

Autonomus Navigation and Collision Avoidance for Quadrotors

Mason Hawver
University of Minnesota
Minneapolis, MN 55455
hawve005@umn.org

Marcus Berg
ber00256@umn.edu

Vaagishwaran Sivakumar
sivak049@umn.edu

1. Introduction

Quadrotors, with their agility, maneuverability, and adaptability, excel in diverse environments, making them ideal for tasks that demand precision, flexibility, and quick responses. Thanks to this, they have become widely utilized in fields such as search and rescue, surveillance, agriculture, package delivery, entertainment, and defense; their impressive performance requires expert human pilots for operation, particularly in high-speed maneuvers within cluttered environments like cities or forests. However, this reliance on human pilots poses limitations on scalability, efficiency, and safety, hindering the full realization of quadrotors' potential applications. Introducing autonomy to quadrotors could revolutionize their utility and unlock opportunities for new industries. Yet, the challenge remains in developing autonomous quadrotors capable of navigating complex environments with both maneuverability and safety.

The challenge of developing autonomous quadrotors capable of navigating complex and unseen environments with both maneuverability and safety lies in bridging the gap between fast perception and effective planning. Addressing these challenges requires robust algorithms capable of handling sensor noise and planning strategies to ensure efficient navigation while maintaining safety under uncertainty.

Various approaches have been taken to address this issue. Some concentrate solely on perception [6, 10, 21, 29] employing robust online mapping and localization, and plan within the maps they generate. While others disregard perception, by modeling the environment to have perfect knowledge, and concentrate solely on planning [1–3, 14, 15, 19]. However, overlooking either aspect results in limitations. Effective navigation requires addressing both perception and planning.

One promising approach to bridge the gap between perception and planning in autonomous quadrotors is by training visual policies. These policies offer end-to-end solutions by directly planning through the visual input from the drone. Recent works [18, 25, 27] have shown promising advancements in this area, demonstrating the efficacy of visual policies in enhancing navigation capabilities in complex en-

vironments. By leveraging data collected by human pilots, these works have successfully trained visual policies to generate local trajectories that imitate human pilots. With deep learning techniques, visual policies can learn to map visual inputs to local trajectories directly, enabling quadrotors to navigate in complex environments without relying on accurate mapping and localization or prior knowledge. And Unlike past approaches, there is no explicit mapping and planning stages where errors can compound over time. Additionally, visual policies can mitigate the effects of sensor noise and uncertainties by learning robust representations from raw visual input, thereby improving overall navigation performance and safety. Through the integration of visual policies, autonomous quadrotors can achieve greater maneuverability and safety in complex and unseen environments, unlocking their full potential for various applications.

2. Related Work

In the pursuit of autonomous drone flight, various methodologies have been explored to address the challenges of perception, mapping, planning, and control. These approaches aim to enable drones to navigate complex environments with both agility and safety, laying the foundation for diverse applications.

2.1. Online Mapping

An established paradigm in autonomous drone flight involves mapping the local environment using sensor data, typically facilitated by Simultaneous Localization and Mapping (SLAM) techniques. However, traditional SLAM methods can impose significant computational burdens, potentially limiting their real-time applicability. To mitigate these challenges, recent advancements have focused on leveraging visual SLAM (V-SLAM) techniques. For instance, works [6] utilize Monocular V-SLAM to generate a sparse map, reducing computational requirements. Similarly, works [10] address scale learning challenges in Monocular V-SLAM by integrating camera and IMU data. Furthermore, works [29] employ Monocular V-SLAM for

emergency landing tasks in cluttered environments, employing techniques like grid mapping and image segmentation to enhance robustness. Despite the progress in V-SLAM, its computational demands during high-speed maneuvers remain a concern.

2.2. Reinforcement Learning

Recent endeavors have explored deep learning-based alternatives that collapse the mapping and planning tasks, offering a direct path from sensor input to flight control. Reinforcement learning (RL) has emerged as a powerful paradigm in this domain. For example, works [1] train a modified AlexNet using reinforcement learning, tailored for real-time constraints. Similarly, works [2] utilize reinforcement learning (RL) and Proximal Policy Optimization (PPO) for autonomous air-traffic control, demonstrating the adaptability of RL techniques in diverse applications. Additionally, works [15] extend Deep Q-Networks for drone control, addressing issues like overestimation with techniques such as Double Q-Learning[12] and Dueling Architecture. Moreover, works [14] employ Proximal Policy Optimization with Long Short-Term Memory [11] networks for drone control, further advancing the capabilities of deep RL in this domain. These advancements in reinforcement learning techniques offer promising avenues for enhancing autonomous drone flight capabilities, particularly in complex and dynamic environments.

2.3. Recent Advancements

Beyond RL-based methodologies, other deep learning approaches have been instrumental in enhancing autonomous drone flight capabilities. For instance, works [18] utilize sensor data to generate trajectories through imitation learning, showcasing the potential of this approach in learning from human demonstration. Moreover, works [9] curate a drone crash database and train a neural network for obstacle avoidance, addressing safety concerns in complex environments. Furthermore, works [16, 22, 30] optimize convolutional neural network (CNN) architectures for deployment in nano-drones, highlighting the importance of efficiency in resource-constrained platforms. Additionally, works [31] developed efficient motion planning techniques by using kinodynamic path-searching and B-spline optimization, offering robustness in trajectory generation. Finally, works [5] propose a navigation guidance system based on Proportional Navigation Guidance for obstacle avoidance without the need for mode switching, streamlining the control process.

A promising avenue in autonomous drone flight is the adoption of end-to-end policies, which directly translate visual inputs into control commands, bypassing intermediate mapping and planning stages. These policies offer streamlined solutions for navigation in complex environments,

capitalizing on deep learning techniques. Through the integration of visual policies, drones can achieve greater maneuverability and safety, unlocking their full potential for various applications. Recent works in this area [18] have demonstrated significant advancements, showcasing the efficacy of visual policies in enhancing navigation capabilities and overcoming challenges associated with perception and planning.

3. Baseline Method

Our approach to developing a visual policy for autonomous quadrotor flight primarily follows the methodology outlined in "Learning High-Speed Flight in the Wild" [18]. This involves initial training in simulation, followed by refinement using real flight data. Simulation provides scalability and comprehensive environmental information, enabling the creation of an expert policy. The expert policy, developed in simulation, serves as a blueprint for training a student policy through imitation learning. Finally, the student policy undergoes fine-tuning on real data before deployment on a quadrotor. The methodology and equations discussed in Section 3.1, 3.2, and 3.3 are taken from the paper [18]. However, as the equations are not standardized, it is essential to introduce them for the reader to understand the relationship between cost, path deviation, and collision.

3.1. Expert Policy

The expert policy utilizes privileged information from the environment and operates without time constraints to map the quadrotor's state, including its visual inputs, to optimal local trajectories toward a goal. This is achieved through global path planning algorithms [17] to find an initial reference trajectory. However, a single trajectory does not capture the full distribution of possible trajectories. For example, if an obstacle is directly in front of the quadrotor it is possible to dodge it to the left and to the right. To address this, a sampling algorithm explores branching trajectories at each time step.

The expert policy implements a sampling-based motion planning algorithm with perfect knowledge of the platform state and environment, available only in simulation. With this it generates a set of collision-free trajectories at each time step representing the desired quadrotor state over the next five seconds, starting from the current state. Trajectories are sampled from a probability distribution conditioned on a reference trajectory and the environment's structure. The distribution favors trajectories that avoid obstacles but still follow the reference trajectory. This distribution is complex and frequently multi-modal in cluttered environments, making analytical computation intractable. Therefore, random sampling, specifically the Metropolis-Hastings (M-H) algorithm [24], is employed to approximate the density function.

The probability distribution governing the trajectory sampling process is defined by the equation:

$$P(\tau|\tau_{\text{ref}}, \mathcal{C}) = \frac{1}{Z} \exp(-c(\tau, \tau_{\text{ref}}, \mathcal{C})) \quad (1)$$

Here, τ represents trajectories, τ_{ref} denotes the reference trajectory, and $\mathcal{C}$ signifies environmental constraints. The cost function $c(\tau, \tau_{\text{ref}}, \mathcal{C})$ incorporates collision avoidance and state optimization objectives, ensuring trajectories are both safe and efficient. In this formulation, the collision cost $C_{\text{collision}}(\tau(t))$ is a critical component, defined to balance proximity to obstacles and adherence to the reference trajectory. It accounts for the distance to the closest obstacle, balancing safety considerations with optimization objectives. The trajectory cost function is expressed as:

$$c(\tau, \tau_{\text{ref}}, \mathcal{C}) = \int_0^1 \lambda_c C_{\text{collision}}(\tau(t)) + \|\tau(t) - \tau_{\text{ref}}(t)\|^2 dt \quad (2)$$

Where λ_c is a weighting parameter used to balance the sampling process between safe and optimal trajectories.

The collision cost $C_{\text{collision}}(\tau(t))$ is delineated as:

$$C_{\text{collision}}(\tau(t)) = \begin{cases} 0, & \text{if } d_c > 2r_c \\ \frac{-d_c^2}{r_q^2} + 4, & \text{if } d_c \leq 2r_c \end{cases} \quad (3)$$

Here, d_c represents the distance to the closest obstacle, r_c signifies the safety radius, and r_q denotes the quadrotor's radius. This comprehensive formulation ensures that the trajectory sampling process is guided by both safety considerations and optimization objectives.

To decrease the sampling space dimensionality, trajectories are represented as cubic B-splines [8], facilitating efficient interpolation and ensuring dynamical feasibility. Three Control points of the B-spline are sampled in a spherical coordinate system, allowing for varied trajectory shapes. Despite this efficient representation, the privileged expert cannot operate in real-time due to computational overheads introduced by sampling. However, the robust trajectories generated by the expert policy can be used as data for a student policy to learn to mimic it via a more efficient deep neural network.

3.2. Student Policy

The student policy unlike the expert policy is subject to real time constraints. Additionally the student policy doesn't have access to privileged information; it only has access to onboard measurements. These measurements include the visual input from its forward facing cameras, a depth map estimated with the SGM [13], and the quadrotor's altitude, velocity, and desired heading towards a goal. The student policy should be able to use this data and produce similar

trajectories similar to the modes of the probability distribution the expert policy samples from. For a further visual understanding of this refer to Figure 3.

To process this data effectively, the student policy employs a neural network architecture. This architecture features layers designed to extract and synthesize relevant features from the depth images and state information. Specifically, it consists of a backbone comprising a MobileNet-V3 network [23] for depth image processing, followed by a four-layer perceptron for state feature extraction. These features are then concatenated and further processed by another four-layer perceptron to predict trajectories and collision costs. Specifically the network proposes three possible trajectories for a quadrotor to follow and their associated costs.

To train the network, they utilized the Mean Squared Error (MSE) loss against the ground truth costs and the predicted costs, as well as the MSE loss against the ground truth trajectories produced by the expert trajectory. However, it relaxes the MSE loss for expert trajectories that are not the closest one to the proposed trajectory hence the winner takes all equation.

The ground truth trajectories and the network predicted trajectories are represented as:

$$\tau_n^k = [p(t_i)]_{i=1}^{10}, \quad t_i = \frac{i}{10}, \quad (4)$$

The collision cost loss function compares the predicted cost c_k with the ground truth cost calculated from the point cloud environment using Equation 3. It is computed as:

$$\sum_{k=0}^{|\mathcal{T}_n|} (c_k - C_{\text{collision}}(\tau_n))^2 \quad (5)$$

And the loss function for the trajectory loss is the relaxed winner takes all [26]. The equation compares each expert trajectory to each trajectory predicted by the student policy. And to make a fair comparison to each trajectory the pairs of trajectories that are closest to each other are weighted more heavily with the $\alpha(\tau_{e,p}^i, \tau_n^k)$ function. The equations are represented as:

$$\text{R-WTA}(\mathcal{T}_e, \mathcal{T}_n) = \sum_{i=0}^{|\mathcal{T}_e|} \sum_{k=0}^{|\mathcal{T}_n|} \alpha(\tau_{e,p}^i, \tau_n^k) \|\tau_{e,p}^i - \tau_n^k\|^2, \quad (6)$$

$$\alpha(\tau_{e,p}^i, \tau_n^k) = \begin{cases} 1 - \epsilon & \text{if } \|\tau_{e,p}^i - \tau_n^k\|^2 \leq \|\tau_{e,p}^j - \tau_n^k\|^2 \quad \forall j \neq i \\ \frac{\epsilon}{M-1} & \text{otherwise.} \end{cases} \quad (7)$$

3.3. Baseline Validation

The baseline method validated their result in simulation across four different environments: forest, narrow gap, disaster, and urban environment. In the forest environment, trees were generated randomly to create varied landscapes with a predefined reference trajectory. For the narrow gap environment, the gaps were randomized laterally. The disaster and urban environment were static but represented more challenging environments. The authors compared their results with FastPlanner described in [31], Reactive Planner described in [7], and a blind approach to validate the complexity of the environment. They tested their model at velocities ranging from 1 m/s to 10 m/s, performing ten trials in each environment at each velocity. They outperformed the Fastplanner, the Reactive Planner, and the blind approach across all velocities, and environments. At low velocities until about 5 m/s, they had 100 percent collision-free trajectories. However, at higher speeds, the accuracy linearly dropped to approximately 60 percent at 10 m/s. To evaluate the effect of latency and sensor noise, the paper compared true depth with the estimated depth as input to the model, which showed that using the stereo depth compared to the ground truth depth had minimal impact on the result.

4. Proposal Method

The proposed method is to implement the data generation pipeline and the model developed in the paper Learning High-Speed Flight in the Wild [18] from scratch, but also explore explaining the methods developed in that paper. Specifically, the goal is to make the autonomous drone more robust and test the model in more environments.

The original plan was to introduce dynamic elements into scenes to assess their impact on the method’s performance. We also intended to investigate alternative applications for the heading value provided to the model, possibly directing it toward specific regions of interest within an image. Additionally, we planned to experiment with the model’s architecture, taking into account recent advancements in transformers and diffusion policies. Lastly, we planned to experiment with the generation of the data set to produce more robust expert trajectories to train the model.

The proposed extensions of the projects were not fully realized, due to challenges outlined in Section 6. We generated random scenes and implemented the privileged expert. However, due to time limitations and the sizable dataset requirement for neural network training, integration of the random scene generation, privileged expert, and training on our generated dataset were not attainable. Collision-free trajectories were achieved in simulation with true depth images by using the a reduced dataset provided by [18]. Subsequently, we validated the results using our dataset.

5. Result

Our group successfully implemented the expert policy and coupled it with scene generation to make a dataset generation pipeline for the student policy. However, due to the limitations outlined in Section 6 we were unable to successfully train a child policy with results similar to the one achieved in the paper Learning High-Speed Flight in the Wild [18].

5.1. Qualitative Result

The qualitative result for the data generation is discussed in Section 5.1.1, and the qualitative result for the student policy in Section 5.1.2.

5.1.1 Qualitative Result Expert Policy

For the dataset generator, our group created two scenes for our expert policy to plan within, one is a castle from Italy and the other is a randomly generated forest. The castle is illustrated in Figure 1 and the forest is illustrated in Figure 2. From these environments our expert policy planned a global reference trajectory and a physical drone rollout that includes all of the force and torque commands that are sent to the quadrotor to make it fly, this is visualized in the Videos 5.1.1 and 5.1.1. From the environments and the global reference trajectory, our sampling based motion planner, implemented with Metropolis Hastings, then finds better local trajectories at each time step for the global trajectory, this can be seen in Figure 3. With all of this we created a small dataset that could be validated with the child policy.

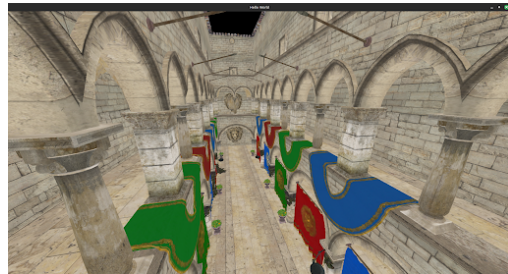


Figure 1. Colosseum environment

- Video 1: https://drive.google.com/file/d/1GZ0NekQChSPg_gIZ4HofZTUg1K24PZR4/view?resourcekey (castle environment)
- Video 2: <https://drive.google.com/file/d/1Z5eWpP0EvWNgXTh1-9oHaORpGk0mdlhy/view?resourcekey> (forest environment)

5.1.2 Qualitative Result Student Policy

The student policy was trained on 30,000 data points each data point included a depth image an IMU observation, and

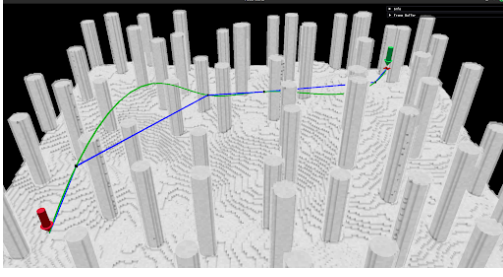


Figure 2. Generated forest environment, with collision free path generated by θ^* [4] (blue line), and min snap trajectory optimization [20] (green line)

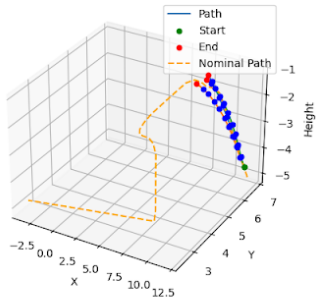


Figure 3. Lowest cost expert trajectories from MH sampling

an associated point cloud. The model hyperparameters were selected with a validation dataset of 3,000 data points, and the model was then tested on a dataset of another 3,000 data points. After several attempts to train the student, we failed to achieve similar results as the baseline for reasons outlined in the challenges Section 6. The network did not fully optimize the space loss, so the resulting trajectories do not imitate the shape and desired direction of the input trajectories. However, it did optimize for the collision loss. This is visualized in Figure 4 and Figure 5, which is also indicated in our quantitative results discussed in Section 5.2. As an important note, we used the dataset provided by the baseline method [18] although we created our own to help mitigate potential issues.

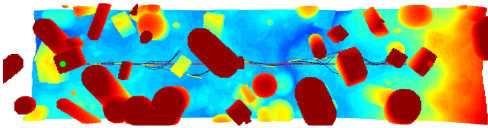


Figure 4. Trajectories generated by the expert policy

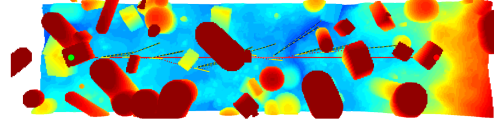


Figure 5. Trajectories generated by the student policy

5.2. Quantitative comparison

The baseline method was able to train on 90,000 data points, but we were only capable of training on a third of the dataset because of the limitations explained in Section 6. Because of this either our model overfit to the training data or the loss plateaued before it could be fully optimized. However, we were still able to calculate a rough collision percent for our student policy. Comparing it to the baseline method is not exactly fair because our model mainly optimizes for the collision loss, eq: 5, and plateaus before any meaningful change happens to the trajectory loss, eq: 6. Our group achieved a student policy that collided 58.62% of the time in the testing dataset, while the baseline method achieved a collision percent between 0% and 40%, as detailed in Section 3.3. However, for better comparison the model should have been evaluated in several environments, with different velocities and compared to the baseline results. And if the model had better results, then it should have been evaluated with our dataset.

6. Challenges

Our group ran into several challenges when training the student policy. After attempting to train the model several times, either the model over-fit to the training data or the model’s performance plateaued before it was capable of performing the task. To curb these results we added dropout layers [28], regularization, data augmentation, parameter freezing, and transfer learning yet we still were unable to train a model to achieve the same results as our baseline method. It became evident that we needed more data to overcome this issue. It was reaffirmed when we included data augmentations. However, the dataset already was consuming all of the system’s RAM and disk space on Google Colab, so increasing the dataset size would require better cloud computing resources than Google Colab Pro Plus. The loss plot can be seen in Figure 6 and it highlights what happened to our model after applying all the tricks to mitigate overfitting. The model quickly plateaus and underfits to the training data instead making it clear the need for more training data.

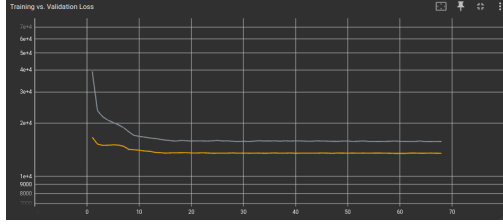


Figure 6. Training loss, grey line, vs Validation loss, orange line. Both underfit to the data as there is still plenty of room for the loss to be minimised.

7. Conclusion

In conclusion, the group trained a visual policy, based on [18], to find collision free trajectories that resulted in collisions 58.6% of the time as discussed in Section 5.2. This model is on the way to being useful but still has too low accuracy to be secure in real practical applications. An increased accuracy is believed to be achieved with more robust data generation and increased training data. In the future, we want to utilize the dataset we created to test a wider range of environments and produce more robust expert trajectories. This would expose the training to a more diverse set of data, which could reduce the risk of over-fitting the model. Furthermore, the network architecture could be updated to incorporate recent advancements in transformers and diffusion policies.

Although the proposed extensions of the projects were not fully realized, the work validates the approach used in the paper. Additionally, new training data was generated to further validate and improve the model. Due to the extent of not only training on a dataset but also creating the environment, planning global collision-free trajectories, and implementing Metro Hasting sampling, the group has recognized the significant effort required to produce datasets that effectively train the network. Another important lesson taken from the project is the limitations of hardware constraints. A significant amount of time was spent finding ways to train the network, highlighting the addressing of hardware constraints early in the project. The group has learned from the lessons and challenges the project has provided, but will make the group more capable of approaching challenging computer vision problems in the future.

References

- [1] Aqeel Anwar and Arijit Raychowdhury. Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. *IEEE Access*, 8: 26549–26560, 2019. 1, 2
- [2] Marc Brittain and Peng Wei. Autonomous air traffic controller: A deep multi-agent reinforcement learning approach. *ArXiv*, abs/1905.01303, 2019. 2
- [3] Marco F Cusumano-Towner, Alexey Radul, David Wingate, and Vikash K Mansinghka. Probabilistic programs for inferring the goals of autonomous agents. *arXiv preprint arXiv:1704.04977*, 2017. 1
- [4] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39:533–579, 2010. 5
- [5] Cristino de Souza, Pedro Castillo, and Boris Vidolov. Reactive drone pursuit and obstacle avoidance based in parallel navigation. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020. 2
- [6] Omid Esrafilian and Hamid D. Taghirad. Autonomous flight and obstacle avoidance of a quadrotor by monocular slam. *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, pages 240–245, 2016. 1
- [7] Pete Florence, John Carter, and Russ Tedrake. Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 304–319. Springer, 2020. 4
- [8] Jean H Gallier. *Curves and surfaces in geometric modeling: theory and algorithms*. Morgan Kaufmann, 2000. 3
- [9] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Kumar Gupta. Learning to fly by crashing. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955, 2017. 2
- [10] Sergio García, María Elena López Guillén, Rafael Barea, Luis Miguel Bergasa, Alejandro Gómez, and Eduardo J. Molinos. Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion. *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 205–210, 2016. 1
- [11] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012. 2
- [12] H. V. Hasselt. Double q-learning. In *Neural Information Processing Systems*, 2010. 2
- [13] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 3
- [14] Victoria J. Hodge, Richard Hawkins, and Robert Alexander. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, 33:2015 – 2033, 2020. 1, 2
- [15] Kjell Kersandt. Deep reinforcement learning as control method for autonomous uavs. 2018. 1, 2
- [16] Lorenzo Lamberti, Elia Cereda, Gabriele Abbate, Lorenzo Bellone, Victor Javier Kartsch Morinigo, Michal Barcis, Agata Barciś, Alessandro Giusti, Francesco Conti, and Daniele Palossi. A sim-to-real deep learning-based framework for autonomous nano-drone racing. *IEEE Robotics and Automation Letters*, 9:1899–1906, 2023. 2
- [17] Sikang Liu, Kartik Mohta, Nikolay Atanasov, and Vijay Kumar. Search-based motion planning for aggressive flight in se (3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446, 2018. 2

- [18] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59): eabg5810, 2021. 1, 2, 4, 5, 6
- [19] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. *Science robotics*, 8(84):eadf7843, 2023. 1
- [20] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011. 5
- [21] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Y. Siegwart, and Juan I. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2016. 1
- [22] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. Ultra low power deep-learning-powered autonomous nano drones. *ArXiv*, abs/1805.01831, 2018. 2
- [23] Siying Qian, Chenran Ning, and Yuepeng Hu. Mobilenetv3 for image classification. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 490–497, 2021. 3
- [24] Gareth O Roberts and Adrian FM Smith. Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stochastic processes and their applications*, 49(2):207–216, 1994. 2
- [25] Jeremy Roghair, Kyungtae Ko, Amir Ehsan Niaraki Asli, and Ali Jannesari. A vision based deep reinforcement learning algorithm for uav obstacle avoidance. *ArXiv*, abs/2103.06403, 2021. 1
- [26] C. Rupprecht, Iro Laina, Robert S. DiPietro, and Maximilian Baust. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3611–3620, 2016. 3
- [27] Wenxiang Shi, Chen Zhao, Kailei Tang, Junru Sheng, Zhiyan Dong, Lihua Zhang, Xiaoyang Kang, and Kai Cao. Flytransformer: A cross-modal fusion policy for uav end-to-end trajectory planning. *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3501–3506, 2023. 1
- [28] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014. 5
- [29] Tao Yang, Pei Li, Huiming Zhang, Jing Li, and Zhi Li. Monocular vision slam-based uav autonomous landing in emergencies and unknown environments. *Electronics*, 7:73, 2018. 1
- [30] Ning Zhang, Francesco Nex, George Vosselman, and Norman Kerle. End-to-end nano-drone obstacle avoidance for indoor exploration. *Drones*, 2024. 2
- [31] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019. 2, 4