Collision Avoidance via Deep RL: Real Vision-Based Flight without a Single Real Image

Fereshteh Sadeghi
University of Washington
Seattle, WA
fsadeghi@cs.washington.edu

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

Abstract

In this paper, we propose a learning-based method for indoor hallway following and obstacle avoidance for mobile robots using monocular images. In contrast to most indoor navigation techniques that aim to directly reconstruct the 3D geometry of the environment, our approach directly predicts the probability of collision given the current image and a candidate action. To obtain accurate predictions, we develop a deep reinforcement learning algorithm for learning indoor navigation, which uses the actual performance of the current policy to construct accurate supervision. The collision prediction model is represented by a deep convolutional neural network that directly processes raw image inputs. In order to address the high sample complexity of deep reinforcement learning and the dangers of trial-and-error learning in the real world, we present an approach for training our collision avoidance system entirely in simulation. By randomizing the rendering settings for our simulated training set, we show that we can train a collision predictor that successfully generalizes to a much more realistic synthetic benchmark environment, and provides reasonable predictions on real-world images. Our comparative evaluation shows that our approach outperforms a number of baselines, as well as a previous learning-based end-to-end navigation method.

1 Introduction

Indoor navigation is one of the basic requirements for robotic systems that must operate in unstructured open-world environments, including quadrotors, mobile manipulators, etc. Many of the most successful approaches to indoor navigation are based on 3D perception [1] and different types of sensors [8][7][3] which imposes additional costs on a robotic platform. In this paper, we explore a learning-based approach for indoor navigation, which directly predicts the safety of candidate 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 [2], our method uses reinforcement learning to obtain supervision that accurately reflects the actual probabilities of collision, instead of separating out obstacle detection and control. Using reinforcement learning to learn collision avoidance and indoor navigation presents a number of severe challenges. First, reinforcement learning tends to be data-intensive, making it difficult to use with platforms such as aerial vehicles. Second, reinforcement learning relies on trial-and-error, which means that the vehicle must experience at least a limited number of collision during training which is extremely problematic for fragile robots such as quadrotors. We present an approach for training our collision prediction network with reinforcement learning on simulated images. In order to ensure that the resulting network can transfer from simulation into the real-world, we describe a randomized rendering scheme that forces the network to handle a variety of obstacle appearances and lighting conditions. We demonstrate that our model trained entirely in simulation can generalize to much more realistic rendering of hallways, as well as real indoor flight.

2 Indoor Collision Avoidance with Deep Reinforcement Learning

The input to our model consists only of monocular RGB images, without depth, IMU inputs. Formally, let $\mathbf{I}_t$ denote the camera observation at time t, and let a_t denote the action. The goal of the model is to predict $P(C|\mathbf{I}_t, a_t)$, where C is the discounted expectation of a collision event:

$$P(C|\mathbf{I}_t, a_t) = \sum_{s=t}^{t+H} \gamma^s P(c_s|\mathbf{I}_t, a_t) \tag{1}$$

where $\gamma \in (0, 1)$ is the discount, c_s is an indicator for a collision at time s, and future actions are assumed to be chosen by the current policy. The horizon H should ideally be ∞, but in practice is chosen such that γ^H is small. Collisions are assumed to end the episode, and therefore can occur only once, ensuring that the sum is always in the range $(0, 1]$. Our model for $P(C|\mathbf{I}_t, a_t)$ 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 a_t by minimizing $P(C|\mathbf{I}_t, a_t)$. 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.

2.1 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 $P(C|\mathbf{I}_t, a_t)$, rather than $P(l|\mathbf{I}_t, a_t)$. We refer will refer to this method as Collision Avoidance via Deep Reinforcement Learning (CADRL). 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 determines the space of actions. Based on our score map, we consider a total of M^2 actions $\mathcal{A} = \{a_1, ..., a_{M^2}\}$ that can be taken after perceiving each observation $\mathbf{I}$. We start each episode by placing the agent at a random location and with random orientation and generate a rollout of size K, given by $(\mathbf{I}_0, a_0, \mathbf{I}_1, ..., a_{K-1}, \mathbf{I}_K)$. The actions a_t are selected according to the policy π after each observation $\mathbf{I}_t$. For each observation $\mathbf{I}_t$ encountered along the episode, we perform $M \times M$ additional rollouts using the policy π for every possible action a_t that can be taken at that time step, and evaluate the return of a_t according to Equation (1). Note that this corresponds to policy evaluation with the cost function $c(\mathbf{I}_t, a_t) = P(c_t|\mathbf{I}_t, a_t)$. 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 $P(C|\mathbf{I}_t, a_t)$. If the there is no collision within H steps, we use $P(c_{t+H}|\mathbf{I}_t, a_t) = c/d_{t+H}$, where d_{t+H} is the distance to the nearest obstacle at time $t + H$, and c is the radius of the vehicle, such that $c/d_{t+H} = 1$ when the vehicle is in collision. This policy evaluation phase provides us with a dataset of observation, action, and return tuples $(\mathbf{I}_t, a_t, P(C|\mathbf{I}_t, a_t))$, 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 labels in the range $[0, 1]$, similarly to the datasets during pretraining, but the labels now correspond to the probability of collision for the current policy π, rather than free space indicators.

3 Learning from Simulation

We manually designed a collection of 3D indoor environments to form the basis of our simulated training setup shown in Figure 1. Our simulated environment provides 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. The walls are textured with randomly chosen textures, chosen from a pool of 200 possible textures, and illuminated with lights that are placed and oriented at random. The randomization of the hallway parameters produces a very large diversity of training scenes, which can be seen in Figure 1. 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 are aligned with the results obtained in other recent works [6], which also used only synthetic renderings to train visual models, but did not explicitly consider wide-ranging randomization of the training scenes.

4 Experimental Results

We evaluate the performance of our trained policy in terms of the duration of collision free flight. We run continuous episodes that terminate upon experiencing a collision, and count how many steps are taken before a collision takes place and set the maximum number of steps to a fixed number throughout each experiment. An episode can begin in any location in the choice environment, but the choice of the initial position can have a high impact on the performance of the controller, since some parts of the hallway, such as dead ends, sharp turns, or doorways, can be substantially more difficult. To make an unbiased evaluation we start each flight from a location chosen uniformly at random within the free space of each environment. 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. We compare the performance of our method with previous methods for learning-based visual obstacle avoidance.

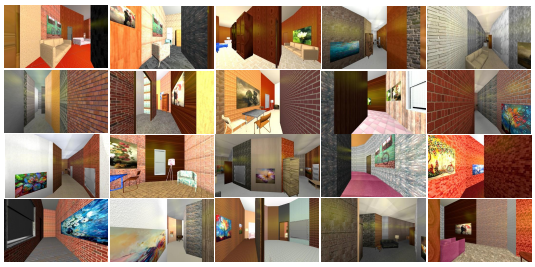

Figure 1: Visualization of our training environments. The rendering uses randomized textures, lighting conditions, and furniture placement to create a diverse range of visual scenes

Left, Right, and Straight (LRS) Controller: This method, based on [4], 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 camera images. Prior work trained such a model using real-world images collected from three cameras carried manually through forest trails (one camera was pointed left, one right, and one straight, and left camera images were supervised as right turns, right images as left turns, and straight camera images as straight motion). This method can be considered a human-supervised alternative to our autonomous collision avoidance policy, and we refer to it as "LRS."

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.

4.1 Realistic Environment Evaluation

We trained our policy on a set of synthetic 3D models of hallways and in a fully simulated environment with no real images. In order to evaluate how well such a model might transfer to a realistic environment, we used a realistic 3D mesh provided by [5]. In [5] and textured with natural images. As the result, evaluating on this data can provide us a close proxy of our performance in a real indoor environment. This experiment also evaluates the generalization capability of our method in a systematically different environment than our training environments. Figure 3 shows the floorplan of this hallway, as well as several samples of its interior view.

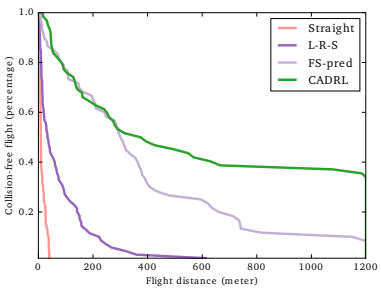

Figure 2: Quantitative results on a realistically textured hallway.

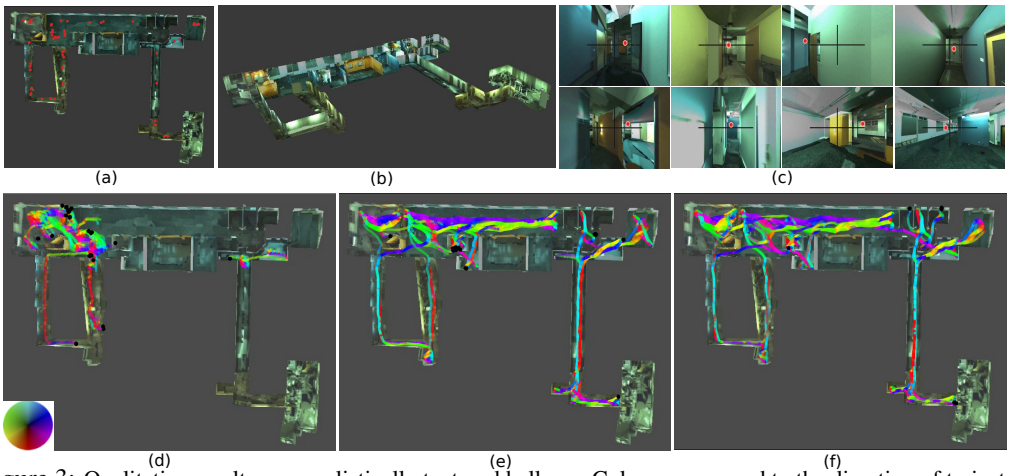

Figure 3: Qualitative results on a realistically textured hallway. Colors correspond to the direction of trajectory movement at each point in the hallway as per the color wheel. (a) Red dots show flight initialization points (b) Overlook view of the hallway (c) Red dots show the control points produced by CADRL. (d) LRS trajectories (e) Perception controller (FS-pred) trajectories (f) CADRL trajectories

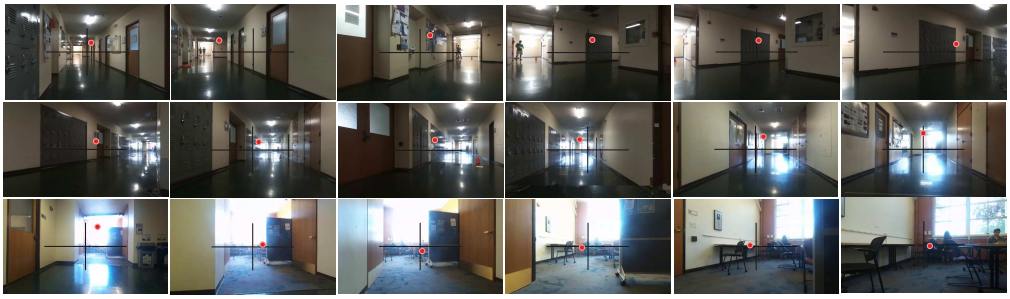

Figure 4: Snapshots from the real-world flight. Red dots show the heading direction selected by CADRL. The black cross shows the image center.

Figure 2 summarizes the performance of our proposed
CADRL method compared with other baselines. Our method outperforms the prior methods and baselines in this experiment by a substantial margin. CADRL 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.

4.2 Real-world flight experiment

We evaluated our learned collision avoidance model in the real world hallways where our goal is going through the hallways, take the turns, avoid drifting as well as avoiding possible collisions. We performed the real flight in the Cory Hall on the UC Berkeley campus. This experiment is aimed to evaluate how well the proposed approach can control a real autonomous aerial vehicle in the indoor environments. For running our experiments we used Parrot Bebop drone controlled via the ROS bebop autonomy package. Figure4 shows snapshots of the images captured by the flying drone and the red dots show the action computed by our policy as control points to navigate the drone. In this experiment, our controller successfully navigated the drone throughout the hallways without any collisions. Due to hardware bias and various conditions, the drone may sometimes drift to the left or right however, our controller can recover from this situations by stabilizing the drone in each step based on the current visual observation. This suggests that while our model is fully trained in simulation without seeing any real image or using any human provided demonstration, it has the capability to generalize well to real world images and conditions.

References

[1] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *EMAV*, 2009.

[2] C. Bills, J. Chen, and A. Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *ICRA*, 2011.

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

[4] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.

[5] J. Kua, N. Corso, and A. Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. In *IS&T/SPIE Electronic Imaging*, 2012.

[6] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *arXiv preprint arXiv:1608.02192*, 2016.

[7] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *IROS*, 2013.

[8] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 2012.