Deep Flight: Autonomous Quadrotor Navigation with Deep Reinforcement Learning

Ratnesh Madaan*, Dhruv Mauria Saxena*, Rogerio Bonatti, Shohin Mukherjee, Sebastian Scherer
The Robotics Institute
Carnegie Mellon University, Pittsburgh, PA 15213
Email: {ratneshm, dsaxena, rbonatti, shohinm, basti}@andrew.cmu.edu
*Equal contribution

***Abstract*—Obstacle avoidance for micro aerial vehicles is an essential capability for autonomous flight in cluttered environments but is often challenging because limited payload capacity only allows the use of simple sensors like monocular cameras. Extracting visual cues from monocular images to infer depth and distance to obstacles for a reactive planner is particularly difficult in low-textured environments. We aim to leverage recent advances in deep reinforcement learning to integrate obstacle perception and planning for collision-free flight with unmanned aerial vehicles (UAVs). We train a deep Q-network to learn a mapping from monocular images to yaw velocity commands for a UAV in simulation. By training the network over 6,200 self-supervised episodes, we achieve an 180% increase in collision-free flight time in a randomized forest of cylinders as compared to a naive random policy. A video of our results is available at https://goo.gl/w5yDTX.**

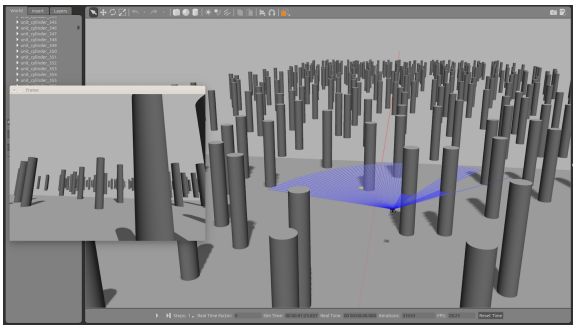

Fig. 1. Our simulation environment in Gazebo. Inset shows robot-centric monocular image. A sequence of four previous frontal images are fed to the DQN at each time step to make a decision. The laser scanner is only used to stop before the quadrotor crashes. We also give negative rewards if the robot comes too close to an obstacle, thereby emulating a local vector field.

I. Introduction

Despite numerous advancements in sensor technologies and motion planning algorithms, autonomous navigation of micro air vehicles is still a challenging problem due to payload weight, size and power constraints [1, 2]. Generally, Micro UAVs can only carry simple sensors like monocular cameras. Navigation with monocular cameras can be tackled by leveraging various cues like optical flow, relative size changes, and depth, each having its own advantages and points of failure. Optical flow uses motion parallax to detect changes in the image, and is of limited use in frontal camera applications because of the small angles relative to the frontal direction and expensive computation [1]. Mori and Scherer [1] explored relative size change by comparing SURF features of frontal images, but their approach requires high-textured environments. Saxena et al. [3] worked on depth recovery from single images using supervised learning, and utilize the depth map to train a policy for ground navigation in simulation and real environments [4]. Saxena does not provide details on failure cases, but we argue that the method is highly dependent on texture-rich environments.

In contrast with approaches that rely on domain-specific visual cues for input to a control system, recent advancements in deep reinforcement learning [5] inspired end-to-end learning of UAV navigation, mapping directly from monocular images to actions. Sadeghi and Levine [6] use a modified fitted Q-iteration to train a policy only in simulation using deep reinforcement learning and apply it to a real robot, using a single monocular image to predict probability of collision and selecting a collision-free direction to fly towards. Gandhi et al. [7] collect a dataset consisting of positive (obstacle-free flight) and negative (collisions) examples, and train a binary convolutional network classifier which predicts if the drone should fly in a direction or not. Both these methods however, fail to take into account three important factors: (1) leveraging a sequence of images, which can implicitly take care of objectives like time to collision. (2) low textured environments, and (3) reaching a specific goal position.

In this work we intend to address the aforementioned issues by using deep reinforcement learning in simulation for mapping sensor input from UAV-centric monocular images directly to yaw commands. Deep Q-Networks (DQNs) and their variants have been shown impressive results on fully observable 2D Markov Decision Processes(MDPs) [5], however attempts to apply them to navigation tasks in partially observable 3D scenarios are scarce, barring playing the game Doom [8, 9]. We investigate the effectiveness of DQNs to learn a policy from a stack of sequential images over a discrete set of yaw commands in randomized forests of low-textured cylinders in a Gazebo simulation environment [10] while flying at a constant speed and altitude to reach a goal point. We evaluate the performance of our network in terms of episode length and cumulative reward per episode.

II. Approach

A. Simulation Environment

We use a simulated model of a quadcopter in Gazebo which is equipped with a camera and a 2D laser sensor. Our environment is a forest of randomly placed cylinders, whose configuration changes at the end of each episode. The simulation environment is depicted in Figure 1. We define a fixed starting point for each episode at the midpoint of one edge of the forest and a goal point located at the midpoint of the opposite edge. The minimum reading of the laser scan is used as a measure of distance from obstacles d_{obs}, and is also used to decide the end of an episode.

We model the navigation task as a deterministic MDP, which can be defined by the state-action-reward-discount tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma\}$, where:

$\mathcal{S}$: is a sequence of 4 consecutive camera images, each 84*84 in size.

$\mathcal{A}$: is a set of 9 uniformly separated yaw velocity commands in the range of [-0.7, 0.7] rad/s (4 anticlockwise, 4 clockwise and 1 corresponding to zero velocity).

γ : 0.99

We tailor our reward function $\mathcal{R}$ to capture four criteria:

- Avoid crashing into obstacles: We give the agent $r_{safe} = 0.25$ for every time step when $d_{obs} > d_{neg} = 2.0$. Further, a high negative crash reward, $r_{crash} = -10.0$ is given if $d_{obs} < d_{crash} = 0.75$.
- Maximize minimum distance from nearest obstacle: A linearly increasing in magnitude negative reward is given from $r_{neg} = 0.0$ to $r_{precrash} = -5.0$ when $d_{neg} < d_{obs} < d_{crash}$.
- Avoid exiting the obstacle field: A crash reward r_{crash} is given if UAV leaves the forest of cylinders
- Reach the goal point: A positive reward proportional to the distance from goal(d_{goal}) at the current time step is given : $r_{goal} = (1 - \frac{d_{goal}}{d_{max}})$, where d_{max} is the distance to goal at the beginning of the episode, or simply the length of the forest environment.

B. Implementation Details

We use the architecture and training parameters proposed by Mnih et al. [5], with the input being 4 images of size 84x84. The first layer consists of 32 (8x8) convolutional filters, followed by 64 (4x4) filter, then 64 (3x3) filters, then a fully connected layer with 512 units and a final layer which outputs 9 possible yaw actions. Rectifier nonlinearities are applied after each layer. We implement our pipeline in Keras and Tensorflow and use a NVidia GTX 980M GPU for training the network.

III. Preliminary Results

During training we ran a total of about 6,200 episodes, or an equivalent of 1.4 million control steps in the environment. As seen in Figures 2-3, we observe significant increase in the reward per episode after roughly 800,000 steps. Figure 4 depicts an increase of 180% in flight time during testing as compared to the initial random policy.

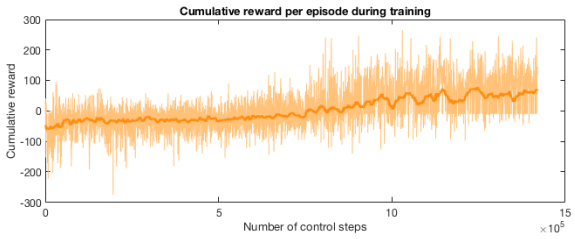

Fig. 2. Train reward with respect to number of control steps.

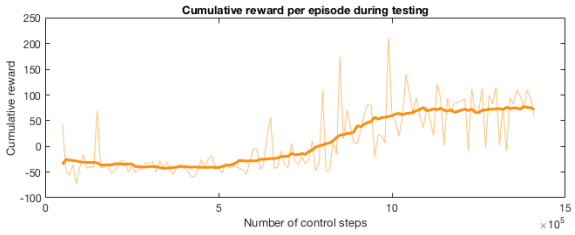

Fig. 3. Test reward with respect to number of control steps.

Fig. 4. Test episode length with respect to number of control steps.

Another interesting qualitative result is that the UAV tends to stay inside the obstacle forest, veering towards the middle of the environment if one of the sides of the image is free from clutter. We believe this preference of uniform clutter in images is linked to our goal driven reward r_{goal} and also due to negative rewards given if the agent goes out of the forest. A video of our learned policy can be found here and our code for training the network, as well as an OpenAI Gym environment [11] using ROS and Gazebo is available here.

IV. Conclusion and Ongoing Work

We show that deep Q-learning has the potential to be used for end-to-end training of obstacle avoidance policies for UAVs, mapping directly from a stack of monocular images to controls in a low-textured environment while flying at a constant speed and reaching a goal point. In addition, we released the code for our simulator to facilitate rapid prototyping of deep RL algorithms for flying robots.

We are working towards deploying our network on a Nvidia Jetson TX-2, on board a DJI Matrice 100. To achieve the same, we are training the DQN on disparity images to facilitate transfer learning. We're also collecting demonstration data to initialize our policy with, and potentially use demonstrations in the loop with RL [12].

Acknowledgments: The authors thank Weikun Zhen for the baseline UAV simulator.

References

[1] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1750–1757. IEEE, 2013.

[2] Sai Prashanth Soundararaj, Arvind K Sujeeth, and Ashutosh Saxena. Autonomous indoor helicopter flight using a single onboard camera. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5307–5314. IEEE, 2009.

[3] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[4] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600. ACM, 2005.

[5] 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*, 518(7540):529–533, 2015.

[6] Fereshteh Sadeghi and Sergey Levine. (CAD)^{2}RL: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[7] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. *arXiv preprint arXiv:1704.05588*, 2017.

[8] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. *arXiv preprint arXiv:1609.05521*, 2016.

[9] Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, Nantas Nardelli, N Siddharth, and Philip HS Torr. Playing Doom with SLAM-Augmented Deep Reinforcement Learning. *arXiv preprint arXiv:1612.00380*, 2016.

[10] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, et al. Learning from Demonstrations for Real World Reinforcement Learning. *arXiv preprint arXiv:1704.03732*, 2017.