Technische Universität Berlin

Fakultät IV - Elektrotechnik und Informatik

Fachgebiet AOT
Prof. Dr. Sahin Albayrak

Master's Thesis Proposal

Temporal comprehension in autonomous drone racing: using a recurrent convolutional neural network that makes navigational decisions based on raw sensor data.

Friedrich Clemens Valentin Mangelsdorf

Matrikel–Nummer 356686

Supervisor Dr. rer. nat. Yuan Xu

Table of Contents

1 Motivation

Advances in drone technology in recent years have opened the door to many application possibilities in the civil sector. [9] With regard to the commercial use of drones, in the near future great economic potential is expected within the fields of infrastructure, transport, insurance, media and entertainment, telecommunication, agriculture, safety and mining. [19] In contrast, more visioniary ideas such as drone delivery have not been broadly realized yet. [21] Across the globe, many big corporations and smaller start-ups are conducting research on drone delivery applications. [11] Until now, only several smaller test projects in sparsely populated areas have been realized. Decisive reasons for this are of a technical nature. In particular, autonomous navigation methods are not yet robust enough for the reliable deployment in densely populated urban areas. [18] This master's thesis is intended to make a contribution to autonomous navigation of drones with basic research on integrating temporal comprehension into the navigation method.

In science as well as in this thesis proposal, the colloquial term "drone" refers to the aircraft class of unmanned aerial vehicles (UAVs). The International Civil Aviation Organization (ICAO) [2] defines UAVs as aircrafts without a human pilot onboard, which are either remote-controlled by a human operator or "preprogrammed and fully autonomous". An UAV is a component of an unmanned aerial system (UAS), which additionally consist of a ground control station, communication link and payload. Basic components of UAVs are an airframe, an electric power system, a flight control system and an air data terminal. [10] For UAVs with autonomous functions, onboard computers provide the required additional computation power. Various systems exist to classify UAVs either by airframe type [5] or by flight key characterstics [24, 27]. With respect to the latter, Watts, Ambrosia and Hinkley [26] proposed a classification system for scientific usage. Table 6.1 shows a selection of the system's UAV sub-classes, i.e., micro air vehicles (MAV), low altitude, short endurance (LASE), low altitude, long endurance (LALE), medium altitude, long endurance (MALE) and high altitude, long endurance (HALE). This proposal can be primarly associated with quadcopters, the representative of MAVs that, according to my observations, prevails among hobbyists and commercial applications.

UAVs originate from the military, were adopted by enthusiasts, have gained more and more acceptance in the industry and finally have become today's standard in aerial inspection services in agriculture, construction, infrastructure, utilities, and mining. [8, 1, 4] With fast aerial maneuverability, UAVs exhibit several substantial advantages over ground vehicles. As they are unaffected by many obstacles and largely independent of infrastructure, destinations can be reached by the shortest route, waiting times can be avoided and the effort and risk of getting to places that are difficult to access can be reduced. [26] In addition, from the privileged perspective of a bird, onboard sensors are able to record extensive data of high quality fast at low cost. [19] This ability is extremly valuable in the context of big data, cloud computing and machine learning. [11] On the downside, MAVs are smaller aircraft vehicles that can move substantially less weight which, besides computational power and mission-specific payload, limits battery capacity, which in turn restricts flight range and duration. For MAVs paying off economically, efficiency is therefore paramount. A central approach to increase efficiency is to shift functions from a human operator to the drone itself, i.e., increase the drone's degree of autonomy. This becomes particularly clear in the example of delivery applications. Because drones can load significantly fewer packages and have to recharge more often than conventional delivery trucks have to refuel, one human pilot per drone would be too ineffective for broad deployment. Drone delivery systems are only profitable if most functions are performed autonomously by the individual delivery drone, or, at best, by the collective of the drone swarm. Consequently, expensive human labor could be reduced and vast room for mathematical optimization could be created. [7, 16]

So far, UAVs have proven their value in fields that are connected with large industrial sites which usually provide flight environments that are essentially demarcated and controlled and therefore very predictable. State-of-the-art autonomous drones are capable of dealing with these environments. Here, mention can be made of drone-in-a-box systems: the box provides takeoff/landing and charging infrastructure for the drone or drone swarm, that autonomously execute preprogrammed flight missions. [6] In comparison, the open-world environments of our daily lives are densely disturbed and are, thus, characterized by high uncertainty. Drones have not yet asserted themselves for commercial applications in these environments. This is particularly true for drones with autonomous functions. Crucial reasons for this are legal restrictions, lack of acceptance and technical complexity, notably with respect to "navigation, communication and automatization"[14]. [21]

Autonomous navigation is a highly relevant topic in current UAV-related public research. Navigation, which is a basic functionality of each UAV, comprises the main task of achieving a desired pose or position while performing necessary sub-tasks at the same time. [18] The necessity of individual sub-tasks depend on the flight environment. In densely populated open-world areas an autonomous navigation method should integrate functions to handle obstacle avoidance and the coordination with other agents, while in rural areas or on controlled industrial sites navigating through pre-planned waypoints only relying on global navigation satellite system (GNSS) sensors may be sufficient. Current autonomous navigation methods lack in robustness for complex environments and do not exhaust the full agility of MAVs.

2 Objectives

Public research has put forth many advanced methods for autonomous navigation of MAVs. However, they are not sophisticated enough to conquer open-world environments of high uncertainty such as urban areas. [18] In current research, deep learning techniques that empower MAVs with comprehension abilities constitute the best approach to face the uncertainty of these environments. State-of-the-art navigation methods integrate feedforward, deep convolutional neural networks that map the current color or depth image to action. Thereby a high, spatial comprehension of the immediate surrounding of the MAV can be achieved. But I suspect that the mere understanding of space, as good as it may be, is not enough for the autonomous navigation of MAVs in open-world environments. Therefore, I aim to develop a navigation method that besides spatial also includes temporal comprehension. The following objectives should be achieved within the scope of the master's thesis.

The method is intended to expand the work of Kaufmann et al. from ETH Zurich in 2018 [13]. They developed a vision-based navigation method that enables a drone to fly autonomously through a drone race track, thereby achieving high reliability with high speed and agility on dynamic flight curves. Their hybrid approach consists of a convolutional neural network (CNN) and a conventional control system that repeatedly executes the following steps at a high frequency: Inputting the current raw RGB image from the onboard, forward-facing camera, the CNN generates a waypoint in the 2D reference frame of the image ($x, y \in [-1, 1]$). The waypoint is transformed from the reference frame of the image to the reference frame of the drone state estimation ($x, y, z \in \mathbb{R}^3$). A trajectory from the current drone state estimate to the waypoint is computed by minimizing jerk, which is the time derivative of acceleration. The states of the trajectory are then tracked by a conventional control algorithm. The end state of the trajectory is never reached since the re-planning of the trajectory is continually triggered by new waypoints. In my thesis I plan to make progress on the first part of the hybrid approach by adding a recursive network to the CNN (CNN-LSTM). Furthermore, I would like to examine if feeding additional features (e.g., state estimates) to the network is beneficial. I expect that thereby, waypoints are not only generated on the basis of the current RGB image, but that also past images (and possibly also current and past state estimates of the drone) are included in the navigation decision. This would possibly result in the following positive effects:

- Due to its "memory", the network is able to generate meaningful waypoints in case that the next gate of the race track is not depicted in the current image, but has appeared in previous images.
- Due to temporally distributed images, the network is able to take the speeds of moving gates (or obstacles) into the account of the navigation decisions.
- Since trajectories are temporally extended maneuvers, a network with temporal comprehension is more able to imitate the expert system. Thus, the resulting trajectory through the race track formed by the successively generated waypoints is more similar to a precomputed optimal (in terms of acceleration, jerk, snap, ...) trajectory.

The approach is implemented utilizing the middleware ROS [23] and simulated with the photo-realistic Flightmare Simulator [22]. For the implementation concept, see section 6.4. Tests to compare my extended approach with the original method should be designed and conducted to investigate if the above enhancements could be realized.

My work is based on the paper of Kaufmann et al. [13] from 2018. Just recently, follow-up work from ETH Zurich [17] with a way more sophisticated autonomous navigation method has been published. The neural network is designed to cover a bigger scope by directly outputting trajectories instead of waypoints in image coordinates. However, I intend to still base my work on the earlier paper, since the more modular design provides a very good starting point with respect to my intended basic research whether temporal comprehension leads to improvements in autonomous navigation. Moreover, the test environment of drone racing is very convenient due to its simplicity. The completion of the race track can be achieved by mere reactive control to the next gate and does not require the formulation of a high level navigation goal whose implementation would pose another challenge.

3 Task Packages

Based on the objectives, the following task packages are defined:

- Build the drone racing simulation.
  - Familiarize with Flightmare [22], a photo-realistic simulator for quadcopters.
  - Familiarize with the Unity Engine [25], to create and manipulate the rendered environments in Flightmare.
  - Transfer the already implemented simulation from Gazebo [15] to Flightmare. For exemplary training data generated in Gazebo see section 6.3.
- Generate training data in simulation.
  - Adjust the already implemented expert system (see section 6.2), so that the training data includes an RGB image and a state estimate of the drone as features and a waypoint and desired speed as labels.
  - Run simulation to generate the data.
- Build the recurrent convolutional neural network (R-CNN).
  - Familiarize with PyTorch [20].
  - Transfer already implemented data input pipeline from TensorFlow [3] to PyTorch.
  - Familiarize with R-CNNs.
  - Design the R-CNN using hyperparameters for later adjustments.
  - Implement the R-CNN in PyTorch.
  - Train the R-CNN and find best values for the hyperparameters. Use mean squared error as metrics for regression.
- Evaluate my approach.

  - Design test scenarios (see section 6.5) for comparison with the base navigation method of Kaufmann et al. [13].
  - Implement and conduct tests in simulation.
  - Evaluate results of tests.

- Write thesis documentation.

- (Optional) Conduct test in real world.
  - Implement my approach on real drone.
  - Find test site and rebuild a test scenario that in simulation produced promising results.
  - Conduct test, evaluate and document in thesis.

4 Time Schedule

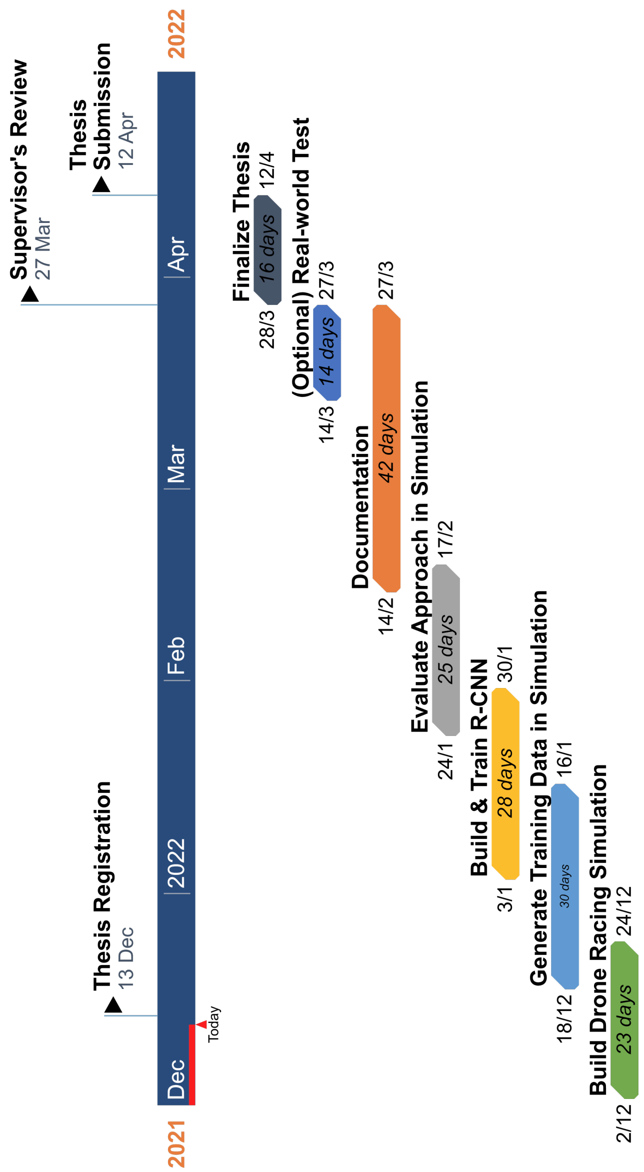


Figure 4.1: Gannt diagram illustrating the time schedule of the master's thesis. *Created with Office Timeline Online*

5 Organizational Matters

- Language of the thesis: English
- Text processing system: LaTeX
- Programming languages: C++, Python
- Supervisor: Dr. rer. nat. Yuan Xu
- Reviewers: Prof. Dr. Sahin Albayrak, Dr.- Ing. Stefan Fricke

6 Annex

6.1 Selected UAV Classes of the Classification System by Watts, Ambrosia and Hinkley

Table 6.1: Selected UAV Classes of the Classification System by Watts, Ambrosia and Hinkley. *Source: assembled from [26].*

Class	Altitude	Endurance	Range	Takeoff/Landing
MAV	$<$ 330 m	$<$ 30 min	$<$ 1 km	Any small area
LASE	$<$ 450 m	$<$ 2 h	$<$ 10 km	Human hand, catapult system or runway
LALE	$<$ 5,000 m	$<$ 20 h	$<$ 100 km	Runway
MALE	$<$ 9,000 m	$<$ 40 h	$<$ 1,000 km	Runway
HALE	$<$ 25,000 m	$<$ 30 h	$<$ 10,000 km	Runway

6.2 The Expert System

In machine learning, an expert system is a program that imitates a human expert in order to solve a problem. It comprises two main components: a *knowledge base* that stores known facts and rules, and an *inference engine* which, by applying the rules to the known facts, generates new facts. [12]

For the navigation method of my thesis, I have already re-implemented the expert system (in the paper, referred to as expert policy) by Kaufmann et al. [13] as a function within a ROS node. The following structure summarizes the expert system:

- Problem to solve

  - Replace the yet untrained CNN when navigating the drone through the drone race track by generating the inputs (i.e., a waypoint in image coordinates ($x, y \in [-1, 1]$), desired speed) to the conventional control system (see chapter 2).
  - While flying through the race track, generate training data (i.e., RGB images) with labels (i.e., a waypoint in image coordinates ($x, y \in [-1, 1]$), desired speed).

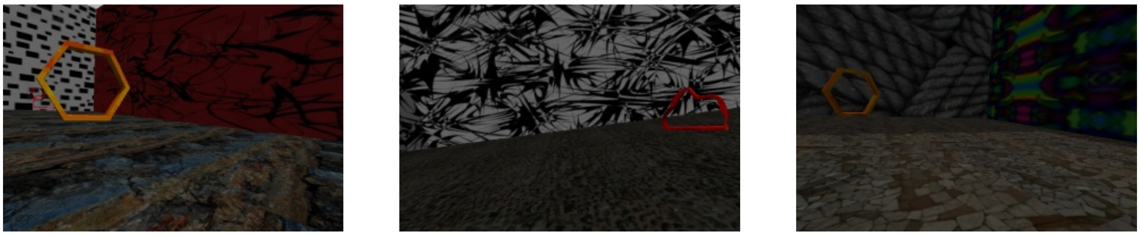
- Knowledge base
  - Known facts
    * Center points of the gates that build the drone race track.
    * A pre-computed, minimum-snap trajectory (referred to as global trajectory) traversing the center points of the gates.
    * Continually updated, ground-truth state of the drone.
    * Current gate (i.e., the gate that needs to be traversed next when completing the race track) and last gate (i.e., the gate that has been traversed most recently)
  - Rules
    * Project the current drone state to a state of the global trajectory and name this state expert state. For more details of the rather complex projection, see the original paper.
    * If the drone is more distant to the expert state than a user-specified parameter, set the expert state to the state of the global trajectory with minimum distance to the drone.
    * Set the desired speed (component of the label) to the velocity of the state that is ahead of the expert state with a user-specified distance. However, the desired speed is bounded by an user-specified minimum and maximum speed as well as an user-specified maximum speed increment with respect to the current speed of the drone.
    * Compute the horizon (a distance) as the product of a user-specified time duration and the desired speed. However, the horizon must not be greater than the distances to the current and last gate.

  * Set the waypoint in image coordinates, $x, y \in [-1, 1]$, (component of the label) by projecting the position of the state, which is ahead of the expert state with a distance equal to the horizon, onto the image. Do this in consideration of the current state of the drone.

- Inference engine
  - Implemented as a set of functions within a C++ ros node, that fetches the known facts via ROS topics or computes and store them as internal variables.
  - The user can de-/activate the expert system as functional part of the navigation method.

In my thesis, I will most likely adapt the expert system, so that the training data includes additional features. In addition to an RGB image, the features will probably include a state estimate of the drone.

6.3 Labeled training data generated in simulation with Gazebo



(a) x = -0.347238
y = 0.201331
v = 0.568805

(b) x = 0.365596
y = -0.043410
v = 0.845403

(c) x = -0.247982
y = 0.166272
v = 0.651439

Figure 6.1: Three samples of labeled training data, generated in randomized simulation with Gazebo. Features: 300x200 RGB image, Labels: x-,y-coordinate of waypoint in image reference frame, normalized desired speed.

6.4 Implementation Concept

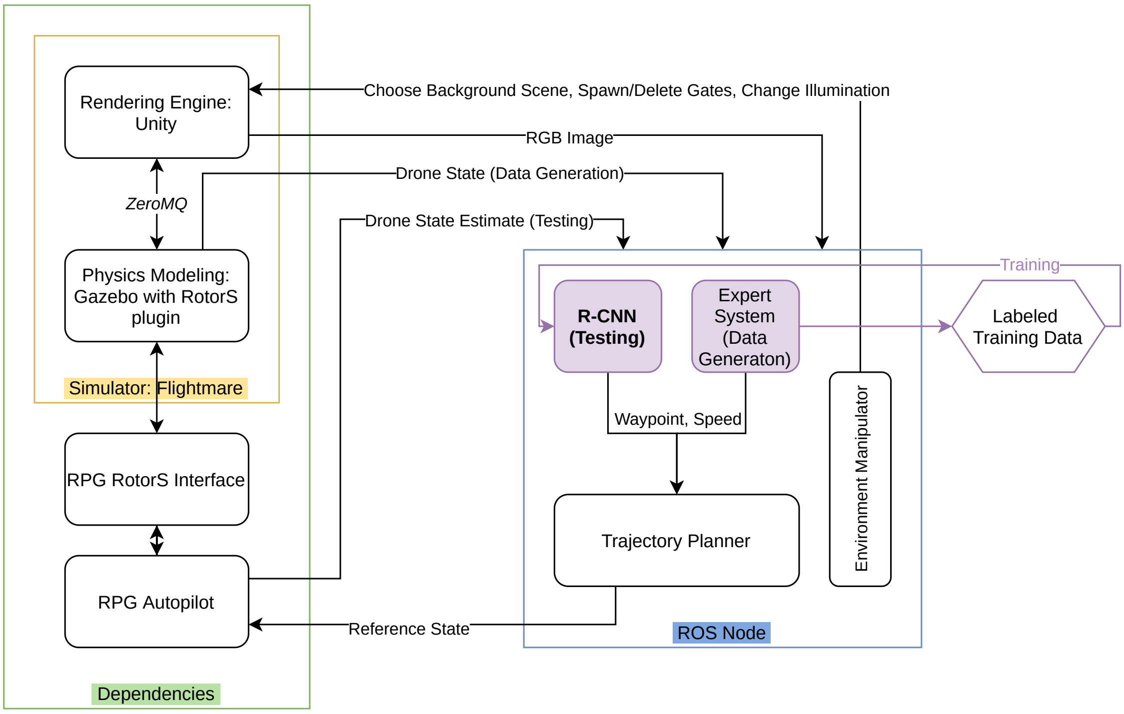


Figure 6.2: Implementation Concept: Besides the internal communication in Flightmare (yellow frame), all communication is carried out via ROS. The ROS node (blue frame) contains code from the base method that I have re-implemented and adjusted to my needs. The purple color marks the parts that will be fully my work. *Created with Diagrams.net.*

6.5 Test Scenarios

This sections introduces three possible scenarios for testing my navigation approach and comparing it with the base method.

Drone Racing Simulate a closed drone racing track. Navigate the drone through the race track using the base method and my approach. Metrics for evaluation would be at various maximum speeds: percentage share of successful runs, average time of completion and closeness to the global trajectory in terms of optimality. The latter could be calculated by applying a cost function on the resulting trajectories through the track.

Sharp curve Simulate two gates with a sharp curve in between. Before the drone passes the first gate, both successive gates must appear in the frame of view of the onboard camera. The curve must be so sharp, that, after traversing the first gate, not only the first but also the second gate has left the frame of view. The base method, whose CNN makes navigation decisions from only the current image, will likely to fail in this scenario due to the absence of a goal. In contrast, my approach uses an R-CNN which is able to internally store information from previous images. In case that the R-CNN is well trained, it should be able to generate meaningful waypoints to complete the sharp curve and traverse the second gate. The percentage share of successful runs would be a convenient metric for evaluation.

Moving gate Simulate a gate that moves along a linear path with a low, constant speed. Both, the base method and my approach, should be able to traverse the moving gate if its speed is not too high. But, the R-CNN of my approach has the potential to anticipate the motion of the gate from the temporally distributed RGB images. Thus, it may be able to reach the moving gate faster on a shorter route. The average time needed to traverse the gate could be used as metric for evaluation.

Bibliography

[1] Visual drone inspection across different industries. `https://www.equinoxsdrones.com/blog/visual-drone-inspection-across-different-industries`. Accessed: 2021-11-30. 2

[2] *Global Air Traffic Management Operational Concept (Doc 9854)*. International Civil Aviation Organization (ICAO), 2005. 1

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 6

[4] Dor Abuhasira. In 2022, percepto is bringing visual inspection automation to all. `https://www.equinoxsdrones.com/blog/visual-drone-inspection-across-different-industries`. Accessed: 2021-11-30. 2

[5] Reg Austin. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment (Aerospace Series Book 55)*. Wiley, 2011. 1

[6] Boaz Ben-Moshe. Power line charging mechanism for drones. *Drones*, 5(4), 2021. 2

[7] Wen-Chyuan Chiang, Yuyu Li, Jennifer Shang, and Timothy L. Urban. Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization. *Applied Energy*, 242:1164–1175, may 2019. 2

[8] Pamela Cohn, Alastair Green, Meredith Langstaff, and Melanie Roller. Commercial drones are here: The future of unmanned aerial systems. `https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems`. Accessed: 2021-11-30. 2

[9] Jason Dunn. Drone use taking flight on small farms. `https://www.munichre.com/topics-online/en/mobility-and-transport/drone-use-taking-flight-on-small-farms.html`. Accessed: 2021-11-30. 1

[10] Thomas Gleason Fahlstrom. *Introduction to UAV Systems 4e*. John Wiley & Sons, 2012. 1

[11] Oriol Rosales Garcia and Antonius Santoso. Comparative evaluation of drone delivery systemsin last-mile delivery. mathesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, June 2019. 1, 2

[12] Peter Jackson. Introduction to expert systems. 1986. 10

[13] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. 4, 5, 7, 10

[14] Robin Kellermann, Tobias Biehle, and Liliann Fischer. Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, 4:100088, 2020. 3

[15] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154, 2004. 6

[16] Jaihyun Lee. Optimization of a modular drone delivery system. In *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE, apr 2017. 2

[17] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. In *Science Robotics*, October 2021. 5

[18] Antonio Loquercio and Davide Scaramuzza. Learning to control drones in natural environments: A survey. In *e ICRA Workshop on Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding*, number CONF, 2018. 1, 3, 4

[19] Michal Mazur, Adam Wisniewski, Jeffery McMillan, Vicki Huff, Richard Abadie, Julian Smith, Stefan Stroh, Piotr Romanowski, Adam Krason, Filip Orlinski, Agnieszka Babicz, Paulina Lulkowska, Ewa Boguszewska, Bartosz Czerkies, Adam Glab, Rozalia Hologa, Jakub Jagiello, Weronika Jankowska-Tofani, Arkadiusz Kramza, Sandra Kuprijaniuk, Remigiusz Piwowarski, Jacek Sygutowski, Grzegorz Urban, Marek Walczak, Marek Wolski, and Ewa Zdrojowy. *Clarity from above.* PwC Poland, May 2016. 1, 2

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[21] Ellen Rosen. Skies aren't clogged with drones yet, but don't rule them out. *The New York Times*, 2019. 1, 3

[22] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. 2020. 5, 6

[23] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. 5

[24] United States Department of Defense. *Unmanned Aircraft System Airspace Integration Plan, Version 2.0*, March 2011. 1

[25] Unity. Unity. `https://unity.com/de/products/unity-platform`. Accessed: 2021-11-30. 6

[26] Adam C. Watts, Vincent G. Ambrosia, and Everett A. Hinkley. Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. *Remote Sensing*, 4(6):1671–1692, jun 2012. 1, 2, 10

[27] Jun Wei, E. Tazewell Ellet, Lisa Ellman, Patrick R. Rizzi, and Lu E. Tazewell Ellett Zhou. China launches first operational rules for civil unmanned aircraft. *Hogan Lovells*, January 2016. 1