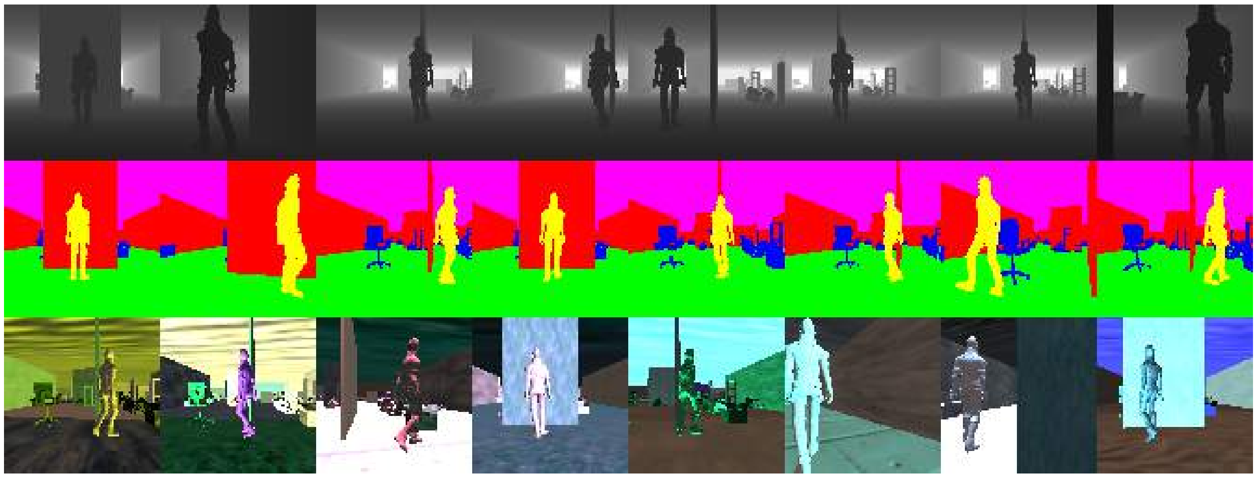

(b) Target following

Fig. 10 Samples of training scenes. Top to bottom rows are depth, segmentation and RGB images rendered with domain randomization technique.

such that the selected target falls in the field of view (FOV) of the agent. The selected object is rendered to yellow, as shown in Fig. 8. Fig. 8 illustrates the procedure of switching the target from a human to a chair. We measure the mean rewards over 1,000 episodes in the simulated environments, and the success rate over 10 episodes in the real world. Table 3 summarizes the results. The results reveal that our agents are able to successfully follow or catch the randomly specified targets in either the simulated environments or the real world, no matter that target size is. The results also indicate that our agents are able to identify and follow collision-free paths while executing the task. The experimental results demonstrates the flexibility of our modular architecture.

X. Video Demonstration

We have prepared a video demonstration and encourage interested readers to view the following link: `https://goo.gl/qcEju3`.

XI. Conclusion

In this report, we presented a new modular architecture for transferring policies learned in simulators to the real world for vision-based robotic control. We proposed to separate the model into a perception module and a control policy module, and introduce the concept of using semantic image segmentation as the meta state for relating these two modules. We trained our model with a standard RL algorithm, and did not apply any domain randomization technique. We performed experiments in two benchmark tasks: an obstacle avoidance task and a target following task, and demonstrated that our proposed method outperforms the baseline models in both virtual and real environments. We further validated the generalizability of our model in handling unfamiliar indoor and outdoor scenes, and the transferability of our model from simulation to the real world without fine-tuning. Finally, in the switching-target following task, we proved that our model is flexible such that the target can be easily switched by visual guidance. We evaluated the proposed architecture on real robots, and used robot operating system (ROS) as the interface between the proposed architecture and the robotic platforms. All models are executed on an NVIDIA Jetson TX2 development board.

References

[1] Maier, D., Hornung, A., and Bennewitz, M., "Real-time navigation in 3D environments based on depth camera data," *Proc. Int. Conf. Humanoid Robots (Humanoids)*, 2012, pp. 692-697.

[2] Biswas, J., and Veloso, M., "Depth camera based indoor mobile robot localization and navigation," *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 1697-1702.

[3] Sadeghi, F., and Levine, S., "CAD^2RL: Real single-image flight without a single real image," *arXiv:1611.04201*, 2016.

[4] Finn, C., and Levine, S., "Deep visual foresight for planning robot motion," *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 2786-2793.

[5] Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J., "Cognitive mapping and planning for visual navigation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S., "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," *arXiv:1705.02550*, 2017.

[7] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 3357-3364.

[8] James, S., and Johns, E., "3D simulation for robot arm control with deep Q-learning," *arXiv:1609.03759*, 2016.

[9] Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R., "Sim-to-real robot learning from pixels with progressive nets," *Proc. Conf. Robot Learning (CoRL)*, 2016, pp. 262-270.

[10] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv:1710.06537*, 2017.

[11] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain randomization for transferring deep neural networks from simulation to the real world," *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 23-30.

[12] Ghadirzadeh, A., Maki, A., Kragic, D., and Björkman, M., "Deep predictive policy training using reinforcement learning," *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 2351-2358.

[13] Zhang, F., Leitner, J., Milford, M., and Corke, P., "Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks," *arXiv:1709.05746*, 2017.

[14] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B., "The cityscapes dataset for semantic urban scene understanding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213-3223.

[15] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A., "Scene parsing through ADE20K dataset," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5122-5130.

[16] Richter, S. R., Hayder, Z., and Koltun, V., "Playing for benchmarks," *Proc. Int. Conf. Computer Vision (ICCV)*, 2017.

[17] Chen, Y.-H., Chen, W.-Y., Chen, Y.-T., Tsai, B.-C., Wang, Y.-C. F., and Sun, M., "No more discrimination: Cross city adaptation of road scene segmenters," *Proc. Int. Conf. Computer Vision (ICCV)*, 2017, pp. 2011-2020.

[18] You, Y., Pan, X., Wang, Z., and Lu, C., "Virtual to real reinforcement learning for autonomous driving," *arXiv:1704.03952*, 2017.

[19] Long, J., Shelhamer, E., and Darrell, T., "Fully convolutional networks for semantic segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431-3440.

[20] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L., "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2016.

[21] Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J., "ICNet for real-time semantic segmentation on high-resolution images," *arXiv:1704.08545*, 2017.

[22] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J., "Pyramid scene parsing network," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881-2890.

[23] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E., “ENet: A deep neural network architecture for real-time semantic segmentation,” *arXiv:1606.02147*, 2016.

[24] He, K., Zhang, X., Ren, S., and Sun, J., “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. vol. 37, no.9, pp. 1904-1916, 2015.

[25] Williams, R. J., “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *J. Machine learning*, Vol. vol. 8, no. 3-4, pp. 229-256, 1992.

[26] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., “Asynchronous methods for deep reinforcement learning,” *Proc. Int. Conf. Machine Learning (ICML)*, 2016, pp. 1928-1937.

[27] Wu, Y., and Tian, Y., “Training agent for first-person shooter game with actor-critic curriculum learning,” 2016.

[28] Parisotto, E., and Salakhutdinov, R., “Neural map: Structured memory for deep reinforcement learning,” *arXiv:1702.08360*, 2017.

[29] He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[30] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “ImageNet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097-1105.

[31] Kingma, D. P., and Ba, J., “Adam: A method for stochastic optimization,” *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.

[32] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y., “ROS: An open-source robot operating system,” *ICRA workshop on open source software*, 2009.