DAI-Labor
TU Berlin

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 derives navigation decisions from current and past 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 civilian application possibilities, especially in the commercial sector [9]. Drones have already become today's standard in aerial inspection services [8, 1, 4]. In the near future, the areas of infrastructure, transport, insurance, media and entertainment, telecommunication, agriculture, safety and mining are considered to be particularly promising [19]. In contrast, real breakthroughs of more visionary applications, such as drone delivery or drone taxis are, if at all, only expected in the more distant future [21]. 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 here by conducting basic research on integrating temporal comprehension with a recurrent convolutional neural network into an autonomous navigation method in the closed test environment of drone racing.

In science as well as in this thesis proposal, the colloquial term "drone" refers to the aircraft class of unmanned aerial vehicles (UAV). 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, additional onboard computers provide the required 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. My master's thesis is primarily associated with quadcopters, the MAV representatives that, according to my observations, prevail among private, scientific and commercial applications as the "typical" drone model.

MAVs exhibit several substantial advantages over ground vehicles due to their aerial maneuverability and agility. 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 costs which is extremly valuable in the present context of big data and machine learning [19, 11]. On the downside, MAVs, as small aircraft vehicles, can move substantially less weight, whereby, computational power, mission-specific payload and battery capacity is severely limited. Limited battery capacity, in turn, restricts flight range and duration. Therefore, for MAVs paying off economically, efficiency is 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 MAVs 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 a 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 [7, 16]. So far, MAVs have proven their commercial value in fields associated with large industrial sites which usually provide flight environments that are essentially demarcated and controlled. State-of-the-art autonomous drones are capable of dealing with these highly predictable 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 contrast, the open-world environments of our daily lives are densely disturbed and are, thus, characterized by high uncertainty. The resulting technical complexity, notably with respect to "navigation, communication and automatization" [14], is a major cause that autonomous drones have not yet asserted themselves for commercial applications in those environments. Other main reasons are legal restrictions and lack of acceptance [21].

Autonomous navigation is a highly relevant topic in current UAV-related public research. State-of-the-art autonomous navigation methods lack in robustness to safely exhaust the full agility of UAVs in complex environments [18]. Navigation, as an integral functionality, comprises the main task of achieving a desired pose or

position while performing necessary sub-tasks at the same time [18]. The complexity of the flight environment determines the necessity of individual sub-tasks and, thus, the complexity of the navigation method. In controlled airspace navigating through pre-planned waypoints only relying on global navigation satellite system (GNSS) sensors may be sufficient. In uncontrolled airspace sub-tasks such as obstacle avoidance and the coordination with other agents are required. Under the assumption that temporal comprehension is a strong measure to increase robustness in autonomous navigation, this thesis intends to explore the effect of using a recurrent convolutional neural network in the environment of drone racing. The drone racing environment provides simplified test conditions, where a mere reactive control from race gate to race gate is sufficient, and is therefore very convenient for this basic research.

2 Objectives

Many advanced methods for autonomous navigation of MAVs integrate feedforward, deep convolutional neural networks (CNN) that, with a high spatial comprehension of the MAV's immediate surrounding, map the current color or depth image to prediction or action. With my thesis I aim to develop a navigation method for the drone racing environment using a recurrent convolutional neural network (R-CNN) that, due to its recurrency, can not only spatially but also temporally comprehend. I would like to examine if infering navigation decisions from present and **past** sensor data is beneficial to the robustness at high speeds and can cope with the event of intermediate target loss (see below).

The baseline of my thesis is a vision-based navigation method for drone racing developed by Kaufmann et al. from ETH Zurich in 2018 [13], that stood out with high reliability and agility on high speed, dynamic flight curves. The method is hybrid, i.e., it consists of a convolutional neural network (CNN) serially connected to a conventional control system At a frequency of 50 Hz, the following steps are repeatedly executed:

1. Input the current raw 300x200 RGB image from the onboard, forward-facing camera to the CNN.

2. The CNN predicts a waypoint in the 2D reference frame of the image and a normalized speed: $x_{\mathrm{IRF}}, y_{\mathrm{IRF}} \in [-1, 1]$, $v_{\mathrm{NORM}} \in [0, 1]$

3. Transform the waypoint from the image reference frame to the reference frame of the autopilot state estimation: $x_{\mathrm{ARF}}, y_{\mathrm{ARF}}, z_{\mathrm{ARF}} \in \mathbb{R}^3$.

4. Compute a minimum-jerk (time derivative of acceleration) trajectory from the current autopilot state estimate to the waypoint. The duration of the trajectory depends on the predicted normalized speed v_{NORM}.

5. Push the states of the trajectory to the autopilot.

6. The autpilot, using conventional algorithms, generates control commands and triggers their execution. Thereby, the end state of the trajectory is never reached since the re-planning of the trajectory is continually triggered by new waypoints.

According to the paper, the baseline has a success rate of 100 % on the simulated racetrack when the maximum speed is not greater than 9 m/s. For higher maximum speeds {10, 11, 12} m/s the success rate decreases to approximately {85, 60, 35} %. Besides maximum speed, the simulated racetrack is designed in way that at any time the currently targeted race gate is located in the frame of view (FOV) of the onboard camera. This is a requirement of the baseline because the CNN derives its navigation decisions only from the current image. In the event of intermediate target loss, i.e., there is no race gate in the FOV and, consequently, on the image, the baseline must result in undefined behaviour. Intermediate target loss could for example happen in the case of a steep curve between two consecutive race gates or in the case of an obstacle that temporally blocks the view to the currently targeted gate.

In my thesis I plan to further develop only the first part of the hybrid approach. I intend to, first, replace the CNN with a R-CNN and, second, feeding additional features, i.e., data from the inertial measurement unit (IMU), to the network. The IMU data encompasses three (x, y, z) linear accelerations and three (x, y, z) angular velocities. I expect that thereby, waypoints are not only generated on the basis of the current RGB image and IMU data, but that also past sensor data is included in the navigation decision. This would possibly result in the following positive effects:

- Making decisions on the basis of a series of sensor data makes the network more robust against outliers. Otherwise, at high speeds, outliers may directly result in a crash.

- Considering the similarity of recurrency to mathematical integration, feeding IMU data to the network could have great potential since the network could be more or less able to internally estimate positions and orientations.

- Due to its "memory", the network is able to generate meaningful waypoints in the case of intermediate target loss, i.e., the next gate of the race track is not depicted in the current image, but has appeared in previous images.

- Because 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 trajectory.

The approach is implemented utilizing the middleware ROS [23] and simulated with the photo-realistic Flightmare Simulator [22] built on Unity. For the implementation concept, see section 6.4. In simulation, the following tests should be conducted to compare my approach to the baseline.

Randomized Figure-8 Drone Racing Conduct test runs with increasing maximum speeds. For each run, build a randomized figure-8 drone racing track and test my approach and the baseline on the track. For each maximum speed, evaluate the percentage share of successful runs, the average time of racetrack completion and the closeness to the global trajectory in terms of optimality. The latter could be computed by recording the reference states that are continually pushed to the autopilot, taking the 4th derivative with respect to time (snap) and integrate the values. The in this way calculated cost could be used to measure closeness to the optimal, minimum snap global trajectory which the expert system used to generate the training data.

Intermediate target loss on 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 baseline, 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. This becomes especially true, if the R-CNN is able to make use of the IMU data estimating poses. The percentage share of successful runs would be a convenient metric for evaluation.

3 Task Packages

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

- Build the drone racing simulation.

    [X] Familiarize with Flightmare [22], a photo-realistic simulator for quad-copters.

    [X] Familiarize with the Unity Engine [25], to create and manipulate the rendered environments in Flightmare.

    [X] Transfer the already implemented simulation from Gazebo [15] to Flight-mare. For exemplary training data generated in Gazebo see section 6.3.

- Generate training data in simulation.

    [X] Adjust the already implemented expert system (see section 6.2), so that the training data includes an RGB image and IMU data of the drone as features and a waypoint and desired speed as labels.

    [X] 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 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.

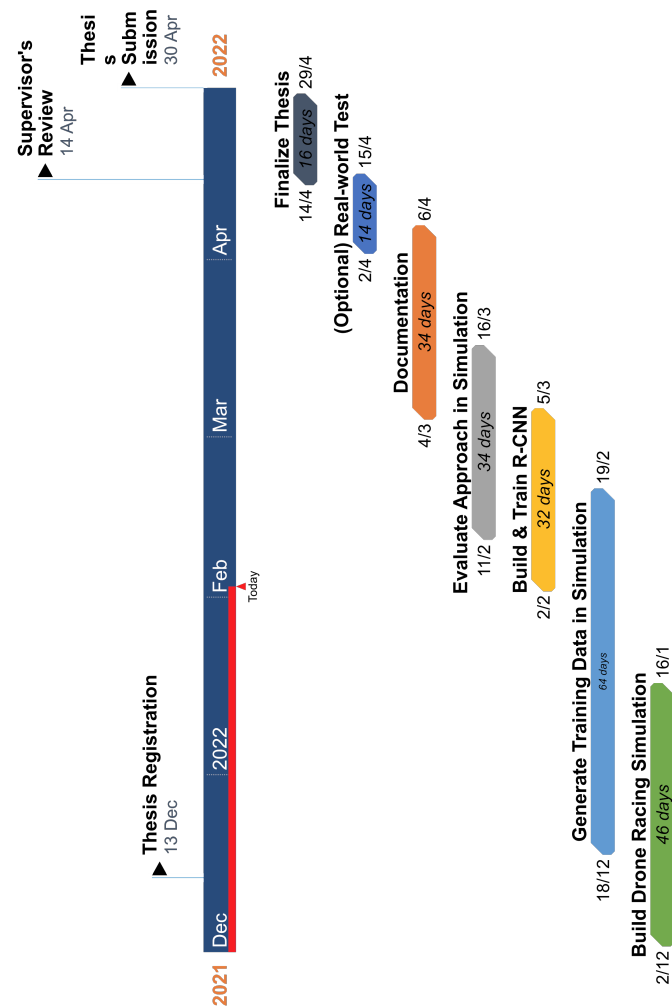 - Conduct test, evaluate and document in thesis.

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: 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). *Compiled 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. I extended the expert in order that it also records the IMU data as labels. The following structure summarizes the expert system:

6 Annex

- 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, IMU data) 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 a user-specified minimum and maximum speed as well as a 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.

6 Annex

6.3 Labeled training data generated in simulation with Flightmare/Unity

(a) **IMU Data**
$a_x = -0.731,\ a_y = -0.030,\ a_z = 9.610$
$\omega_x = 0.074,\ \omega_y = -0.006,\ \omega_z = 0.479$

Labels
$x_{\text{IRF}} = -0.302,\ y_{\text{IRF}} = 0.077$
$v_{\text{NORM}} = 0.892$

(b) **IMU Data**
$a_x = -0.650,\ a_y = -0.058,\ a_z = 9.675$
$\omega_x = 0.001,\ \omega_y = 0.148,\ \omega_z = 0.519$

Labels
$x_{\text{IRF}} = -0.428,\ y_{\text{IRF}} = 0.136$
$v_{\text{NORM}} = 0.920$

(c) **IMU Data**
$a_x - 0.714 =,\ a_y = 0.059,\ a_z = 9.608$
$\omega_x = 0.058,\ \omega_y = -0.153,\ \omega_z = -0.362$

Labels
$x_{\text{IRF}} = 0.403,\ y_{\text{IRF}} = -0.003$
$v_{\text{NORM}} = 0.893$

(d) **IMU Data**
$a_x = -0.625,\ a_y = 0.034,\ a_z = 9.620$
$\omega_x = 0.003,\ \omega_y = -0.068,\ \omega_z = -0.207$

Labels
$x_{\text{IRF}} = 0.183,\ y_{\text{IRF}} = -0.128$
$v_{\text{NORM}} = 0.756$

(e) **IMU Data**
$a_x = -0.648,\ a_y = 0.044,\ a_z = 9.781$
$\omega_x = 0.019,\ \omega_y = 0.029,\ \omega_z = -0.295$

Labels
$x_{\text{IRF}} = 0.293,\ y_{\text{IRF}} = 0.104$
$v_{\text{NORM}} = 0.749$

Figure 6.1: Five Samples of labeled training data, generated in simulation of a randomized drone figure-8 racetrack in five different scenes with Flightmare/Unity. Features: 720x480 RGB image, linear accelerations and angular velocities of IMU. Labels: x-, y-coordinate of waypoint in image reference frame, normalized speed.
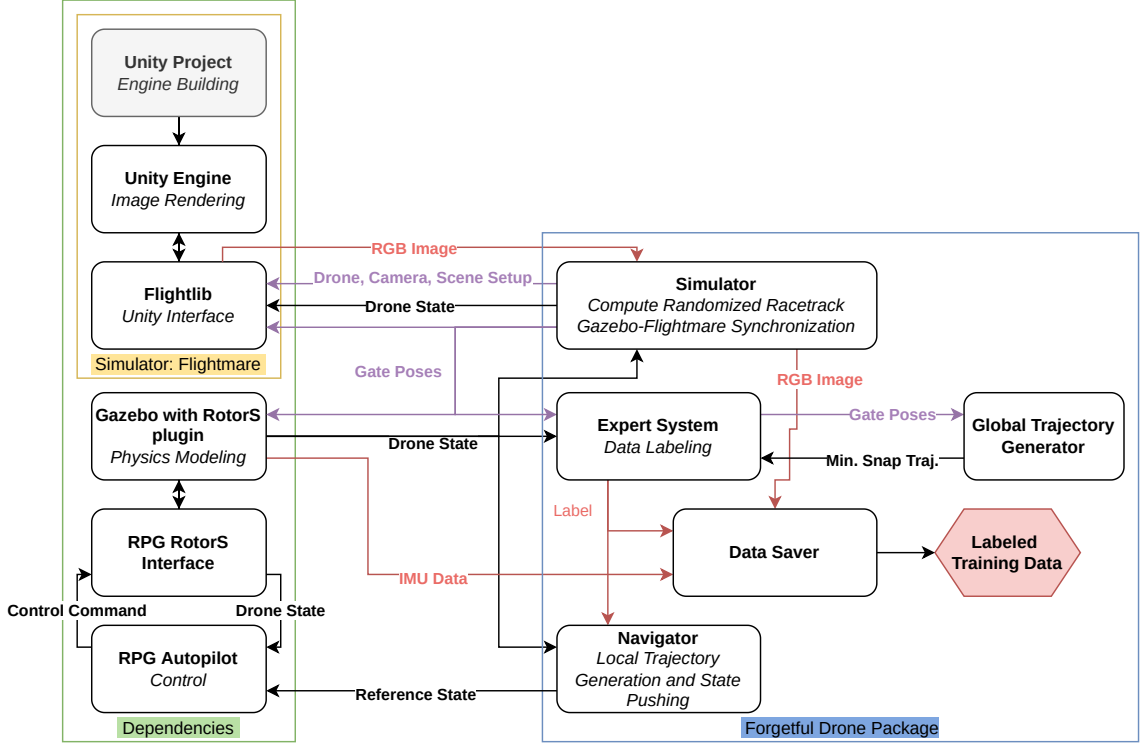
6.4 Implementation Concepts

Figure 6.2: **Implementation Concept, Training Data Generation:** The Simulator computes a drone racetrack consisting of randomized gate poses. It sends the race gate poses to Flightmare to render the gates, to Gazebo to model collisions and to the Expert System. The Expert System forwards the gate poses to the Global Trajectory Generator which computes a minimum snap trajectory and sends it back to the Expert System. Then the Simulator sets up Flightmare (drone and camera configurations, scene, site and race gate type). Then a run to generate training data starts. The Simulator continually receives the ground truth state of the drone and sends it to Flightmare for rendering. Additionally, the Simulator continually receives an RGB image from Flightmare and sends it to the data saver. The Expert system, based on the ground truth state of the drone and the global trajectory, computes the label (2D waypoint in image coordinates, normalized speed) and sends it to the Data Saver and the Navigator. In addition to the RGB image and the labels, the Data Saver is receiving the ground truth IMU data from Gazebo. The data saver saves the features (RGB image and IMU data) and the labels. The Navigator, based on the labels, computes a minimum jerk local trajectory and pushes the states to the RPG Autopilot. The RPG Autopilot tracks the reference states receiving ground truth drone states.
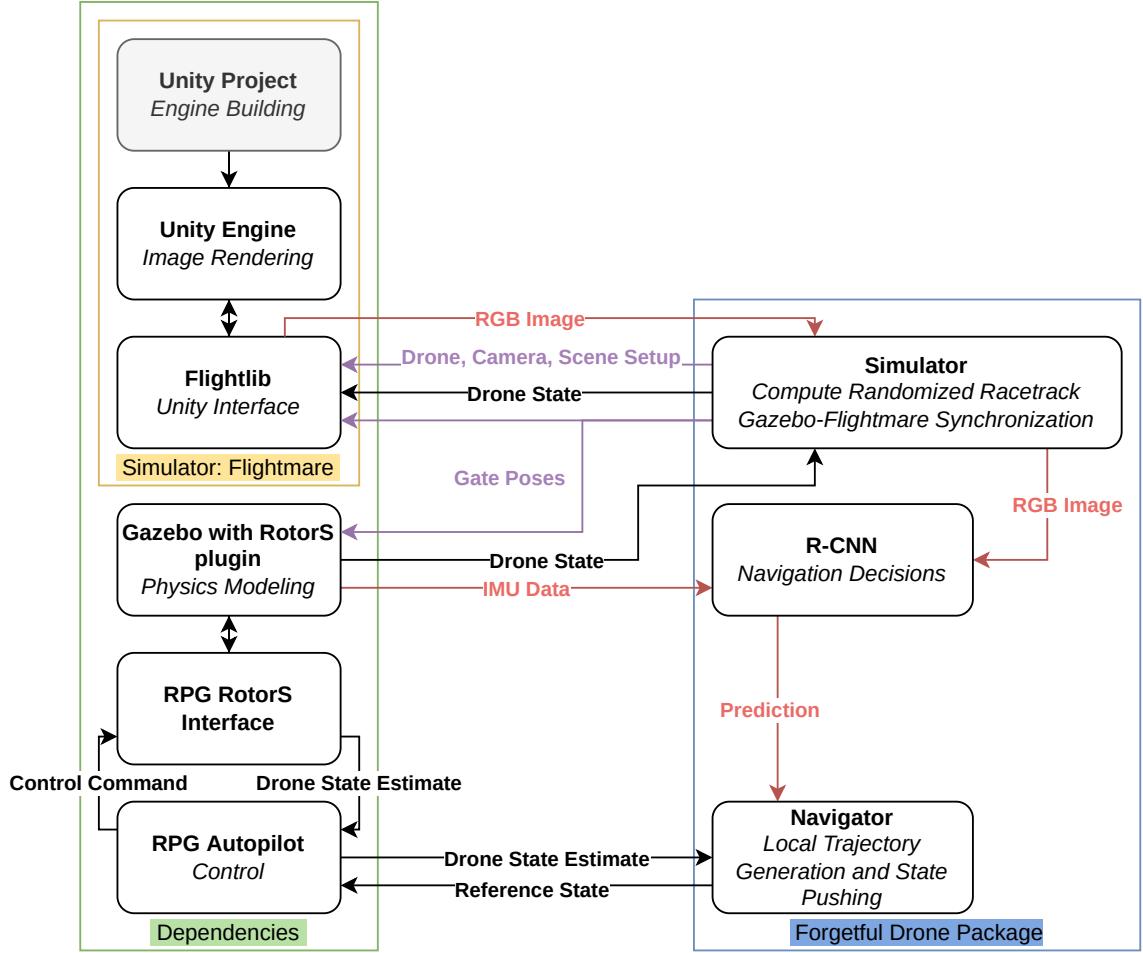
Figure 6.3: **Implementation Concept, Network Testing:** In comparison with above, the R-CNN substitutes the expert system by making predicitons based on the RGB image and the IMU data. All ground truth drone states are replaced with state estimates, except for the synchronization of Gazebo and Flightmare.

Bibliography

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

[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. 7

[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. 1

[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

Bibliography

[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. 1

[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. 2

[12] Peter Jackson. Introduction to expert systems. 1986. 11

[13] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. 4, 8, 11

[14] Robin Kellermann, Tobias Biehle, and Liliann Fischer. Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, 4:100088, 2020. 2

[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. 7

[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.

Bibliography

[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, 2, 3

[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. 7

[21] Ellen Rosen. Skies aren't clogged with drones yet, but don't rule them out. *The New York Times*, 2019. 1, 2

[22] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. 2020. 6, 7

[23] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. 6

[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. 7

[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, 11

[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