Interpretation of neural networks and advanced image augmentation for visual control of drones in human proximity

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics

presented by
Marco Ferri

under the supervision of
Prof. Alessandro Giusti
co-supervised by
Dr. Dario Mantegazza

February 2021

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Marco Ferri
Lugano, 22 February 2021

To someone

"Sometimes it is the people no one
can imagine anything of, who do
the things no one can imagine."
The Imitation Game

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam vulputate erat quis justo varius vehicula. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; In ut placerat velit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In elementum egestas turpis et auctor. Vestibulum gravida lorem nec egestas ornare. Duis varius arcu imperdiet, feugiat odio in, facilisis est. Quisque interdum vitae odio ut vehicula. Etiam molestie enim non risus maximus, vitae efficitur mauris sollicitudin. Phasellus consequat nulla at nulla tempus varius. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nulla faucibus aliquam nisl, vel luctus arcu semper vel. Aliquam ipsum risus, feugiat quis nulla ac, aliquet imperdiet ante. Ut et massa sem. Donec eu ex augue. Nam urna nunc, commodo ac nunc et, auctor mattis nunc. Nam pellentesque laoreet purus, a tristique nisi auctor non. Integer sed congue lorem. Maecenas faucibus turpis nec ultrices tempor. Vivamus condimentum nibh sit amet molestie tempor. Pellentesque cursus diam maximus nisi gravida, sed consequat mauris malesuada. Fusce eget nisl vehicula, porta enim sit amet, ornare massa. Nunc et ex a eros tempor mattis in quis quam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat ex nec ante tempus, quis bibendum nisi posuere. Cras augue nulla, ornare vel risus quis, vulputate bibendum sem.

Proin placerat euismod cursus. Nulla ornare lobortis ligula et pellentesque. Nullam cursus neque ut fermentum euismod. Sed sit amet luctus orci. Nunc convallis urna id nisl vestibulum ullamcorper. Aliquam ullamcorper porta dui et aliquam. Pellentesque urna nibh, finibus sed condimentum eget, interdum ut tellus. Morbi aliquet, erat et rhoncus cursus, lectus nibh vulputate nisl, eu interdum dui dolor quis ipsum. Donec id libero sit amet orci gravida pretium. Mauris in magna non nunc posuere consequat. Donec diam tortor, viverra posuere velit et, convallis commodo massa. Fusce consectetur posuere ex, nec tincidunt neque posuere tempus. Vivamus vitae accumsan ligula. Nulla facilisi. Donec pellentesque commodo lorem ac semper.

Acknowledgements

Thanks to...

Contents

Abstract **IV**

Acknowledgements **V**

List of Figures **VIII**

List of Tables **IX**

1 Introduction **1**
1.1 Objective . . . 1
1.2 Outline . . . 2

2 Theoretical Foundation **3**
2.1 Robotics . . . 3
2.2 Machine Learning . . . 3
2.3 Human-drone Interaction . . . 3
2.3.1 The State of the Art of Human–Drone Interaction . . . 3
2.3.2 Vision-based Control of a Quadrotor in User Proximity . . . 4
2.3.3 Embedded Implementation of Visual Controller for Nano-Drones 5
2.4 Network Interpretability . . . 6
2.5 Network Generalization . . . 6

3 System Description **7**
3.1 Environment . . . 7
3.1.1 Parrot Bebop Drone 2 . . . 7
3.1.2 OptiTrack . . . 8
3.1.3 Drone Arena . . . 8
3.1.4 Robot Operating System (ROS) . . . 9
3.1.5 Control & Data collection . . . 10
3.2 Model . . . 12
3.2.1 Dataset . . . 12
3.2.2 Architecture . . . 15
3.2.3 Performance . . . 16

3.2.4 Generalization . . . 17
3.3 Development . . . 18
3.3.1 Tools . . . 18
3.3.2 Frameworks . . . 18

4 Initial Experiments & Design **21**
4.1 Model Interpretation . . . 22
4.1.1 Regression to Classification . . . 22
4.1.2 Loss Functions . . . 22
4.1.3 Visual Results . . . 22
4.1.4 Proposed Approach . . . 22
4.2 Person Masking . . . 22
4.2.1 Canny Edge Detection . . . 22
4.2.2 Grabcut . . . 22
4.2.3 YOLO . . . 22
4.2.4 Face & Head Detection . . . 22
4.2.5 Mask R-CNN . . . 22

5 Implementation **23**

6 Evaluation **24**

7 Conclusion **25**
7.1 Final Thoughts . . . 25
7.2 Future Works . . . 25

8 Latex features **26**

A Extra Figures **28**

B Acronyms **31**

Bibliography **32**

Figures

3.1 Parrot Bebop Drone 2 7
3.2 Schematic OptiTrack system with 12 OptiTrack Prime cameras . . 9
3.3 Drone arena at Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA) 9
3.4 Markers placed on top of drone and user's head 10
3.5 OptiTrack and data collection illustration 11
3.6 A frame with digital artifact caused by connection issues 12
3.7 A complete overview of images in the training set 13
3.8 A movements sequence which led to images with no person presents 14
3.9 Target variables distribution for the regression task 14
3.10 Schematic ProximityNet architecture 15
3.11 ProximityNet R Squared (R^2) results from Mantegazza et al. [2019] 16
3.12 ProximityNet qualitative GT vs prediction results from Mantegazza et al. [2019] 17
3.13 ProximityNet trajectories, from Mantegazza et al. [2019], for positioning in front of the user initially rotated by 90 degrees 17

8.1 list of figures caption 26
8.2 list of figures caption 26

A.1 ProximityNet complete architecture (part 1) 29
A.2 ProximityNet complete architecture (part 2, from layer 18 to outputs) 30

Tables

8.1 Schema originale del dataset 27

Chapter 1

Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1.1 Objective

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc

eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

1.2 Outline

The thesis is composed of X chapters, whose main points are presented as follows:

- Chapter 5 summarises the previous research on the topic, evaluating the approaches adopted by the authors;
- Chapter 5 provides the background knowledge needed to properly understand the research contents;
- Chapter 5 presents the tools used for the data collection and all the additional frameworks we relied on;
- Chapter 5 thoroughly illustrates the methodology used, their benefits and limitations, also including descriptions of the kind of data used and how they are collected;
- Chapter 5 explores the analysis conducted and shows evaluation results;
- The 5 addresses the results of the experiments, concludes the thesis by discussing the implications of our findings, possible improvements and outlines future works.

Chapter 2

Theoretical Foundation

2.1 Robotics

2.2 Machine Learning

2.3 Human-drone Interaction

A good variety of research can be found on human-robot interaction and a lot is yet to come. In the field, drones represent a specific segment due to their ability of freely move in the 3D space, opening the access to new use cases while representing a real challenge for professionals and researchers.

In this section we firstly present a general overview on the topic, then we focus on related work by IDSIA.

2.3.1 The State of the Art of Human–Drone Interaction

A recent article, published in Nov 2019 for IEEE Access (Tezza and Andujar [2019]), explores literature and state of the art for human-drone interaction. Drones range from small toy-grade remote-controlled aircraft to fully-autonomous systems capable of decision-making through a large variety of sensors. Their usage grew a lot in the last years and it is expected to keep growing, thanks to decreasing costs and powerful features they can provide both for personal, commercial and social usage.

United States Federal Aviation Administration (FAA) expects that total drone registrations will increase by more than 60% between 2018 and 2022 with a particular increment in the commercial sector rather than the hobbyist one, even though the latter still counts the largest number of units. Moreover, FAA reports that almost half of the drones usage is for aerial photography (48%), followed by industrial inspection (28%) and agriculture (17%). Accordingly to Tezza and Andujar [2019], drones will become ubiquitous to society, and in the next decade they will be extensively used in advertising, shipping, sports, emergency, and many other fields for

augmenting human capabilities.

Main concerns about drones today regards safety issues caused by propellers and limited flight times, usually no longer than 30 minutes due to limited battery capacity. Research in the sector of human-drone interaction mainly focus on their control (through gestures, voice or custom interfaces), communication between the user and the drone itself (in terms of acknowledgment and intents), perception of users' safety during flight, and innovative use cases.

2.3.2 Vision-based Control of a Quadrotor in User Proximity

Our work is built upon the original master thesis (Mantegazza [2018]) and paper (Mantegazza et al. [2019]) *Vision-based Control of a Quadrotor in User Proximity: Mediated vs End-to-End Learning Approaches* from Dario Mantegazza, developed at IDSIA in Lugano. In his thesis, the author proposes a machine learning model for teaching a drone to interact with a person by continuously flying to face the user frontally, towards the direction of the head. The problem is approached as a reactive control procedure and addressed with supervised learning, thus provides an interesting starting point for many other robotics applications.

The author collect data and test his model on the Parrot Bebop 2, a 500grams drone commonly used for photography and leisure purposes, capable of effective video stabilization. However, the software runs off-board, on a dedicated computer remotely connected through WiFi.

A considerable amount of flights is recorded for building the training data by programmatically flying the drone in front of a person, controlling it through an omniscient controller which knows both the drone's and user's pose. Images produced by the front-facing camera of the drone are used as input for a custom designed Residual Neural Network (ResNet) architecture to infer the relative user's position with respect to (wrt) the drone. Practically, the neural network performs a regression on the four variables that form the user's pose (X, Y, Z, YAW) and learns to predict their values by using spacial information contained into the input images.

In the paper, the author also makes a comparison between the mediated approach described above and another end-to-end approach that directly learns control signals[1] for the drone, instead of the user's pose. Both solutions provide similar results, but the former can be adapted to other tasks by simply designing a custom controller, providing a more transparent and analyzable solution.

Even though this kind of problems on human recognition and pose estimation could be faced with more advanced deep learning algorithms, making a simple regression on four variables allows the network to be small, so that the prediction task is light, fast to execute, and possibly portable on low-end devices.

Network and dataset defined in Mantegazza et al. [2019] have been used for

[1] desired pitch, roll, yaw and vertical velocity

our entire project, so the original code repository[2] is available for reference. Next chapters constantly make use of this particular model architecture, that will be further explained in section 3.2.

Having no official name, for enhancing readability, the custom ResNet architecture proposed by the author will be simply called *ProximityNet*. For a better understanding, also a good descriptive video is available at `https://drive.switch.ch/index.php/s/MlEDrsuHcSl5Aw5`.

2.3.3 Embedded Implementation of Visual Controller for Nano-Drones

Autonomous navigation is an important and well-known area of research in robotics, which usually requires to accomplish complex and computationally-expensive tasks such as localization, mapping and path planning. Recent studies have started to approach autonomous driving through deep learning and imitation learningHussein et al. [2017], where neural networks learn by imitating human behavior in specific tasks.

In 2018, researchers at the UZH University of Zürich have demonstrated that ResNets are able to provide satisfactory performance in the field [Loquercio et al. [2018]]. They developed DroNet, a forked Convolutional Neural Network (CNN) that predicts, from a single gray-scale image, a steering angle and a collision probability. In other words, the model learns to steer and avoid obstacles from forward-looking videos recorded by cars and bikes while driving in real contexts. In this case, both the prediction and controller tasks were powered off-drone on a dedicated computer, remotely connected through WiFi.

A year later, ETH Zürich was able to develop PULP-DroNet, porting the CNN on the Crazyflie[3], a nano-drone with a size of only 3 3×3 centimeters for a weight of 27 grams. They propose a general methodology for deploying on-board deep learning algorithms for ultra-low-power devices Palossi et al. [2019b], without any needs of an external laptop to run the software.

Inspired by PULP-DroNet, IDSIA adapted its ProximityNet to work on-board the Crazyflie with excellent results Zimmerman [2020]. The nano-drone is able to achieve good quantitative and qualitative performance, regardless any problem deriving from working with such low-end devices. Main challenges are represented by low computational power, energy consumption management, and low-fidelity camera with no video stabilization[4].

[2] `https://github.com/idsia-robotics/proximity-quadrotor-learning`

[3] `https://www.bitcraze.io/products/crazyflie-2-1/`

[4] Himax HM01B0 camera, able to produce 320×320 megapixel (MP) at 60 FPS. However, frame rate is incredibly reduced during data collection due to platform limitation for image transfer.

2.4 Network Interpretability

2.5 Network Generalization

Chapter 3

System Description

This chapter aims to provide a generic view of our system.

First, we briefly describe the existing environment, its main components and how they interact for flying and controlling the drone. Next, we explain network architecture and data used for the machine learning model, able to predict the user's pose given an image. Finally, we list tools, software and libraries to achieve the goal of making the drone able to fly in any other environment.

3.1 Environment

Since we mainly focus on improving the ProximityNet model mentioned in section 2.3.2, we need to understand the environment in which the original research has been conducted, physically located at the Swiss AI Lab Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA) in Lugano.

3.1.1 Parrot Bebop Drone 2

Figure 3.1: Parrot Bebop Drone 2

The entire work is built around the Parrot Bebop Drone 2 (figure 3.1), a lightweight drone (500 grams) with a size of $382 \times 328 \times 89$ millimeters. A 2700 mAh swappable battery gives power to four brushless engines and dual-core processor with

quad-core GPU for a maximum flight time of 25 minutes. Connectivity is provided through 2.4 GHz 802.11a/b/n/ac Wi-Fi that enables remote control via mobile app or Parrot Skycontroller (up to a distance of 2km).

The drone is equipped with many simultaneous sensor to compute drone's velocities, orientation, altitude, attitude and GPS coordinates to ensure the maximum stability during the whole flight. However, for this project we mainly care about its camera, able to shoot 14 megapixel (MP) photos and record Full HD 1080p videos at 30 frames per second (FPS). Even though the original field of view (FOV) is 180°, raw camera images pass through a software stabilization that produces 16:9 images with a horizontal FOV of 90°. The 3-axis digital stabilization technique implemented by Parrot is able to compensate for drone's pitch and roll, in order to provide correct-oriented horizontal images and stable videos regardless the drone's movements. Full specifications provided by the official Parrot Documentation [2015].

3.1.2 OptiTrack

For tracking drone's movement a motion capture (MoCap) system is required, able to record 3D coordinate of objects and people in space. The technique is widely use for motion tracking in a large variety of fields such as film making and animation, virtual reality, sport, medicine and even military. A common way to implement a MoCap systems is by using special cameras placed around the area to be tracked, able to collect optical signals from passive[1] or active markers[2] inside the area.

IDSIA adopt OptiTrack, which is producing real-time MoCap systems since 1996 and are the today world's choice for low-latency and high-precision 6 degrees of freedom (DoF) tracking for ground and aerial robotics, both indoor and outdoor. Full documentation is available on the OptiTrack Website.

3.1.3 Drone Arena

At IDSIA, a dedicated room has been equipped with an OptiTrack MoCap system composed by 12 OptiTrack Prime^{x}13 infrared (IR) cameras for medium-sized areas (figure 3.3a, 1.3 MP, 240 FPS, $\pm$0.20 mm 3D accuracy in a 9×9 meters area with 14mm marker), to track movements of passive markers placed on the person's head facing the drone and on the drone itself. Schematic and actual representation of the arena are shown in figures 3.2 and 3.3b. Such composition is able to track a theoretical number of 18 drones inside an available area of 6×6 meters (here surrounded by a safety net), with a virtual fence of 4.8×4.8 meters which virtually constraints the total area in which the drone is allowed to fly.

[1] a passive marker reflect light

[2] an active marker emits its own light

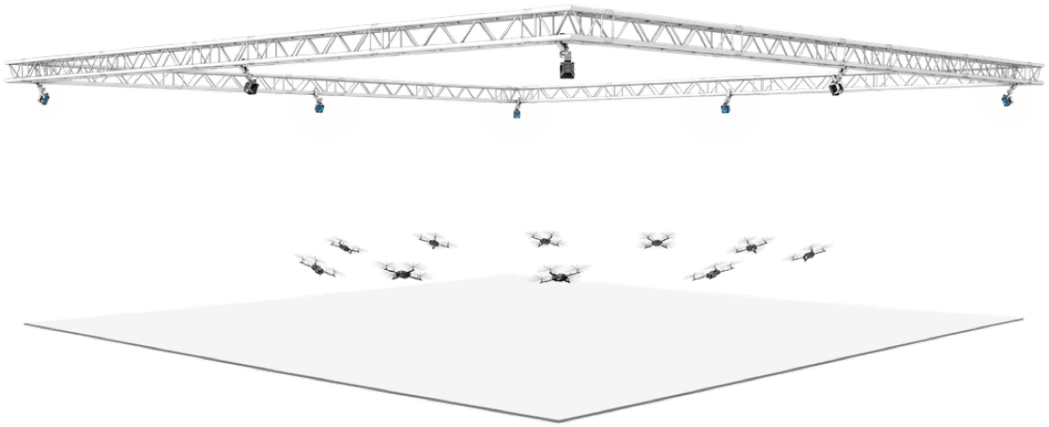
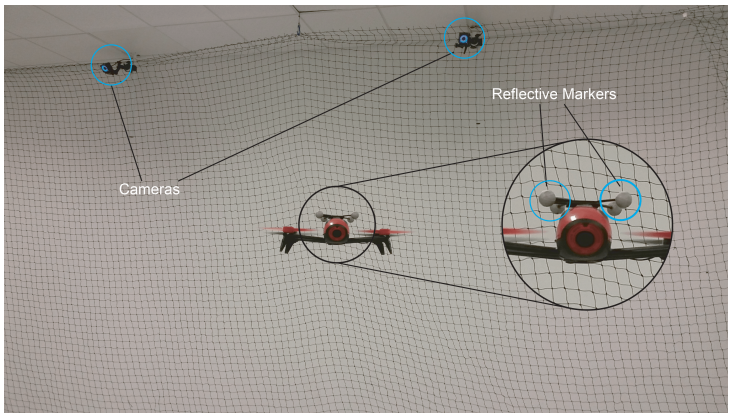


Figure 3.2: Schematic OptiTrack system with 12 OptiTrack Prime cameras

(a) Prime^{x}13 camera



(b) Drone with passive markers, from Mantegazza [2018]

Figure 3.3: Drone arena at IDSIA

3.1.4 Robot Operating System (ROS)

Robot Operating System (ROS) is an open-source robotics middleware suite of software libraries and tools for building distributed and modular robot applications. It provides hardware astraction and orchestration, implementation of commonly used functionality, message-passing between processes, and package management. ROS organizes its components in graph architecture composed by nodes which communicates via a publish/subscribe mechanism, supporting a wide variety of robots also used for education. The main client library is available in C++, Python and Lisp.

Some of the most important ROS features include Standard Message Definitions, Robot Geometry and Description, Remote Procedure Calls, Diagnostics, Pose Estimation, Localization, Mapping, and Navigation. It also provides additional tools, such as Rviz (3D visualization of robots and various types of sensor data) and

Gazebo (3D indoor and outdoor multi-robot simulator, complete with dynamic and kinematic physics, and a pluggable physics engine).

ROS has grown to include a large community of active users worldwide. Historically, the majority of the users were in research labs, but increasingly we are seeing adoption in the commercial sector, particularly in industrial and service robotics.

Further documentation is available on the official ROS Website.

3.1.5 Control & Data collection

Inside the arena, the drone is controlled by a ROS script which relies on the user's pose with respect to (wrt) the drone - from now on, the *target pose* (i.e., the pose of the user seen by the drone reference frame) - to compute acceleration commands for making the drone hover in front of the person, in the direction of the head at a predefined 1.5 meters distance.

During data collection, both user's and drone's poses are deduced by the OptiTrack system by using proper markers placed on the drone and on the person's head, as shown in picture 3.4. The target poses over time, mathematically computed by the script, are accurately synchronized with the video stream from the front-facing camera and saved into `rosbag` files.

Figure 3.4: Markers placed on top of drone and user's head, from Mantegazza [2018]

Data collected into the drone arena have been used to build the dataset for training a machine learning model, which should be able to infer the target pose by seeing a picture taken by the drone's camera. Figure 3.5 shows an illustration of the system from a bird-eye view.

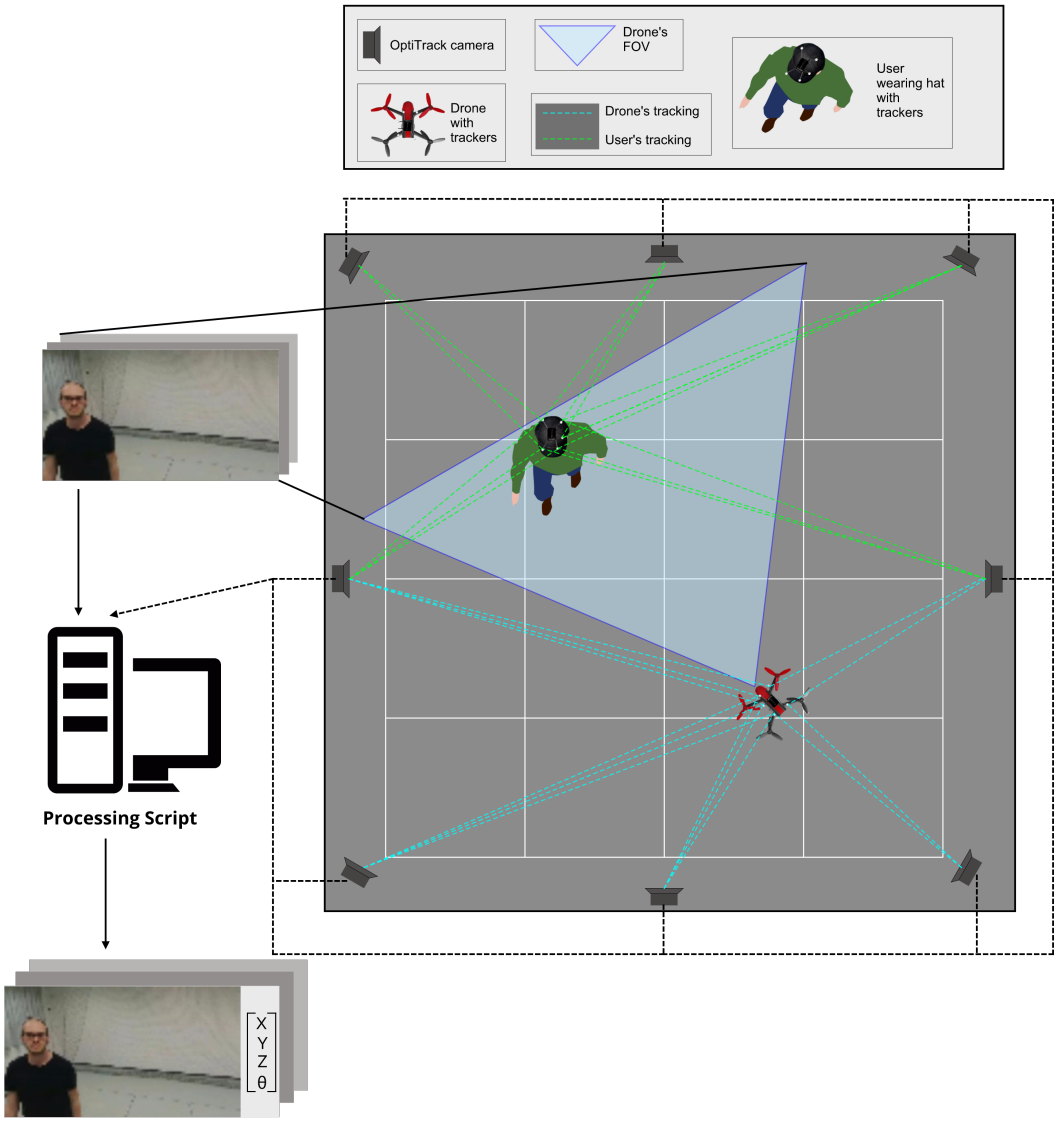


Figure 3.5: OptiTrack and data collection illustration, from Mantegazza [2018]

3.2 Model

Our entire work is based on the work introduced in section 2.3.2, whose basic working and environment has been presented before. This section further inspects dataset composition, network architecture and model performance as declared in Mantegazza et al. [2019].

3.2.1 Dataset

Data have been entirely collected in the dedicated drone arena located at IDSIA. A good dataset should ideally provide images from various scenarios, but such kind of data are not easy to record since the ground truth must be given by a complex and expensive MoCap system, particularly difficult to be moved and reassembled outdoor.

For building both the training and the testing set, several flight sessions have been recorded using an omniscient controller, driving the drone towards user's pose inferred by the OptiTrack. Dataset contains a total of 13 different people which differs both in physical characteristics and outfit, moving in different ways under various (artificial) light conditions. Many objects are present on the background of recorded images, and some experiments involve more than one person in front of the drone[3]. In total, 45 minutes of usable videos were used to compose the dataset, which counts about 63'000 and 11'000 frames respectively for training and testing.

A complete overview of images present in the training set is shown in figure 3.7. Please notice that a few frames in the dataset are affected by digital artifact, mainly caused by connection issues during video recording (figure 3.6); also, there are frames in which no person is present at all, because of particular movements sequences during which the drone actually lose the user (figure 3.8).

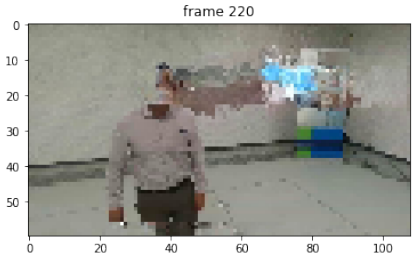


Figure 3.6: A frame with digital artifact caused by connection issues

[3]anyway, the drone always had to follow the nearest user (properly equipped with OptiTrack markers)

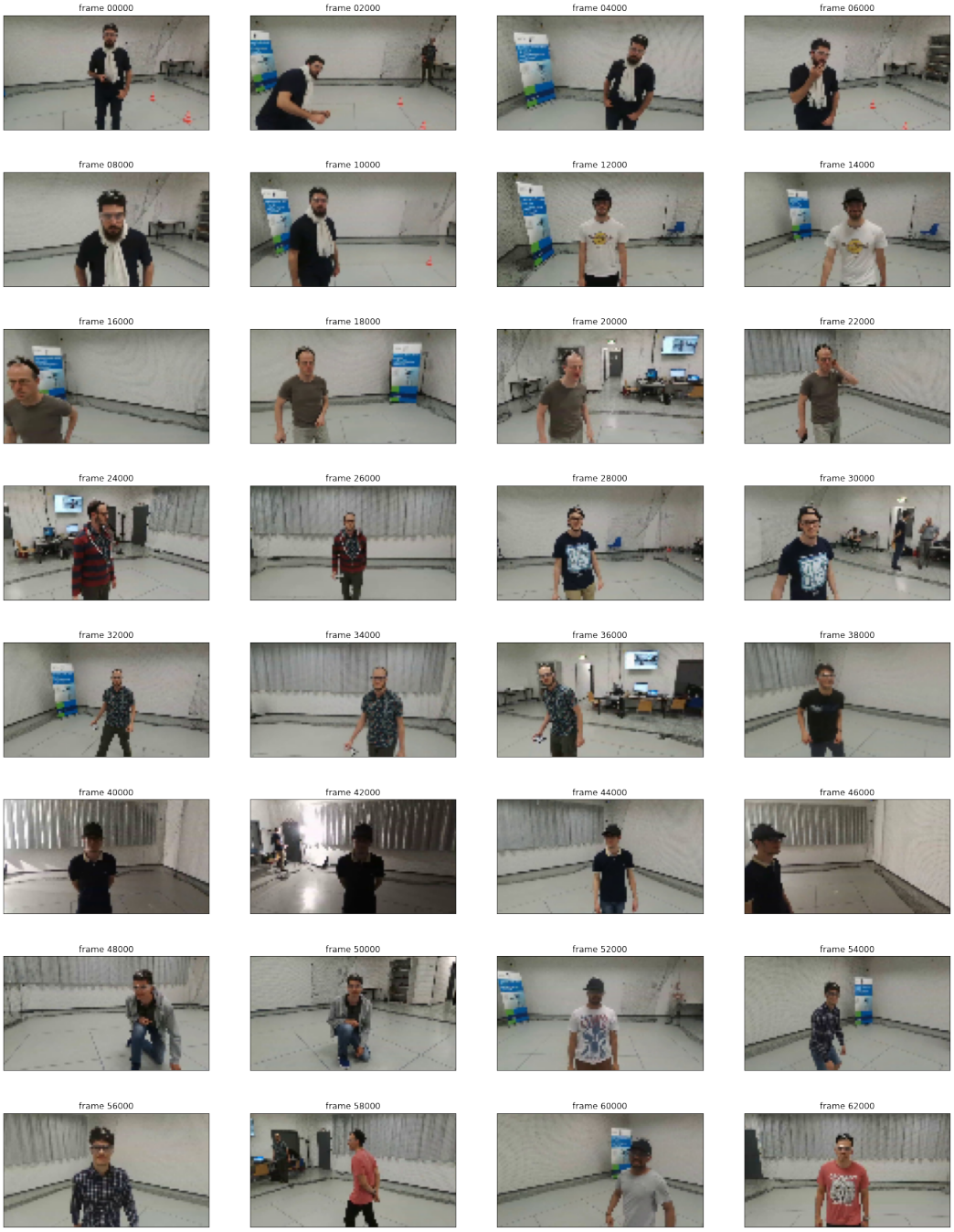


Figure 3.7: A complete overview of images in the training set

Figure 3.8: A movements sequence which led to images with no person presents

The ground truth is represented by the four variables, associated with each captured image, that explain the user's pose wrt the drone. Their interpretation is shown here, together with their distribution in the training set (figure 3.9).

- `X`, stays for the distance of the user from the drone and affects the pitch (acceleration along the X axis); if the user is at the correct distance in front of the drone, this variable will be equal to 1.5
- `Y`, represents the horizontal alignment of the user in front of the drone and affects the roll (acceleration along the Y axis); if the user is horizontally centered in front of the drone, this variable will be equal to 0
- `Z`, represents the vertical alignment of the user in front of the drone and affects the velocity along the Z axis; if the user is vertically centered in front of the drone, this variable will be equal to 0
- `W`, represents the angle created between head's pointing direction and drone position, is influenced by head orientation and affects the yaw (angular velocity around the Z axis); if the user is perfectly facing the drone, this variable will be equal to 0

From the variables distribution shown in figure 3.9 we notice that, most of the time, the user is somehow centered in the image, which is an effect caused by the flight controller based on known poses. The variation of the variables is affected by the user's movements in space, the more sudden they are, the greater the deviation.

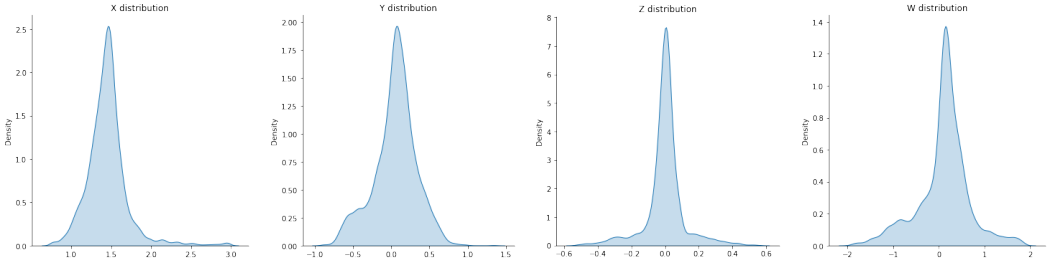


Figure 3.9: Target variables distribution for the regression task

3.2.2 Architecture

As perfectly suited for working with images, proposed network resembles a Convolutional Neural Network (CNN) which has been later improved by the author for being a Residual Neural Network (ResNet). The network is composed by a total of 1'332'484 trainable parameters, accepts a single 60×108 pixels image in input, and outputs 4 regression variables which corresponds to the user's pose coordinates.

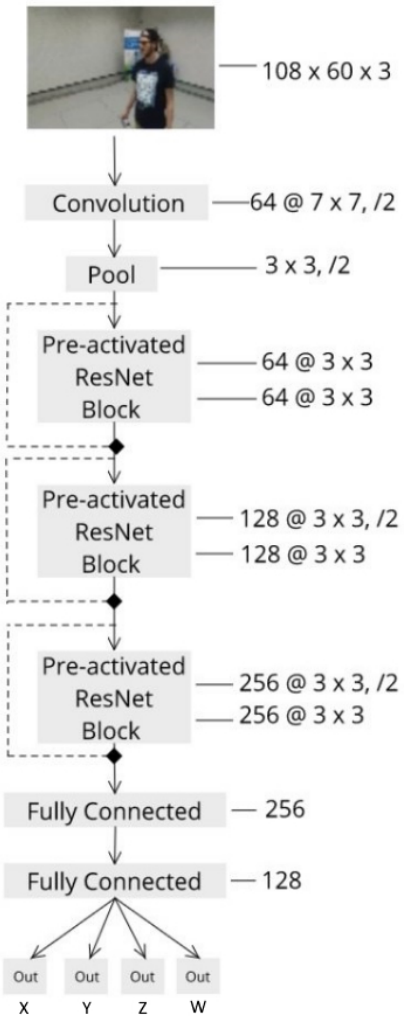


Figure 3.10: Schematic ProximityNet architecture, from Mantegazza et al. [2019]

Figure 3.10 provides an illustration of the architecture, while a complete list of all layers is available in figures A.1 and A.1 of the appendix A. Each ResNet block is provided with batch normalization, ReLU activations (for info, Brownlee [2019]) are used for all layers except for the output neurons, which are associated with a linear activation function (for info, Z^2 Little [2020]).

3.2.3 Performance

In the original paper, ProximityNet is trained using the Mean Absolute Error (MAE) loss function with the Adaptive Moment Estimation (ADAM) optimizer (for into, Kingma and Ba [2014]) and a base learning rate of 0.001, progressively reduced on validation loss plateaus that last more than 5 epochs. A maximum of 200 epochs are run in total, but with an early stopping policy with a patience of 10 epochs on the validation loss.

Performance are evaluated both quantitatively and qualitatively, which are both carried out on the end-to-end model rather than the mediated one here considered. However, as explained in section 2.3.2, both obtains similar results so we can consider following details to be valid also for the mediated approach.

For quantitative evaluation, the chosen metric is R^2[4], which has an interval of $[-\inf, 1]$ where 1 represents the optimality.

The author also conducts an experiment about the minimum cardinality of the dataset for obtaining acceptable performance. Results are available in figure 3.11, directly taken from the paper. As shown, decent performance requires at least 5'000 samples and keep improving as their number increases.

Specifically, predictions seem more accurate for variables Z and W with a R^2 score of 0.82 and 0.88, respectively. Different the findings for X and Y which only reach a R^2 of 0.59 and 0.57, respectively.

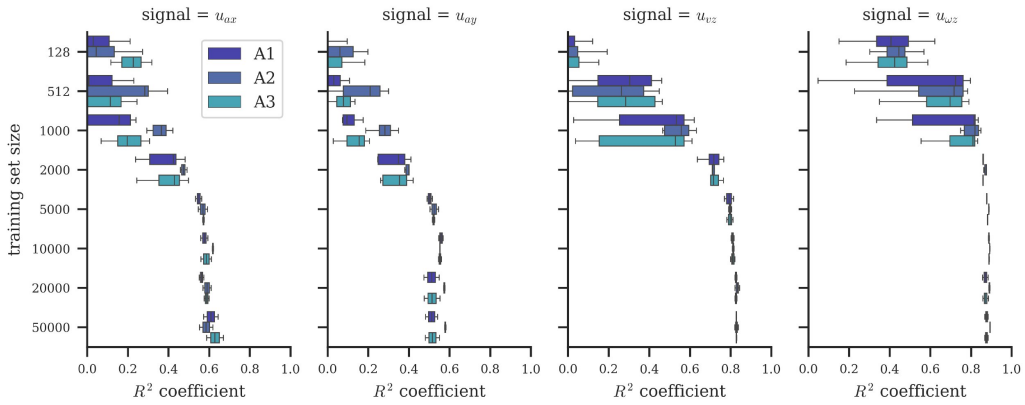


Figure 3.11: ProximityNet R^2 results from Mantegazza et al. [2019]

The previous considerations on the variables are confirmed by the qualitative evaluation, obtained by comparing ground truth and predictions during a short simulation. Figure 3.12[5] shows that X and Y predictions are considerably worse

[4] R^2 interpretation will be explained in the evaluation chapter 6

[5] A1, A2 and A3 in the chart stands for different models, but they achieve same results anyway

than results achieved by Z and W, when compared with the ground truth.

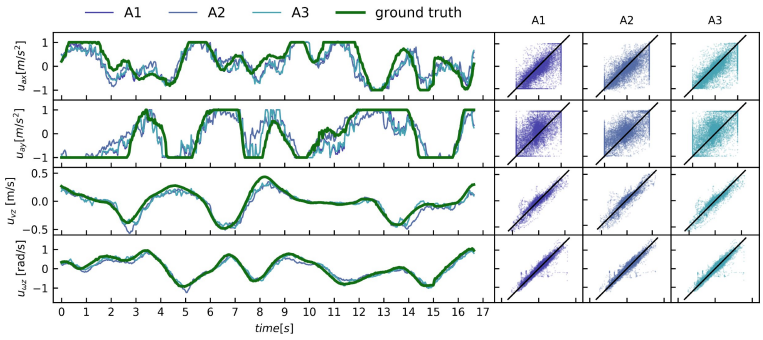


Figure 3.12: ProximityNet qualitative GT vs prediction results from Mantegazza et al. [2019]

3.2.4 Generalization

Even though the ProximityNet achieved quite good performance on the test set, its behavior must be proven on the real drone to certify the model usability.

Mantegazza et al. [2019] reports experiments conducted inside the arena by flying the drone without the MoCap system, only relying on the learned model for computing user's pose. The outcome is incredibly good, with the drone actually performing its task without many issues. Figure 3.13 presents trajectories followed by the drone during five consecutive runs (with two different models) in which the quadrotor had to face a user initially rotated by 90 degrees. Although the paths are sometimes different from what designed by the omniscient controller (the ground truth), they are still reasonable and flying capabilities in the arena seem very promising.

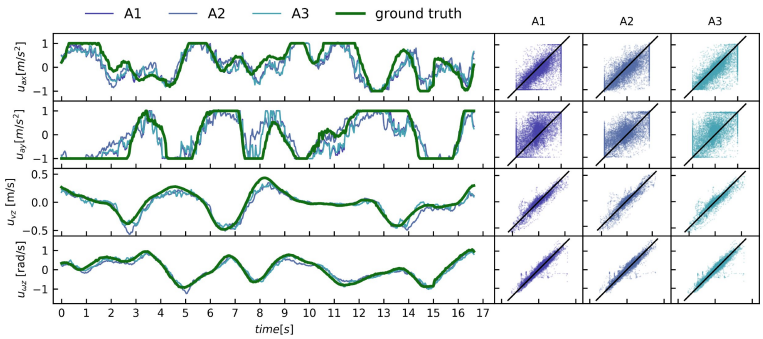


Figure 3.13: ProximityNet trajectories, from Mantegazza et al. [2019], for positioning in front of the user initially rotated by 90 degrees

Finally, we consider model performance in unknown environments, possibly outdoor. The official paper does not talk about the topic, but direct contacts with the author suggested that flying performance outside of the drone arena were not consistent with model behavior inside the environment it already knows. Accordingly to this finding, it seems that the model is not able to generalize the task when outside of the arena.

The goal of our work is to explore ways of improvement, which aim to generalize the model to make it able to theoretically predict the user's pose in any other unknown scenario. Next chapters firstly try to understand main issues and limitation of the model, then provide a possible solution.

3.3 Development

Finally, this section presents tools and software used to conduct our research for improving the existing system from a machine learning point of view, according to the objective of the thesis.

3.3.1 Tools

The entire source code is written in Python 3. First experiments were carried out with Jupyter Notebooks via Google Colab on a GPU-accelerated runtime, while the final code is provided as classic Python scripts to be executable on a custom machine.

For debugging a Windows 10 laptop equipped with an NVIDIA GeForce GTX 950M graphic card has been used, while actual training has been performed on a dedicated Ubuntu 18.04 workstation available at IDSIA mounting four NVIDIA GeForce RTX 2080 Ti [6].

3.3.2 Frameworks

The original work from Mantegazza et al. [2019] is written in Python and based on TensorFlow 1 and Keras. These libraries have been kept, but our project uses their updated versions for ease of use. Other intensively used frameworks are listed below.

`Numpy` Largely used in the whole project for computation on arrays. Numpy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays.

[6] multiple available GPUs are used for single-GPU computing, not simultaneously

`Pickle` Mainly used for saving and restoring Numpy arrays. The pickle module implements binary protocols for serializing and de-serializing a Python object structure.

`Matplotlib` First choice for building charts, visualize images or any kind of figure. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Its `pyplot` module is inspired by MATLAB.

`OpenCV` Mainly used for efficient image and video manipulation together with Matplotlib. OpenCV is an open-source library that includes several hundreds of computer vision algorithms.

`TensorFlow 2` All the project strongly relies on TensorFlow (TF) from start to end: network interpretation, person masking, training and quantitative evaluation. Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. In version 2, it introduces a lot of comforts for easier development with a less steep learning curve.

`Keras` Used for defining the network architecture, training and evaluating the model. Keras is the high-level API of TensorFlow 2: an approchable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

`TensorBoard` Used with TensorFlow to precisely profile data generator performance for optimizing training time. TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics, visualizing the model graph, and much more.

`Sklearn` Only used for automatically compute some evaluation metrics, Sklearn is a simple and efficient tools for predictive data analysis reusable in various contexts built on NumPy, SciPy, and Matplotlib.

`tf-keras-vis` Used for applying GradCAM and other interpretability techniques. Open-source library for network interpretation, available on GitHub thanks to Kubota [2020]. Derived from the original keras-vis (Google [2020]) high-level toolkit for visualizing and debugging your trained keras neural net models.

`akTwelve Mask_RCNN` Used for human detection and segmentation in background replacement. Open-source implementation of Mask R-CNN on Python 3, Keras, and TensorFlow available on GitHub thanks to Kelly [2020]. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Feature Pyramid Network (FPN) and a ResNet101 backbone.

Chapter 4

Initial Experiments & Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4.1 Model Interpretation

4.1.1 Regression to Classification

4.1.2 Loss Functions

4.1.3 Visual Results

4.1.4 Proposed Approach

4.2 Person Masking

4.2.1 Canny Edge Detection

4.2.2 Grabcut

4.2.3 YOLO

4.2.4 Face & Head Detection

4.2.5 Mask R-CNN

Chapter 5

Implementation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Chapter 6

Evaluation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Chapter 7

Conclusion

7.1 Final Thoughts

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

7.2 Future Works

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 8

Latex features

```
ACCURACY     10-FOLD CROSS VALIDATION:     0.8290 (std dev 0.005217)
PRECISION    10-FOLD CROSS VALIDATION:     0.8371 (std dev 0.009706)
RECALL       10-FOLD CROSS VALIDATION:     0.8176 (std dev 0.008412)
F1           10-FOLD CROSS VALIDATION:     0.8273 (std dev 0.004608)
```

- prodotti e utenti nella stessa rete, collegati attraverso le recensioni
- rete di soli prodotti, collegati tramite similarità
- rete di soli utenti, collegati tramite similarità

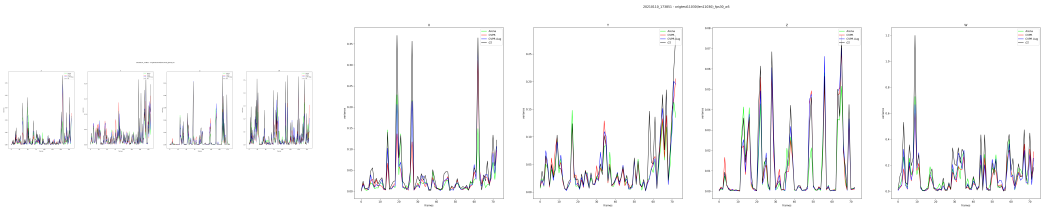


Figure 8.1: current caption

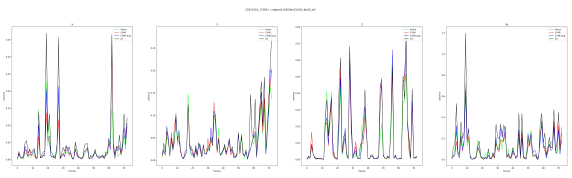


Figure 8.2: current caption [?].

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.

Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

```
def tfdata_generator(files, input_size, batch_size, backgrounds,
                     bg_smoothmask, aug_prob = 0, noises = [],
                     prefetch = True, parallelize = True,
                     deterministic = False, cache = False, repeat = 1) :

    map_parallel  =  tf.data.experimental.AUTOTUNE if parallelize else
                                    None
    backgrounds  =  tf.convert_to_tensor(backgrounds) # saves time
                                   during training
    noises  =  tf.convert_to_tensor(noises) # saves time during
                                   training

    gen  =  tf.data.Dataset.from_tensor_slices(files)
    gen  =  gen.map(lambda filename :  map_parse_input(filename,
                                   input_size), map_parallel,
                                   deterministic)
```

Listing 8.1: Protocol used by the manual controller to decide, for each robot, the message to transmit and the colour.

Table 8.1: Schema originale del dataset

Campo	Descrizione
reviewerID	ID utente
reviewerName	Nome utente
asin	ID prodotto

`Numpy` Good library

`Keras` Very good for ML

Appendix A

Extra Figures

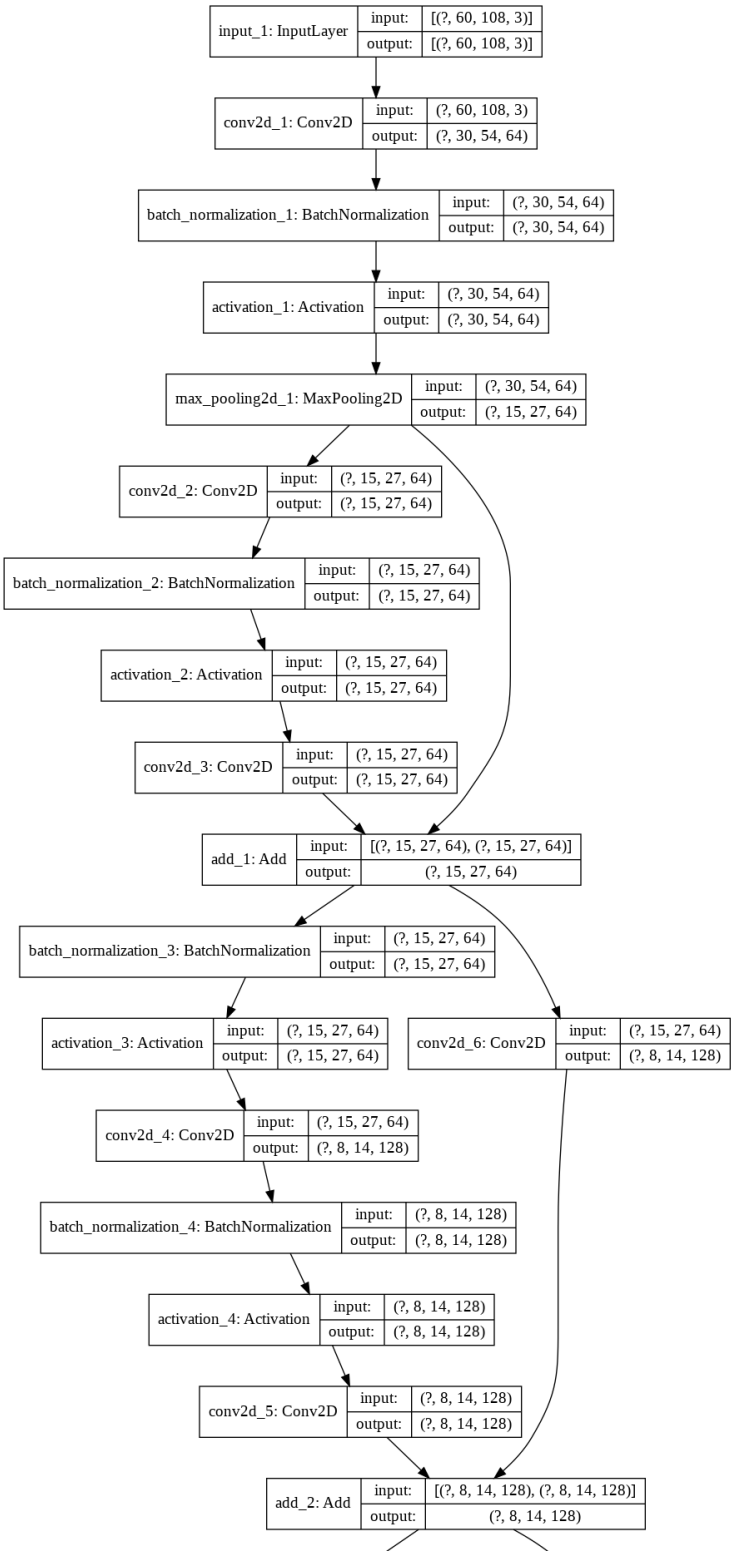


Figure A.1: ProximityNet complete architecture (part 1, from input to layer 18)

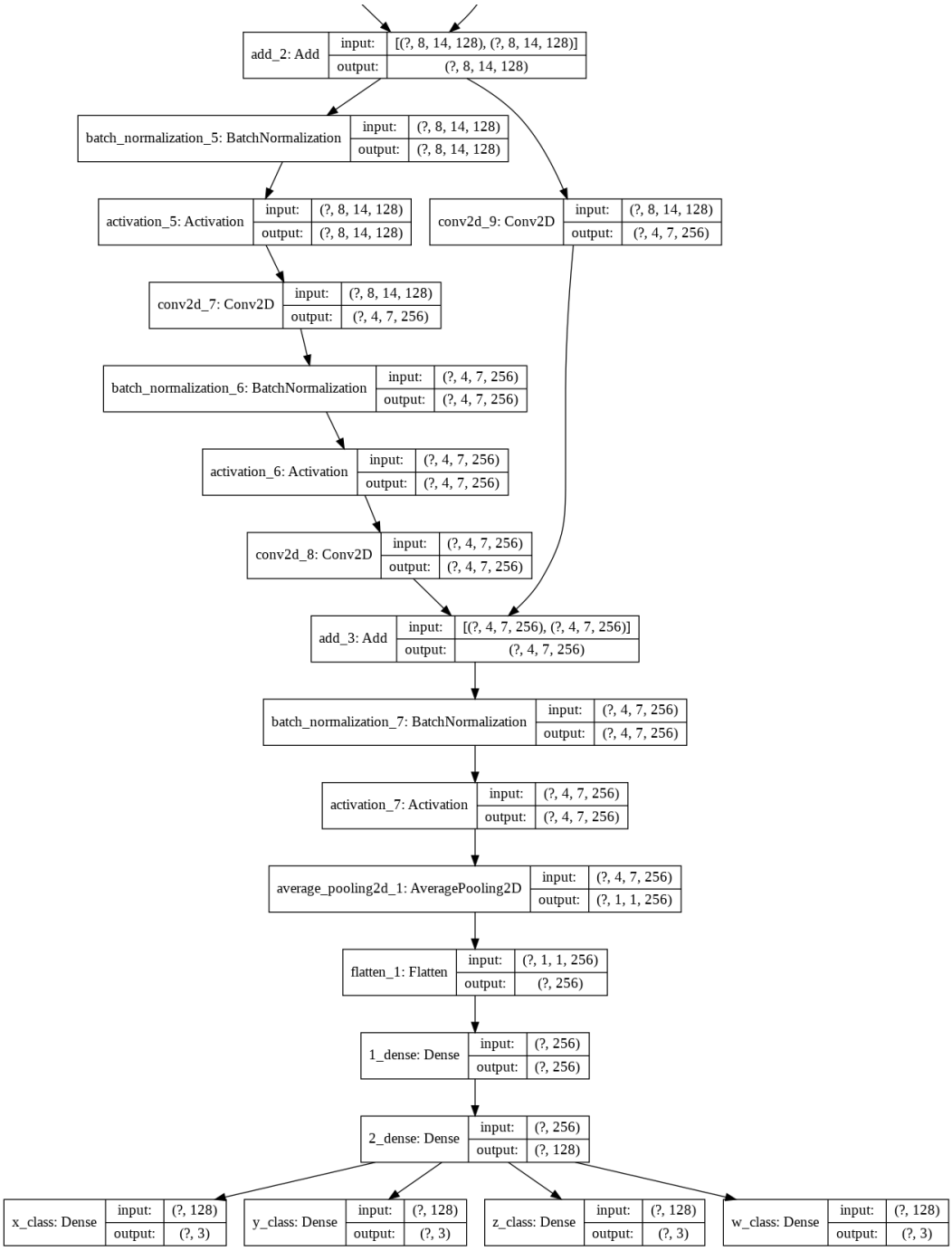


Figure A.2: ProximityNet complete architecture (part 2, from layer 18 to outputs)

Appendix B

Acronyms

ADAM Adaptive Moment Estimation

CNN Convolutional Neural Network

DoF degrees of freedom

FAA United States Federal Aviation Administration

FOV field of view

FPS frames per second

IDSIA Istituto Dalle Molle di Studi sull'Intelligenza Artificiale

IR infrared

MAE Mean Absolute Error

MoCap motion capture

MP megapixel

R^2 R Squared

ResNet Residual Neural Network

ROS Robot Operating System

wrt with respect to

Bibliography

Waleed Abdulla. Mask r-cnn for object detection and segmentation. https://github.com/matterport/Mask_RCNN, 2019.

Jason Brownlee. A gentle introduction to the rectified linear unit (relu), 2019. URL https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks.

Gazebo. Robot simulation made easy. URL http://gazebosim.org/.

Google. keras-vis is a high-level toolkit for visualizing and debugging your trained keras neural net models. https://github.com/raghakot/keras-vis, 2020.

Ahmed Hussein, Mohamed Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50, 04 2017. doi: 10.1145/3054912.

Adam Kelly. Mask r-cnn in tensorflow 2. https://github.com/akTwelve/Mask_RCNN, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Yasuhiro Kubota. tf-keras-vis is a visualization toolkit for debugging tf.keras models in tensorflow2.0+. https://github.com/keisen/tf-keras-vis, 2020.

Antonio Loquercio, Ana Isabel Maqueda, Carlos R. Del Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 2018. doi: 10.1109/lra.2018.2795643.

Dario Mantegazza. Defning a new controller for a drone with user partial pose estimation, 2018.

Dario Mantegazza. Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches. https://github.com/idsia-robotics/proximity-quadrotor-learning, 2020.

Dario Mantegazza, Jérôme Guzzi, Luca M. Gambardella, and Alessandro Giusti. Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches, 2019.

OptiTrack Website. Professional motion capture systems for robotics. URL `https://optitrack.com/applications/robotics/`.

D. Palossi, F. Conti, and L. Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 604–611, May 2019a. doi: 10.1109/DCOSS.2019.00111.

D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini. A 64mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 2019b. ISSN 2327-4662. doi: 10.1109/JIOT.2019.2917066.

Parrot Documentation. Bebop drone 2 full specifications, 2015. URL `https://s.eet.eu/icmedia/mmo_35935326_1491991400_5839_16054.pdf`.

ROS Website. Professional motion capture systems for robotics. URL `https://www.ros.org/`.

Rviz. General purpose 3d visualization tool for ros. URL `http://wiki.ros.org/rviz`.

D. Tezza and M. Andujar. The state-of-the-art of human–drone interaction: A survey. *IEEE Access*, 7:167438–167454, 2019. doi: 10.1109/ACCESS.2019.2953900. URL `https://ieeexplore.ieee.org/document/8903295`.

Z^2 Little. Activation functions (linear/non-linear) in deep learning, 2020. URL `https://xzz201920.medium.com/activation-functions-linear-non-linear-in-deep-learning-relu-sigmoid-softmax-swish-leaky-relu-a6333be712ea`.

Nicky Zimmerman. Embedded implementation of reactive end-to-end visual controller for nano-drones, 2020.