CAD^{2}RL: Real Single-Image Flight Without a Single Real Image

Fereshteh Sadeghi
University of Washington
fsadeghi@cs.washington.edu

Sergey Levine
University of California, Berkeley
svlevine@eecs.berkeley.edu

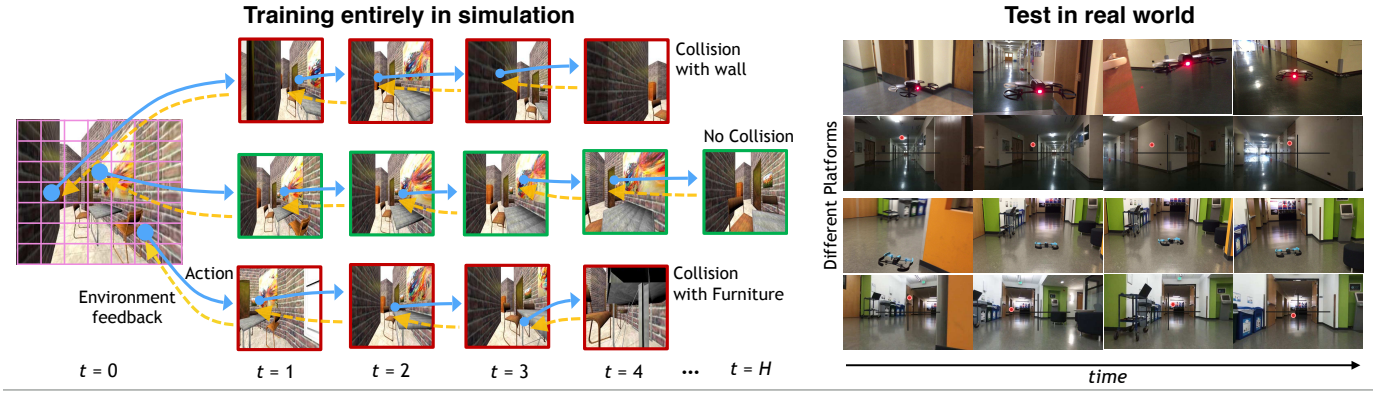

Fig. 1. We propose the **C**ollision **A**voidance via **D**eep **R**einforcement **L**earning algorithm for indoor flight which is entirely trained in a simulated **CAD** environment. Left: CAD^{2}RL uses *single image* inputs from a monocular camera, is exclusively trained in simulation, and does not see any real images at training time. Training is performed using a Monte Carlo policy evaluation method, which performs rollouts for multiple actions from each initial state and trains a deep network to predict long-horizon collision probabilities of each action. Right: CAD^{2}RL generalizes to real indoor flight.

***Abstract*—Deep reinforcement learning has emerged as a promising and powerful technique for automatically acquiring control policies that can process raw sensory inputs, such as images, and perform complex behaviors. However, extending deep RL to real-world robotic tasks has proven challenging, particularly in safety-critical domains such as autonomous flight, where a trial-and-error learning process is often impractical. In this paper, we explore the following question: can we train vision-based navigation policies entirely in simulation, and then transfer them into the real world to achieve real-world flight without a single real training image? We propose a learning method that we call CAD^{2}RL, which can be used to perform collision-free indoor flight in the real world while being trained entirely on 3D CAD models. Our method uses single RGB images from a monocular camera, without needing to explicitly reconstruct the 3D geometry of the environment or perform explicit motion planning. Our learned collision avoidance policy is represented by a deep convolutional neural network that directly processes raw monocular images and outputs velocity commands. This policy is trained entirely on simulated images, with a Monte Carlo policy evaluation algorithm that directly optimizes the network's ability to produce collision-free flight. By highly randomizing the rendering settings for our simulated training set, we show that we can train a policy that generalizes to the real world, without requiring the simulator to be particularly realistic or high-fidelity. We evaluate our method by flying a real quadrotor through indoor environments, and further evaluate the design choices in our simulator through a series of ablation studies on depth prediction. For supplementary video see: https://youtu.be/nXBWmzFrj5s**

I. Introduction

Indoor navigation and collision avoidance is one of the basic requirements for robotic systems that must operate in unstructured open-world environments, including quadrotors, mobile manipulators, and other mobile robots. Many of the most successful approaches to indoor navigation have used mapping and localization techniques based on 3D perception, including SLAM [3], depth sensors [44], stereo cameras [37], and monocular cameras using structure from motion [8]. The use of sophisticated sensors and specially mounting multiple cameras on the robot imposes additional costs on a robotic platform, which is a particularly prominent issue for weight and power constrained systems such as lightweight aerial vehicles. Monocular cameras, on the other hand, require 3D estimation from motion, which remains a challenging open problem despite considerable recent progress [13, 20]. In this paper, we explore a learning-based approach for indoor navigation, which directly predicts collision-free motor commands from monocular images, without attempting to explicitly model or represent the 3D structure of the environment. In contrast to previous learning-based navigation work [5], our method uses reinforcement learning to obtain supervision that accurately reflects the actual probabilities of collision, instead of separating out obstacle detection and control. The probability of future collision is predicted from raw monocular images using deep convolutional neural networks.

Using reinforcement learning (RL) to learn collision avoidance, especially with high-dimensional representations such as deep neural networks, presents a number of major challenges. First, RL tends to be data-intensive, making it difficult to use with platforms such as aerial vehicles, which have limited flight time and require time-consuming battery changes. Second, RL relies on trial-and-error, which means that, in order to learn to avoid collisions, the vehicle must experience at least a limited number of collisions during training. This can be extremely problematic for fragile robots such as quadrotors.

A promising avenue for addressing these challenges is to

train policies in simulation, but it remains an open question whether simulated training of vision-based policies can generalize effectively to the real world. In this work, we show that we can transfer indoor obstacle avoidance policies based on monocular RGB images from simulation to the real world by using a randomized renderer, without relying on an extremely high degree of realism or visual fidelity. Our renderer forces the network to handle a variety of obstacle appearances and lighting conditions, which makes the learned representations invariant to surface appearance. As the result, the network learns geometric features and can robustly detect open spaces.

In contrast to prior work on domain adaptation [35, 41], our method does not require any real images during training. We demonstrate that our approach can enable navigation of real-world hallways by a real quadrotor using only a monocular camera, without depth or stereo. By training entirely in simulation, we can also use a simple and stable RL algorithm that exploits the ability to reset the environment to any state. Figure 1 shows a diagram of our CAD^{2}RL algorithm. The algorithm evaluates multiple actions at each state using the current policy, producing dense supervision for the Q-values at that state. Training the Q-function to regress onto these Q-values then corresponds to simple supervised learning. This algorithm sidesteps many of the hyperparameter tuning challenges associated with conventional online RL methods, and is easy to parallelize for efficient simulated training.

The main contribution of our work is an approach for training collision avoidance policies for indoor flight using randomized synthetic environments and deep RL. We designed a set of synthetic 3D hallways that can be used to generate large datasets of randomized scenes, with variable furniture placement, lighting, and textures. Our synthetic data is designed for the task of indoor robot navigation and can also be used as a testbed for RL algorithms. Our proposed RL method is also a novel contribution of this work, and is particularly simple and well-suited for simulated training.

We present an extensive empirical evaluation that assesses generalization to the real world, as well as ablations on a supervised proxy task that studies which aspects of the randomized simulation are most important for generalization. Our simulated comparative evaluation shows that our approach outperforms several baselines, as well as a prior learning-based method that predicts turning directions [14]. Our real-world experiments demonstrate the potential for purely simulation-based training of deep neural network navigation policies. Although the policies trained entirely in simulation do experience some collisions in the real world, they outperform baseline methods and are able to navigate effectively around many kinds of obstacles, using only monocular images as input. We therefore conclude that simulated training is a promising direction for learning real-world navigation for aerial vehicles as well as other types of mobile robots.

II. Related Work

Any robotic system that must traverse indoor environments is required to perform basic collision avoidance. Standard methods for collision-free indoor navigation take a two step approach to the problem: first map out the local environment and determine its geometry, and then compute a collision-free path for reaching the destination [39]. This approach benefits from independent developments in mapping and localization as well as motion planning [38, 27, 4]. The 3D geometry of the local environment can be deduced using SLAM with range sensors [3], consumer depth sensors [44, 16], stereo camera pairs [37], as well as monocular cameras [8]. In [25], laser range scanned real images are used to estimate depth in a supervised learning approach and then the output is used to learn control policies. In [15] simultanouse mapping and planning using RGB-D images is done via a memory network. Reconstruction from monocular images is particularly challenging, and despite considerable progress [20, 13], remains a difficult open problem. In a recent approach, IM2CAD, CAD model of a room is generated from a single RGB image [18]. While the synthetic data generated by [18] could be used for various robotics simulations, the computational overhead makes it less suitable for autonomous indoor flight, where quick inference for finding open spaces is more critical than categorical exact 3D models.

In our work, we sidestep the challenges of 3D reconstruction by proposing a learning algorithm that can directly predict the probability of collision, without an explicit mapping phase. Learning has previously been used to detect obstacles for indoor flight [5, 19], as well as to directly learn a turn classifier for outdoor forest trail following [14]. In contrast to the work of [5], our method directly learns to predict the probability of collision, given an image and a candidate action, without attempting to explicitly detect obstacles. However, our approach still affords considerable flexibility in choosing the action: a higher-level decision making system can choose any collision-free action based, for example, on a higher-level navigational goal. This is in contrast to the prior work, which simply predicts the action that will cause the vehicle to follow a trail [14]. Unlike [14], our method does not require any human demonstrations or teleoperation.

Besides presenting a deep RL approach for collision avoidance, we describe how this method can be used to learn a generalizable collision predictor in simulation, such that it can then generalize to the real world. Simulated training has been addressed independently in the computer vision and robotics communities in recent years. In computer vision, a number of domain adaptation methods have been proposed that aim to generalize perception systems trained in a source domain into a target domain [42, 17]. In robotics, simulation to real-world generalization has been addressed using hierarchies of multi-fidelity simulators [12], priors imposed on Bayesian dynamics models [11]. At the intersection of robotics and computer vision, several works have recently applied domain adaptation techniques to perform transfer for robotic perception systems [41, 35, 34]. In contrast to these works, our method does not use any *explicit* domain adaptation. Instead, we show how the source domain itself can be suitably randomized in order to train a more generalizable model, which we experimentally show can make effective predictions on a range of systematically different target domains.

Our method combines deep neural networks for processing raw camera images [22] with RL. In the seminal work of Pomerleau [29], a fully connected neural network is used for generating steering commands for the task of road following

using raw pixels and laser range finder. Recently, a similar approach was proposed by [7] for a self-driving car. We also generate direction commands from raw visual inputs. However, unlike these prior works, we use RL and do not require any human demonstration data. Furthermore, our method commands the vehicle in 3D, allowing it to change both heading and altitude. Vision-based RL has previously been explored in the context of Q-iteration [33], and more recently for online Q-learning using temporal-difference algorithms [26]. However, these methods were evaluated primarily on synthetic video game domains. Several recent works have extended deep RL methods to real-world robotics applications using either low-dimensional estimated state [10] or by collecting an exhaustive real-world dataset under gridworld-like assumptions [43]. In contrast, we propose a simple and stable deep RL algorithm that learns a policy from raw monocular images and does not require seeing any images of the real-world test environment.

III. Collision Avoidance via Deep RL

Our aim is to choose actions for indoor navigation that avoid collisions with obstacles, such as walls and furniture. While we do not explicitly consider the overall navigation objective (e.g. the direction that the vehicle should fly to reach a goal), we present a general and flexible collision avoidance method that predicts which actions are more or less likely to result in collisions, which is straightforward to combine with higher-level navigational objectives. The input to our model consists only of monocular RGB images, without depth or other sensors, making it suitable for low-cost, low-power platforms, though additional sensory inputs could be added in future work. Formally, let $\mathbf{I}_t$ denote the camera observation at time t, and let $\mathbf{a}_t$ denote the action, which we will define in Section III-A. The goal of the model is to predict the Q-function $Q(\mathbf{I}_t, \mathbf{a}_t)$:

$$Q(\mathbf{I}_t, \mathbf{a}_t) = \sum_{s=t, \mathbf{a} \sim \pi}^{t+H} \gamma^{s-t} \mathcal{R}(\mathbf{I}_s, \mathbf{a}_s), \quad (1)$$

where $\gamma \in (0, 1)$ is the discount factor, and actions are assumed to be chosen by the current policy π, which we discuss in Section III-A. The horizon H should ideally be ∞, but in practice is chosen such that γ^H is small. $\mathcal{R}$ is the reward function and is equal to zero if collision event happens. Collisions are assumed to end the episode, and therefore can occur only once. Otherwise, the reward at time s is defined as $\min(1, \frac{d_s - r}{\tau_d - r})$, where r is the radius of the vehicle, d_s is the distance to the nearest obstacle at time s, and τ_d is a small threshold distance. This reward function encourages the vehicle to stay far from any obstacles. We could also use the latest Q-function estimate to label the last time step $t + H$, but we found this to be unnecessary to obtain good results. $Q(\mathbf{I}_t, \mathbf{a}_t)$ is learned using reinforcement learning, from the agent's own experience of navigating and avoiding collisions. Once learned, the model can be used to choose collision-free actions $\mathbf{a}_t$ simply by maximizing the Q-function. Training is performed entirely in simulation, where we can easily obtain distances to obstacles and simulate multiple different actions to determine the best one. By randomizing the simulated environment, we can train a model that generalizes effectively to domains with systematic discrepancies from our training environment. We will first describe the formulation of our model and reinforcement learning algorithm, and then present details of our simulated training environment.

A. Perception-Based Control

Our perception-based policy uses an action representation that corresponds to positions in image space. The image $\mathbf{I}_t$ is discretized into an $M \times M$ grid of bins, and each bin has a corresponding action, such that $\mathbf{a}_t$ is simply the choice of bin. Once chosen, the bin is transformed into a velocity command $\mathbf{v}_t$, which corresponds to a vector from the camera location through the image plane at the center of the bin $\mathbf{a}_t$, normalized to a constant target speed. Intuitively, choosing a bin $\mathbf{a}_t$ causes the vehicle to fly in the direction of this bin in image space. A greedy policy can use the model $Q(\mathbf{I}_t, \mathbf{a}_t)$ to choose the action with the highest expected reward. We will use $\pi(\mathbf{I}) = \mathbf{a}$ to denote this policy.

This representation provides the vehicle with enough freedom to choose any desired navigation direction, ascend and descent to avoid obstacles, and navigate tight turns. One advantage of this image-space grid action representation is the flexibility that it provides for general navigational objectives, since we could easily choose the bin using a higher-level navigational controller, subject to the constraint that the probability of collision not exceed some user-chosen threshold. However, in order to evaluate the method in our experiments, we simply follow the greedy strategy.

B. Initialization via Free Space Detection

In order to initialize our model with a reasonable starting policy, we use a heuristic pre-training phase based on free space detection. In this pretraining phase, the model is trained to predict $P(l|\mathbf{I}_t, \mathbf{a}_t)$, where $l \in \{0, 1\}$ is a label that indicates whether a collision detection raycast in the direction $\mathbf{v}_t$ corresponding to $\mathbf{a}_t$ intersects an obstacle. The raycast has a fixed length of 1 meter. This is essentially equivalent to thresholding the depth map by one meter. This initialization phase roughly corresponds to the assumption that the vehicle will maintain a predefined constant velocity $\mathbf{v}_t$. The model, which is represented by a fully convolutional neural network as described in Section III-D, is trained to label each bin with the collision label l, analogously to recent work in image segmentation [9]. The labels are obtained from our simulation engine, as described in Section IV.

C. Reinforcing Collision Avoidance

The initial model can estimate free space in front of the vehicle, but this does not necessarily correspond directly to the likelihood of a collision: the vehicle might be able to maneuver out of the way before striking an obstacle within 1 meter, or it may collide later in the future even if there is sufficient free space at the current time step, for example because of a narrow dead-end. We therefore use deep reinforcement learning to finetune our pretrained model to accurately represent $Q(\mathbf{I}_t, \mathbf{a}_t)$, rather than $P(l|\mathbf{I}_t, \mathbf{a}_t)$. To this end, we simulate multiple rollouts by flying through a set of training environments using our latest policy. Our score map of $M \times M$ bins, explained in III-A, determines the

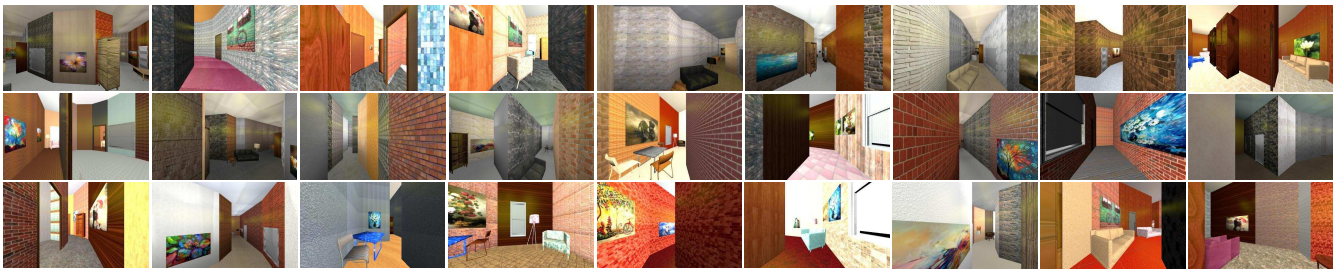

Fig. 2. Examples of rendered images using our simulator. We randomize textures, lighting and furniture placement to create a visually diverse set of scenes.

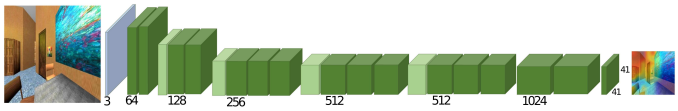

Fig. 3. We use a fully convolutional neural network to learn the Q-function. Our network, shown above, is based on VGG16 with dilated operations.

space of actions. Based on our score map, we consider a total of M^2 actions $\mathbf{a} = \{a^1, ..., a^{M^2}\}$ that can be taken after perceiving each observation $\mathbf{I}$. To generate the training set at each iteration, we sample a collection of states by placing the agent at a random location and with random orientation and generate a rollout of size K, given by $(\mathbf{I}_0, \mathbf{a}_0, \mathbf{I}_1, \mathbf{a}_1, ..., \mathbf{a}_{K-1}, \mathbf{I}_K)$. These states should in principle be obtained from the state distribution of the current policy. Using the model obtained from our pretraining step, the initial policy is simply $\arg\max_{i \in \{1,...,M^2\}} P(l|\mathbf{I}, a^i)$. We found that we could obtain good performance by sampling the states independently at random, though simply running the latest policy starting from an initial state distribution would also be a simple way to obtain the training states. Once the training states are obtained, we perform $M \times M$ rollouts from *each* training state using the policy π for every possible action $a^i, i \in \{1, ..., M^2\}$ and evaluate the return of a^i according to Equation (1). Since evaluating Equation (1) requires rolling out the policy for H steps for every action, we choose $H = 5$ to reduce computation costs, and instead use a simple approximation to provide smooth target values for $Q(\mathbf{I}, a^i)$.

This policy evaluation phase provides us with a dataset of observation, action, and return tuples $(\mathbf{I}_t, \mathbf{a}_t, Q(\mathbf{I}_t, \mathbf{a}_t))$, which we can use to update the policy. Since we evaluate every action for each image $\mathbf{I}_t$, the dataset consists of densely labeled images with Q values reflecting the expected sum of future rewards for the current policy π.

Our method can be interpreted as a modification of fitted Q-iteration [33], in the sense that we iteratively refit a Q-function estimator to samples, as well as a variant of modified policy iteration (MPI) [31] or Monte Carlo policy evaluation, in the sense that we estimate Q-values using multi-step rollouts of the current policy. To our knowledge, ours is the first algorithm of this class to be extended to deep reinforcement learning with raw image inputs. The particular details of the approach, including the evaluation of each action at each state, are specifically designed for our simulated training setup to exploit the capabilities of the simulation and provide for a simple and stable learning algorithm. We perform rollouts in simulated training hallways. This allows us to perform multiple rollouts from the state at each time step, perform ground truth collision detection raycasts for pretraining, and removes concerns about training-time collisions. Unlike conventional RL methods that perform rollouts directly in the test environment [26], we perform rollouts in simulated training hallways. However, this also means that our model must have generalization from the simulated training hallways to real-world environments at test time. To that, we developed a randomized simulated training environment, which we describe in the next section.

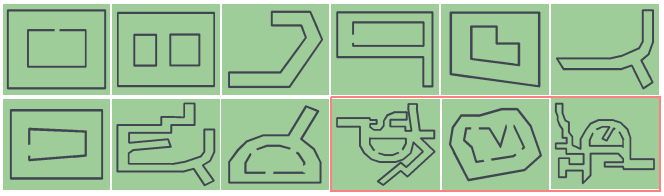

Fig. 4. Floor plans of the synthetic hallways. The last three hallways are used for evaluation while the first 9 are used during training.

D. Network Architecture

In order to represent the Q-function and the initial open space predictor, we use a deep fully convolutional neural network with dilation operations, built on the VGG16 [40] architecture following [9] as shown in Figure 3. The output score map corresponds to a grid of 41×41 bins, which constitutes the action space for deep reinforcement learning. The network is trained with stochastic gradient descent (SGD), with a cross-entropy loss function.

IV. Learning from Simulation

Conventionally, learning-based approaches to autonomous flight have relied on learning from demonstration [2, 1, 30, 34]. Although the learning by demonstration approach has been successfully applied to a number of flight scenarios, the requirement for human-provided demonstrations limits the quantity and diversity of data that can be used for training. Since dataset size has been demonstrated to be critical for the success of learning methods, this likely severely limits the generalization capacity of purely demonstration-based methods. If we can train flight controllers using larger and more diverse datasets collected autonomously, we can in principle achieve substantially better generalization. However, in order to autonomously learn effective collision prediction models, the vehicle needs to see enough examples of collisions during training to build an accurate estimator. This is problematic in real physical environments, where even a single collision can lead to damage or loss of the vehicle. To get the benefits of an autonomous learning from the agent's own experience and overcome the limitations of data collection in learning from demonstration method, we use a simulated training environment that is specifically designed to enable effective transfer to real-world settings.

We manually designed a collection of 3D indoor environments to form the basis of our simulated training setup. The environments were built using the Blender [6] open-source 3D modeling suite. Our synthetic dataset contains different hallways, shown in Figure 4, and represent a variety of structures that can be seen in real hallways, such as long straight or circular segments with multiple junction connectivity, as well as side rooms with open or closed doors. We use furnitures with various type and size to populate the hallways. The walls are textured with randomly chosen textures(e.g. wood, metal, textile, carpet, stone, glass, etc.), and illuminated with lights that are placed and oriented at random. In order to provide a diversity of viewpoints we render pretraining images by flying a simulated camera with randomized height and random camera orientation.

The randomization of the hallway parameters produces a very large diversity of training scenes, a sample of which can be seen in Figure 2. Although the training hallways are far from being photo-realistic, the large variety of appearances allows us to train highly generalizable models, as we will discuss in the experimental evaluation. The intuition behind this idea is that, by forcing the model to handle a greater degree of variation than is typical in real hallways (e.g., wide ranges of lighting conditions and textures, some of which are realistic, and some not), we can produce a model that generalizes also to real-world scenes, which might be systematically different from our renderings. That is, the wider we vary the parameters in simulation, the more likely we are to capture properties of the real world somewhere in the set of all possible scenes we consider. Our findings in this regard are aligned with the results obtained in other recent works [32], which also used only synthetic renderings to train visual models, but did not explicitly consider wide-ranging randomization of the training scenes.

V. Experimental Results

Despite that reinforcement learning evaluations emphasize mastery over generalization here our focus is on to evaluate the generalization capability of our proposed approach. Testing generalization is specially important from robotics perspective since the autonomous agent should be able to generalize to the diverse real-world settings. To this end, we evaluate our performance by running several experiments both in synthetic and real environments none of which had been seen during the training time. We compared our results against a set of baselines and also qualitatively evaluate our performance in various real-world scenarios. Additionally, we present an ablation study on a real-world RGB-D dataset to quantitatively evaluate our proposed randomized simulator for simulation to real-world transfer. In all the experiments (synthetic and real-world flights), CAD^{2}RL is trained on a fixed set of synthetic 3D models of hallways and in a fully simulated environment without being exposed to any real images.

A. Realistic Environment Evaluation

In order to evaluate how well such a model might transfer to a realistic environment, we used a realistic 3D mesh provided by [21]. Testing on this data can provide us a close proxy of our performance in a real indoor environment and also evaluates the generalization capability of our method in a systematically different environment than our training environments. Figure 6 shows the floorplan of this hallway, as well as several samples of its interior view. We generated 60 random initialization point from various locations in the hallways. These points are fixed and all baselines are evaluated on the same set of points so that their performance is directly comparable. Figure 6.a depicts the initialization points as red dots. The velocity of the quadrotor is fixed to 0.2 meters per time step in this experiment, and the maximum number of steps is set to 6000 which is equal to 1.2 kilometers.

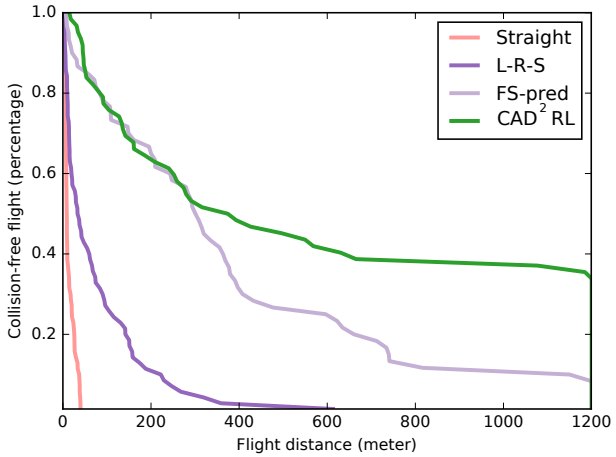

Fig. 5. Quantitative results on a realistically textured hallway. Our approach, CAD^{2}RL, outperforms the prior method (L-R-S) and other baselines.

Our aim is to evaluate the performance of our trained policy in terms of the duration of collision free flight. To do this, we run continuous episodes that terminate upon experiencing a collision, and count how many steps are taken before a collision takes place. We set the maximum number of steps to a fixed number throughout each experiment. We evaluate performance in terms of the percentage of trials that reached a particular flight length. To that end, we report the results using a curve that plots the distance traveled along the horizontal axis, and the percentage of trials that reached that distance before a collision on the vertical axis. This provides an accurate and rigorous evaluation of each policy, and allows us to interpret for each method whether it is prone to collide early in the flight, or can maintain collision-free flight at length. Note that achieving completely collision-free flight in all cases from completely randomized initial configurations is exceptionally difficult.

In this experiment, we compare against two baselines explained below. We also report the performance of our base Free Space prediction (FS-pred) controller to analyze the improvement obtained by incorporating deep reinforcement learning. In the FS-pred, the model described in III-B is used.

Straight Controller This lower bound baseline flies in a straight line without turning. In a long straight hallway, this baseline establishes how far the vehicle can fly without any perception, allowing us to ascertain the difficulty of the initialization conditions.

Left, Right, and Straight (LRS) Controller This baseline, based on [14], directly predicts the flight direction from images. The commands are discretized into three bins: “left,” “right,” or “straight,” and the predictions are made by a deep

convolutional neural network from raw images. For training the model, prior work used real-world images collected from three cameras pointing left, right and straight that were carried manually through forest trails. We simulated the same training setup in our training environments. We finetuned a VGG16 [40] model, pretrained with ImageNet classification. This method can be considered a human-supervised alternative to our autonomous collision avoidance policy.

1) Quantitative Evaluation: Figure 5 summarizes the performance of our proposed CAD^{2}RL method compared with other baselines. Our method outperforms the prior methods and baselines by a substantial margin. Qualitatively, we found that the LRS method tends to make poor decisions at intersections, and the coarse granularity of its action representation also makes it difficult for it to maneuver near obstacles. CAD^{2}RL is able to maintain a collision-free flight of 1.2 kilometers in about 40% of the cases, and substantially outperforms the model that is simply trained with supervised learning to predict 1 meter of free space in front of the vehicle. This experiment shows that although we did not use real images during training, our learned model can generalize to substantially different and more realistic environments, and can maintain collision-free flight for relatively long periods.

2) Qualitative Evaluation: To be able to qualitatively compare the performance and behavior of CAD^{2}RL with our perception based controller and the LRS method, we visualized the trajectory of the flights overlaid on the floor-plan of the hallway as shown in Figure 6. For this purpose, we sorted the trajectories of each method based on the traveled distance and selected the top 25 longest flights from each method. The trajectory colors show the flight direction at each point. The black dots indicate the locations of the hallway where collisions occurred. This visualization shows that CAD^{2}RL could maintain a collision-free flight in various locations in the hallway and has fewer collisions at the dead-ends, corners, and junctions compared with the other two methods. LRS often experienced collisions in corners and is more vulnerable to bad initial locations. The policy trained with free space prediction outperformed the LRS method, but often is trapped in rooms or fail near junctions and corners. This illustrates that the controller trained with RL was able to acquire a better strategy for medium-horizon planning, compared to the directly supervised greedy methods.

B. Real World Flight Experiments

We evaluated our learned collision avoidance model by flying a drone in real world indoor environments. These flights required flying through open spaces, navigating hallways, and taking sharp turns, while avoiding collisions with furniture, walls, and fixtures. We used two different drone platforms: the Parrot Bebop 1.0 and the Bebop 2.0, both controlled via the ROS Bebop autonomy package [28]. We perform real-world flight in several different scenarios and evaluate our performance both quantitatively and qualitatively.

1) Quantitative Evaluation: For quantitative evaluation, we ran controlled experiments on the task of hallway following. We fixed all the testing conditions while navigating the drone with either of the CAD^{2}RL and a baseline controller. The testing conditions include the initial velocity, angular speed, drone platform and the test environment. As was concluded from the experiments in section V-A, FS-pred was the strongest baseline, and we therefore included it as a comparison in this experiment. We ran experiments in two different buildings, Cory Hall and SDH (Sutardja Dai Hall), both located on the UC Berkeley campus. These buildings have considerably different floor plans, wall textures, and lighting conditions, as can be seen in Figure 7.c and Figure 7.d. Our testing environment in Cory Hall contained three turns and two junctions, while the SDH test environment had one turn and one junction. The width of the Cory hall hallway is ~ 3 meters while the SDH hallway is ~ 2 meters wide.

Table V-B1 summarizes the results. The safe flight time is given by the average length of a collision free flight in terms of distance or time between collisions. CAD^{2}RL experienced fewer collisions and has longer expected safe flight. This suggests that the CAD^{2}RL policy makes fewer mistakes and is more robust to perturbations and drift. Both methods performed better in Cory, since SDH has narrower hallways with glossy textureless walls as well as stronger air currents. While we fixed the test environment and the flying speed, the traveled distance and time is slightly different from one algorithm to another due to the fact that the algorithms generated different commands and navigated the drone to slightly different locations in the hallways.

2) Qualitative Evaluation: We performed real world flight in various indoor scenarios. We briefly explain each scenario and sequence snapshots are shown in Figure 7.

(a) Flying near furniture, around corners, and through a window: As shown in Figure 7.a. the drone starts from one end of a hallway connected to a small lounge area with furniture. The drone first flies toward the open lounge area, and then turns toward a corner of the room. There, it detects an opening in the wall which is visually similar to an open doorway or window, and adjust its height to fly through it. The drone then encounters a reflective glass door, which reflects the hallway behind it. Since no such structures were present during training, the reflective door fools the controller, causing the drone to crash into the door. Note that the controller navigates multiple structures that are substantially different, both visually and geometrically, from the ones encountered during simulated training.

(b) Flying up a staircase: Here, our goal is to evaluate the generalization capability of the controller to changes in elevation. A staircase provides a good example of this. To avoid colliding with the stairs, the drone must continuously increase altitude. As can be seen from the snapshots in the Figure 7.b, the controller produces actions that increase the altitude of the drone at each step along the staircase. Since we used an altitude limit for safety reasons, the drone only flew halfway up the staircase, but this experiment shows that the controller could effectively generalize to structures such as staircases that were not present during training.

(c) Navigating through narrow corridors: In this scenario, the drone flies through a corridor. The drone successfully takes a turn at Frames 1-4 in Figuree 7.c to avoid flying into a dead end. The corridors in this test scenario are narrow (~ 2 meters) and have strong air currents due to air conditioning. n

(d) Flying through junctions and rooms: Here, the drone

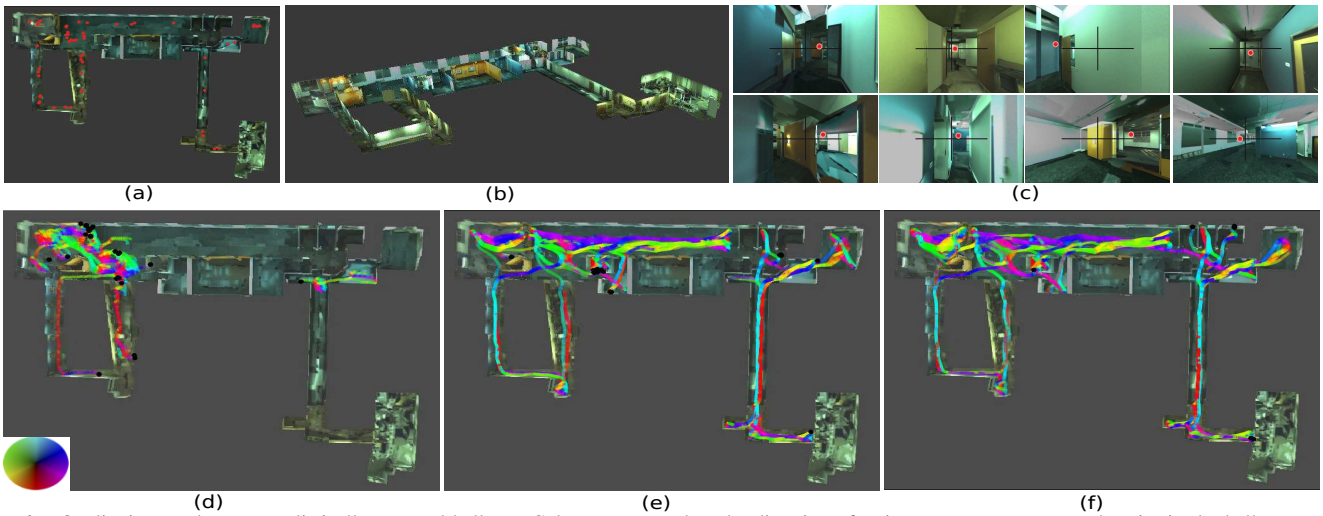

Fig. 6. Qualitative results on a realistically textured hallway. Colors correspond to the direction of trajectory movement at each point in the hallway as per the color wheel. (a) Red dots show flight initialization points (b) Overlook view of the hallway (c) Red dots show the control points produced by CAD^{2}RL. (d) LRS trajectories (e) Perception controller (FS-pred) trajectories (f) CAD^{2}RL trajectories.

TABLE I
REAL WORLD FLIGHT RESULTS.

Environment	Traveled Distance (meters)	Travel Time (minutes)	Collision (per meter)	Collision (per minute)	Safe Flight (meters)	Safe Flight (minutes)	Total Collisions
Cory FS-pred	162.458	12.01	0.080	1.081	12.496	0.924	13
Cory CAD^{2}RL	163.779	11.950	0.0366	0.502	27.296	1.991	6
SDH FS-pred	53.492	4.016	0.130	1.742	7.641	0.573	7
SDH CAD^{2}RL	54.813	4.183	0.072	0.956	13.703	1.045	4

navigates through a long hallway with junctions. At the end it enters a doorway which is connected to a study room. The controller successfully navigates the drone through the narrow door and into the room without colliding with the chairs.

(e) Flying through a maze of random obstacles in a confined space: We built a small U-shaped maze out of low obstacles in the lab. This maze is built using chairs and pieces of board with various appearance and colors. To prevent the drone from simply flying over the obstacles, we limited the altitude to 3 feet. Note that flying the drone at low altitude is challenging, as the air turbulence becomes significant and affects the drone's stability. The cardboard shifts due to air turbulence, and the open area is very narrow ($\sim$ 1 meter), making this a challenging test environment. The sequence in Figure 7.e shows that the controller successfully navigates the drone throughout the maze, making a turn near the red chair and turning back into the maze, without colliding.

(f) Avoiding dynamic obstacles: In this scenario, the drone begins in the lab with no obstacles and an altitude of around 3 feet. We then place a chair in the path of the drone, as seen in frames 3-4 of Figure 7.f. The controller recovers and avoids an imminent collision with the chair, passing it on the left.

The above qualitative evaluation study shows the generalization capability of our trained model and demonstrates the extent of the maneuvering skills learned by CAD^{2}RL. Although our model is specifically trained for the task of hallway navigation, the limited number of furniture items present in simulation also force the policy to be robust to oddly shaped obstacles, and train it to change altitude to avoid collisions. Navigating through the obstacles in the scenarios (a), (b), (e), and (f) required collision avoidance with general obstacles and other than just walls. We observed that our model could perform reasonably well in these cases, and could often recover from its mistakes, though particularly novel situations proved confusing.

C. Ablation Study for Real World Transfer

In this section, we present an ablation study to identify how important the randomization of the environment is for effective simulation to real-world transfer. Since conducting statistically significant real-world flight trials for many training conditions is time-consuming and subject to confounding factors (air currents, lighting conditions, etc.), we instead opted for a proxy task that corresponds to free-space prediction from real RGB images, with ground truth labels obtained via a depth camera. The goal in this task is to predict, for each point in the image, whether there is an obstacle within a certain threshold distance of the camera or if the pixel corresponds to free space. Although this proxy task does not directly correspond to collision-free flight, the reduced variance of the evaluation (since all methods are tested on exactly the same images) makes this a good choice for the ablation study. While we obtained reasonably good performance for avoiding collisions in the hallways, more detailed depth estimation [36, 24, 23] could also be used without loss of generality.

We used the same architecture as in Section III-B for the free-space prediction network and trained free-space predictors using rendered images from different simulated setups. We compared the obtained results against a similar network trained using our proposed randomized simulation. We used the same number of images sampled similarly from various locations in the hallways. The ablated networks are trained with images rendered from (a) a simulator that used **F**ixed **T**extures and

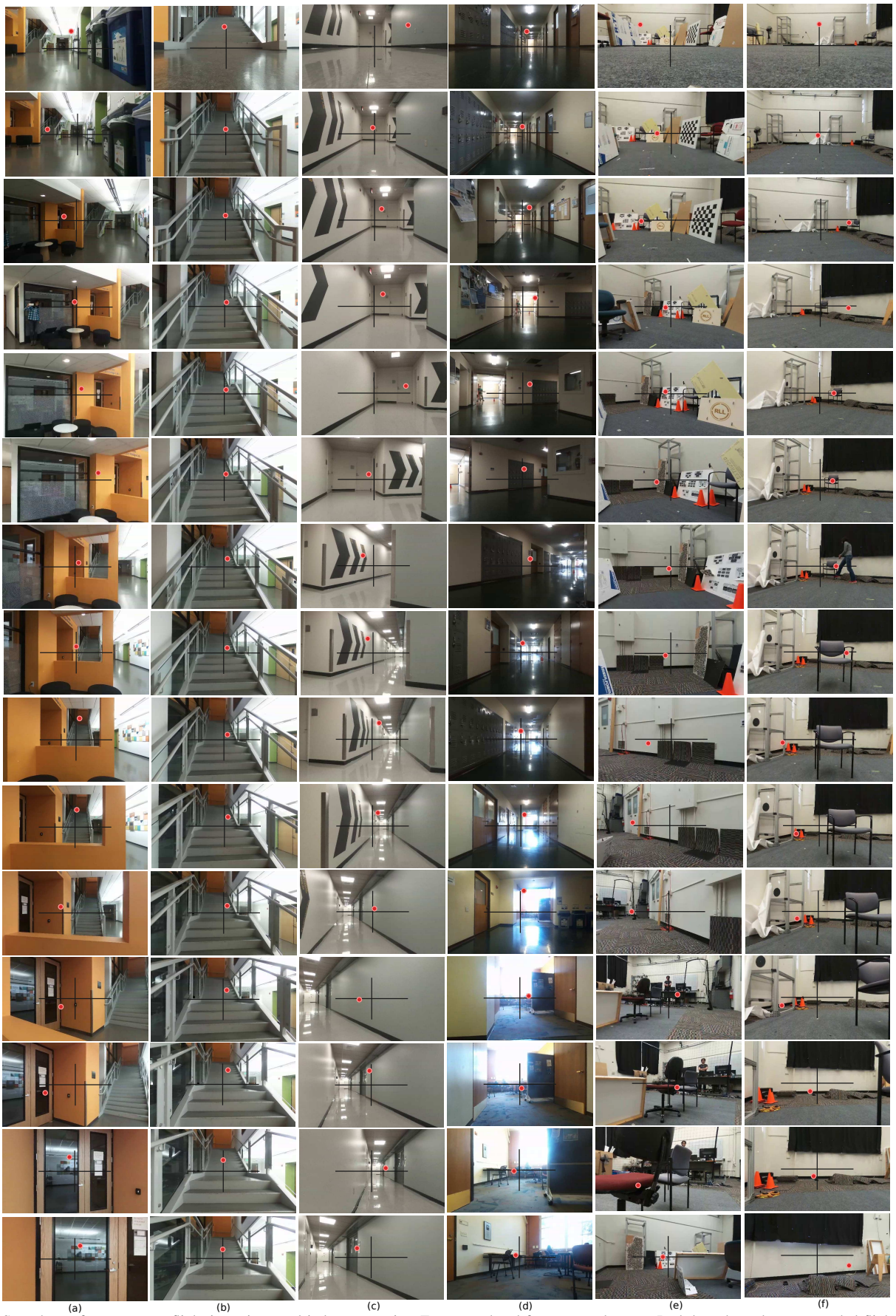

Fig. 7. Snapshots of autonomous flight in various real indoor scenarios. Frames ordered from top to bottom. Red dots show the commanded flight direction by CAD^{2}RL. (a) Flying near furniture, around corners, through a window; (b) Flying up a staircase; (c) Navigating in narrow corridors; (d) Navigating through junctions, fly through rooms; (e) Flying through a maze of random obstacles in a confined space; (f) Avoiding dynamic obstacles.

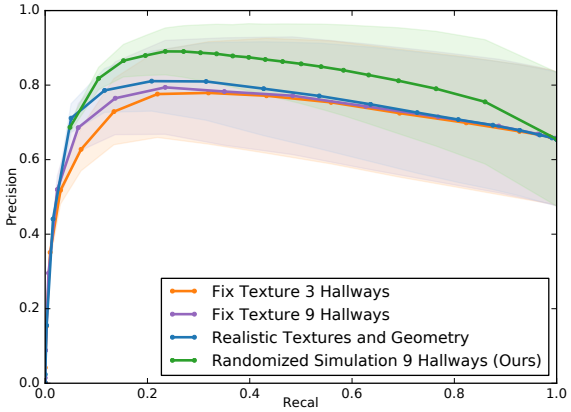

Fig. 8. Quantitative results for free-space prediction with different simulators. The network trained on randomized hallways outperforms networks trained on less randomized simulations and even on realistic textured hallways.

lighting, and only 3 of our training hallways (FT3); (b) **F**ixed **T**extures and lighting using all 9 training hallways (FT9) (c) the more **R**ealistic **T**extures and **G**eometry hallway provided by [21] (RTG) and (d) our approach, with randomized textures and lighting from 9 hallways. While (a), (b), and (d) are captured from synthetic hallways, in (c) the data is captured via a SLAM-based reconstruction system from the Cory Hall in the UC Berkeley campus. Therefore, this data has realistic geometry textured with natural images, and allows us to understand how the method would perform if trained on reconstructed RGBD data.

Our dataset contains RGB-D images captured from 5 hallways, in Cory Hall and SDH (Sutardja Dai Hall) located in UC Berkeley, with various lighting and texture conditions. We used a Kinect v2 and our dataset contains a total of 620 RGB-D images. Several example images of this data are shown in Figure 9. We used the depth channel to automatically annotate the images with free-space vs. non-free-space labels.

For each pixel in the input image, the network produces a probability value for free-space prediction. To evaluate the accuracy of free-space prediction we sweep a threshold from 0 to 1 to label each pixel using our prediction network. We compute the precision and recall at each threshold and plot the precision-recall curve as the performance metric. Precision is the number pixels correctly labeled as free-space divided by the total number of pixels predicted as free-space, while recall is the the number pixels correctly labeled as free-space divided by the total number pixels belonging to the free-space according to the ground truth. Since we use monocular images, there is a scale ambiguity in the size of hallways as well as in the range of sensible depths, which may not match between the simulated and real images. To overcome this ambiguity and to make a fair comparison, we labeled image pixels (for free-space vs non-free-space) by varying the depth threshold from 1 to 4 meters (steps of $\sim 30cm$) and computed the average precision/recall corresponding for each threshold over 13 runs.

Figure 8 shows the results, with shaded areas showing the standard deviation in precision. The network trained with the synthetic data rendered by our proposed randomized simulator outperforms the other networks. The images used for FT3 and FT9 are rendered on the same hallways as RT9, except that the textures and lighting are not randomized. As a result, these networks do not learn texture and color invariant features and

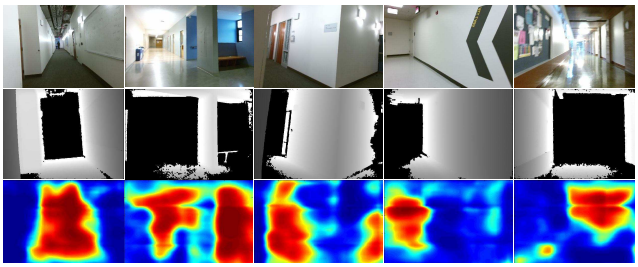

Fig. 9. Examples of the collected pairs of RGB (top row) and depth (mid row) data for the free-space test set. The free-space probability map predicted by our approach is shown in the bottom row.

cannot generalize well to the real images. In RTG, the images are rendered with realistic geometry and textures, and thus they are less affected by the scale ambiguity. Furthermore, the realistic textures in RTG are obtained from similar hallways as the one we used for our RGB-D test set. Despite this, the network trained on a realistic rendering of the same hallway actually performs worse than the network trained on our randomized simulator, by a substantial margin. For qualitative analyis, we show the probability map of free-space prediction obtained from our approach in the last row of Figure 8. We see that high probabilities are assigned to free spaces. Although the free-space prediction proxy task is not a perfect analogue for collision-free flight, these results suggest that randomization is important for good generalization, and that more realistic renderings should not necessarily be preferred to ones that are less realistic but more diverse.

VI. Discussion

We presented a method for training deep neural network policies for obstacle avoidance and hallway following, using only simulated monocular RGB images. We described a new simple and stable deep reinforcement learning algorithm for learning in simulation. We also demonstrate that training on randomized simulated scenes produces a model that can successfully fly and avoid obstacles in the real world, and quantitatively evaluated our randomized scenes on a proxy free-space prediction task to show the importance of randomization for real-world transfer. Our simulated evaluation further shows that our method outperforms several baselines, as well as a prior end-to-end learning-based method. Our aim in this work is to evaluate the potential of policies trained *entirely* in simulation to transfer to the real world, so as to understand the benefits and limitations of simulated training. To attain the best results in real environments, future work could combine simulated training with real data. Extending our approach via finetuning or domain adaptation is therefore a promising direction for future work that is likely to improve performance substantially, and lead to effective learned real-world visual navigation policies using only modest amounts of real-world training. Our approach could incorporate data from other sensors, such as depth cameras, which should improve the performance of the learned policies.

Acknowledgment

The authors would like to thank Larry Zitnick for helpful discussions and insightful remarks. This work was made possible by an ONR Young Investigator Program Award and support from Google, NVIDIA, and Berkeley DeepDrive.

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *NIPS*, 2006.

[2] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

[3] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unstructured and unknown indoor environments. In *EMAV*, 2009.

[4] Andrew J Barry and Russ Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *ICRA*. IEEE, 2015.

[5] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *ICRA*, 2011.

[6] Blender Community. Blender: Open Source 3D modeling suit. http://www.blender.org.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[8] K. Celik, S.J. Chung, M. Clausman, and A. Somani. Monocular vision SLAM for indoor aerial vehicles. In *IROS*, 2009.

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[10] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. *arXiv preprint arXiv:1609.07845*, 2016.

[11] Mark Cutler and Jonathan P How. Efficient reinforcement learning for robots using informative simulated priors. In *ICRA*, 2015.

[12] Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement learning with multi-fidelity simulators. In *ICRA*, 2014.

[13] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[14] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.

[15] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017.

[16] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *International Journal of Robotics Research*, 2012.

[17] Judy Hoffman, Sergio Guadarrama, Eric S Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko. Lsda: Large scale detection through adaptation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014.

[18] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *CVPR*, 2017.

[19] Dong Ki Kim and Tsuhan Chen. Deep neural network for real-time autonomous indoor navigation. *arXiv preprint arXiv:1511.04668*, 2015.

[20] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality ACM International Symposium on*. IEEE, 2007.

[21] John Kua, Nicholas Corso, and Avideh Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. In *IS&T/SPIE Electronic Imaging*, 2012.

[22] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[23] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015.

[24] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[25] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*. ACM, 2005.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[27] K. Mohta, V. Kumar, and K. Daniilidis. Vision based control of a quadrotor for perching on planes and lines. In *ICRA*, 2014.

[28] Mani Monajjemi. Bebop autonomy. http://bebop-autonomy.readthedocs.io.

[29] Dean A Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989.

[30] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In *ICRA*, 2015.

[31] Martin L Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 1978.

[32] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *arXiv preprint arXiv:1608.02192*, 2016.

[33] Martin Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*, 2005.

[34] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *ICRA*. IEEE, 2013.

[35] Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[36] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *NIPS*, 2005.

[37] Korbinian Schmid, Teodor Tomic, Felix Ruess, Heiko Hirschmller, and Michael Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *IROS*, 2013.

[38] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In *ICRA*, 2013.

[39] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015.

[42] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

[43] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *arXiv preprint arXiv:1612.05533*, 2016.

[44] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 2012.