Master's Thesis EX028/2018

Autonomous Drone for Liferaft Detection

Alexander Börjesson & Niclas Hillborg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden, 2018

Autonomous Drone for Liferaft Detection
Alexander Börjesson & Niclas Hillborg

Supervisor: Ph.D. Oskar Wigström, Chalmers - Department of Electrical Engineering
Examiner: Associate Professor Petter Falkman, Chalmers - Department of Electrical Engineering

Master's Thesis EX028/2018
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone: +46 31 772 1000

Abstract

Using unmanned aerial vehicles (UAVs) is increasing in a lot of fields and the fact that they can be made to operate autonomously is valuable. Utilizing drones for maritime search and rescue (SAR) missions could improve safety for rescue personnel and increase the efficiency of getting distressed people in safety. Placing drones on larger crane equipped ships could enable usage of drones, to remotely attached a line to a liferaft and utilize the ship's crane in order to lift the liferaft onboard. This thesis presents a foundation for deploying a DJI Phantom 4 PRO drone configured as an autonomous drone for attaching a line to the liferaft and utilize image detecting to detect the liferaft via the drone's live video feed using a deep learning algorithm. A Convolutional Neural Network (CNN) has been trained and tuned for predicting if the liferaft is presence in an image, a ground station containing a computer running the CNN and drone maneuver program and an Android communication application has been developed, making the drone fully autonomous. The results obtained prove that the proposed system design used in this thesis transform the Phantom 4 PRO into an autonomous drone. Utilizing the live video feed of the autonomous drone enables the designed system to locate and execute a flight path towards the detected liferaft. However, the proposed designed system has only been tested in simplified scenarios and more data for the CNN are needed to make the prediction of the raft more robust. Investigations if the Phantom 4 PRO can carry the required load is needed and a better way of tracking the liferaft must be developed for a more versatile solution.

Keywords: Autonomous drone, drones for SAR missions, remotely connecting life rafts, Convolutional Neural Network.

Acknowledgments

A big thank you to our supervisor Oskar Wigström for impeccable support and guidance with various problems and ideas during this project.

We would also like to thank Fredrik Falkman at SSRS for great contact support and for lending us a huge liferaft.

We would like to thank our examiner Petter Falkman as well for great support.

Alexander Börjesson &
Niclas Hillborg
Gothenburg, June 21, 2018

Contents

List of Acronyms

CNN	convolutional neural network
SAR	search and rescue
UAV	unmanned aerial vehicles
SSRS	swedish sea rescue society
GPS	global positioning system
NN	neural network
RNN	recurrent neural network
RC	radio controller
PD	Proportional-Derivative
LSVRC2014	large scale visual recognition challenge 2014
ReLu	rectified linear unit

List of Figures

List of Symbols

x_i	binary inputs for the perceptron
w_i	weights to compute the binary output
σ	sigmoid function
$C(w, b)$	gradient descent cost function
a	vector of outputs
η	learning rate
φ_n	the n^{th} point's latitude
λ_n	the n^{th} point's longitude
R	mean radius of earth (6.371km)
d	distance to target in meters
θ_b	bearing to target point clockwise from north
m	mass of the drone
$\ddot{x}$	acceleration in all three dimensions
g_z	downward pointing gravity vector
$R(\phi, \psi, \theta)$	rotation matrix
ϕ	rotation angle of X-axis, i.e. roll
ψ	rotation angle of Y-axis, i.e. pitch
θ	rotation angle of Z-axis, i.e. yaw
$\sum_{i=1}^{4} T_i$	total upward thrust generated by the motors
b	propellers drag constant
ω_{m_i}	angular velocity of each motor
J	diagonal form inertia matrix consisting of each axis inertia
ω	angular velocity in three dimensions
τ	torque around each axis
d	length from the center of the drone to each motor
k	the propellers lift constant
T	thrust generated by each motor
β	difference between bearing to target and actual yaw angle of drone
v	forward speed of the drone

1 Introduction

During a large-scale maritime rescue operation, it is a great challenge to get distressed people in safety. The boats that the sea rescue team are using are usually not built to safely hold more than 10-20 people. Using helicopters are not effective either due to limited space onboard the helicopters and the time it takes to lift people up. According to Fredrik Falkman at the Swedish Sea Rescue Society (SSRS), it takes approximately 7 minutes per person to be winched up to the helicopter. Could Unmanned Aerial Vehicles (UAVs) be utilized to solve this challenge?

Today, UAVs have become increasingly popular in a broad field of applications, like construction inspections, surveillance, delivery in hard to access areas, commercial or recreational photography, and search and rescue (SAR) operations [1]. During SAR operations, using the UAVs instead of rescue personnel and vehicles could revolutionize the operations by improving efficiency in localizing and tracking capabilities. Implementing the UAVs to operate as SAR vehicles would also result in high repeatability to redo similar rescue operation scenarios in the future. Utilizing a cloud to share information regarding various rescue, the UAV's could be linked to the cloud to share operations and procedures that each UAV has faced, which enables each and every single UAV to learn how to interpret challenging rescue scenarios. SAR missions are often characterized by a number of constraints such as time, human losses, operational environments. These constraints are critical and hard to minimize with considerations to the methods used today [2]. Using UAVs can provide serious support to the already proven and used methods used for SAR operations. In typical SAR scenarios, the UAV will be deployed in an area of interest, perform sensory operations to collect data of the presence of victims and report collected data to a remote ground station or rescue team.

The UAVs are fast, can operate autonomously and thereby perform operations hard to execute by human operators at low cost and risk. The efficiency of using the UAVs in rescue operations has been proven in 2005, where UAVs were used in the aftermath of Hurricane Katrina to search for trapped survivors in damaged areas [3].

To solve the challenge of the large-scale maritime rescue operation, an effective alternative would be to utilize nearby cargo ships since they can hold a large number of people. The problem with this solution is to get the distressed people off the liferaft and onboard the ship. In order to bring the people safely onboard, an effective way would be to lift the liferaft with the people still inside. However, this would require a way to connect a line to the liferaft which could be done utilizing a UAV. If a UAV would be stationed at every ship with the possibility to lift a 50 person liferaft, the rescue operation would be more effective.

1.1 Objective

In order to make the large-scale maritime rescue operation more efficient, SSRS sees great potential in position UAVs on ships with a crane powerful enough to lift a 50 person liferaft and utilize the drone for connecting a line to the liferaft. However, no existing solution is available on the market today and therefore, SSRS reached out to Chalmers University of Technology for help in developing a possible solution. To achieve SSRS vision, an autonomous drone would be beneficial since no training of the ship's personnel would be needed. It would also be a hard maneuver to fly the drone in the correct way to successfully connect the line and an autonomous drone would increase the performance of this operation. The UAV used in this thesis is the DJI Phantom 4 PRO drone [4], which is a quadcopter drone and was chosen since it has a good camera, long battery life, and can become autonomous by using the DJI Mobile SDK [5]. A ground station will be constructed which handles calculation of control signals for desired motions and a deep learning algorithm will be used to distinguish a liferaft in the drone's camera live feed. The complete flight mission will be very simplified and only used to show that the Phantom 4 PRO can be used as an autonomous drone and that the ground station constructed can be used for further development.

This thesis's objective is to lay the foundation for using a DJI Drone as an autonomous drone for connecting a line to a liferaft and to show that the drone can distinguish the liferaft in its live feed using a deep learning algorithm.

1.2 Project outline

The objective, to utilize the Phantom 4 PRO as an autonomous drone for connecting the line to the liferaft and to find the liferaft in its live feed, requires a couple of sub-tasks to be solved. One is to make sure that the drone can distinguish a liferaft from other boats and objects. This is solved using a Convolutional Neural Network (CNN) to classify objects. Once the liferaft is identified, the drone needs to fly to it. To do this, distance and heading are calculated. By making sure the liferaft is centered, it is possible to calculate the distance to the target by using the known height above water and in what angle relative to the horizontal plane the camera is pitched. Since the drone that is used is a DJI Phantom 4 PRO, it is not possible to program the onboard computer to do these tasks and instead, a ground station must be constructed. This ground station consists of a computer running the classifier and control strategy and an Android phone connected to the drones radio controller (RC). The phone is running an application which handles communication between the computer and the drone. Due to time constraints, the drone will be flying without a line and the final flight mission presented in this thesis is only to prove that the drone can distinguish the liferaft and fly autonomously to the raft and back home.

2 Hardware

In this section, the main hardware used in the thesis is presented, which is the drone used, a game capture card used to get the live feed from the drone to the computer and the life raft which the complete system will be tested with.

2.1 DJI Phantom 4 PRO drone

The drone used is the DJI Phantom 4 PRO. It is developed by a Chinese company called DJI and no modifications of the drone have been made. This drone is very popular in the recreational photography and filming community because of its compact size, good camera, long battery life and user-friendliness.

Figure 1: Picture of the DJI Phantom 4 PRO drone.

The Phantom 4 PRO is equipped with a gimbal stabilized 1-inch 20MP 4K camera, vision system in the front, back and underneath plus infrared sensing systems on the sides for object avoidance as well as ultrasonic sensors for height measurements. It is also equipped with a GPS and magnetometer sensor which enables the drone to know its position and hover in place in a stable manner.

In order to control the drone, an RC is needed which communicates with the drone on 2.4GHz or 5.8GHz depending on the surrounding. If the live feed from the drone is desirable, a smartphone with a compatible application must be connected to the RC through an USB-cable. DJI has developed their own application which gives pilots a lot of modes, settings, and options to alter and to see the live feed from the drone. This app is available through either App Store or Google Play. However, since the goal of this thesis is to make the drone autonomous, an app specific for this purpose had to be developed.

Estimated flight time for the Phantom 4 PRO is 30 minutes, top speed in Sports Mode is $72km/h$ and flight range is $7km$ according to DJI. Even if DJI specifies that the range of the drone is $7km$, it will not be utilized since the drone laws in Sweden [6] require a permit to fly without visual contact with the drone and therefore the drone will not be flown further away than $200m$.

2.2 Razer Ripsaw game capture card

The live feed from the camera is needed in order for the classification function to work and for continuous update of the rafts position. This could be done wireless but to minimize delays in the video transmission, a Razor Ripsaw game capture card was used [7]. The Ripsaw is developed for gamers who want to record video of their games, for example, live streaming to the internet and the low latency made it suitable to use for the work done in this thesis.

On the Phantom 4 PRO's RC, an HDMI Output Module [8] was mounted in order to get the camera live feed from the drone. Then the Ripsaw was connected to the HDMI output port on the HDMI Output Module and to the computer with a USB-cable. The reason for not connecting the HDMI output port on the RC directly to the computer is because the computers used in this thesis only supports HDMI output and not input, which is what is needed in this case. By using the Ripsaw, this problem was solved and the live feed could be seen on the computer and used in the computer program for classification and feedback from the image.

2.3 Viking 25 person liferaft

To be able to test the liferaft detection, a Viking 25 person liferaft [9] was borrowed from the SSRS. It is a common liferaft and is certified for being lifted with 25 persons in it. Since the long-term goal of this project is to be able to use the drone to connect a line to the liferaft and then use a winch to lift the raft, it was ideal to have access to this liferaft. However, due to the large size of the raft and lack of access to a boat to tow the raft with, no tests have been done with the raft in the water.

Figure 2: Picture of the Viking 25 person liferaft.

3 Theory

This section presents theory used in this thesis. It starts with DJI Mobile SDK API used to enable autonomous flight of the drone, then how communication between the computer and the phone is handled, continues with theory about Neural Network, then Haversine formula for distance, bearing and position calculations, followed by equations for the drone, how to maneuver the drone depending on the difference between yaw angle and bearing, and lastly what control strategies that has been used.

3.1 DJI Mobile SDK API

In order to get access to the functions in the DJI Phantom 4 PRO drone, a mobile application connected to the drone's RC is needed [5]. DJI has developed a library for both Android and iOS devices which give you access to functions like reading drone states, start or stop recording video, get the live feed from the drone's camera or turn the motors on, just to name a few. On DJI's website, there are tutorials on how to implement different types of functions for your application [10] and this has been the stepping stone to learn how to communicate with the drone in this thesis. There is also a wide variety of projects of different kinds that can be found on GitHub which has been studied to learn the DJI Mobile SDK API [11, 12, 13]. Only an Android application will be developed in this project due to the very limited knowledge of Swift programming, which is required to develop an iOS application.

Since it is desirable to make the drone autonomous, a way to skip the drone's radio controller's input is needed when maneuvering the drone. The way to do this is to use DJI function library Virtual Sticks. As DJI present it in their Simulator tutorial [14], the Virtual Sticks can be used to enable a smartphone as the controller instead of the radio controller. In this application, the virtual sticks are designed as touch buttons that can be moved by sliding a finger over the screen. The area, in which the virtual touch stick is allowed to move within, is defined as a square box with an XY-coordinate system. Origin of the coordinate system is in the center of the box and the X and Y value is input signals to the drone. Since the input signals to the Virtual Sticks function in this tutorial is a coordinate value in either the X or Y direction, depending on which motion is intended, the touch screen part can instead be substituted for a value sent by the computer. This will allow for a control program running on the computer to calculate the suitable input signal for the desired motion.

One limitation of using the Virtual sticks is that they require a good Global Positioning System (GPS) connection to work properly, which is only possible to get outside. Without the proper GPS connection, the drone's motors will turn off if a negative throttle signal is sent to the drone and this will happen regardless if the drone is flying or not. Another limitation is that if a self-developed mobile application is used for the drone, DJI has put limitations on how far and high you are allowed to fly. The distance is limited to 50 meters away from the RC and maximum altitude is 30 meters. When these limits are reached, the drone will not fly any further and just hover. A way to circumvent these limitations has not been found and requesting DJI to disabled them has not been successful.

3.2 Communication between computer and phone

In order for the drone to fly in a satisfying way, data needs to be sent back and forth between the drone and the computer. As mentioned earlier, a phone which is connected to the RC with an USB-cable is needed. The communication is done through a socket connection [15] where the computer is the server and the phone is the client. In order to avoid disturbance and unwanted behavior, the communication is done on a private WiFi network where only the computer and the phone is connected. Ports used for the communication is predefined in the computer program and in the app. They are defined as the phone will always send its data on port 9000 to the computer and receive data on port 9001 to make sure that the sent and received data is not mixed. These ports are not chosen for any specific reason more than that they are available and not used for anything else in our application. In order to have a successful communication, the phone, as the client, must also know the IP-address of the computer, the server, which is entered in the app by the operator.

The communications work in such a way that the server, i.e. computer, is listening on port 9000 for any incoming request for connection. Once the client, i.e. phone, makes a connection request on port 9000 it is either accepted or denied. If the connection request is accepted, a socket connection is established and data can be sent. The same procedure is done when the server will send data to the client but instead, the connection is established over port 9001. See figure 3 for a visual representation.

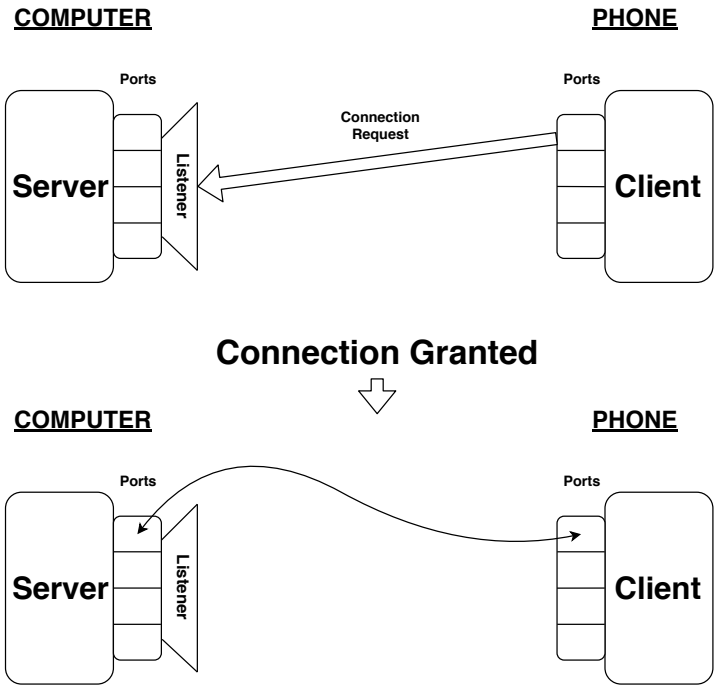

Figure 3: Socket communication between computer and phone.

3.3 Neural Network

A neural network (NN) is a model of artificial neurons stacked in layer upon layers and together they form a network with an input layer, an output layer, and hidden layers in between. By designing specific network architectures, the networks can be deployed for specific tasks, such as, for convenience, image classification. Due to the high accuracy and efficiency of these networks, the usage has increased and this makes them useful for a number of various applications, for example, in banking where they can be used to make investment decisions [16].

The model of an artificial neuron contains weights and bias and these parameters are optimized by an optimization algorithm in order to find a set of parameters. These parameters are used to minimize the error between the input data and the output prediction. The layers of an NN are interconnected and generally, the number of hidden layers is large. For illustrating purposes, a figure of a simplified NN is described in figure 4 where the NN model consists of an input layer with five neurons, the hidden layer of seven neurons, and the output layer of three neurons. This type of network is called *fully connected NN*.

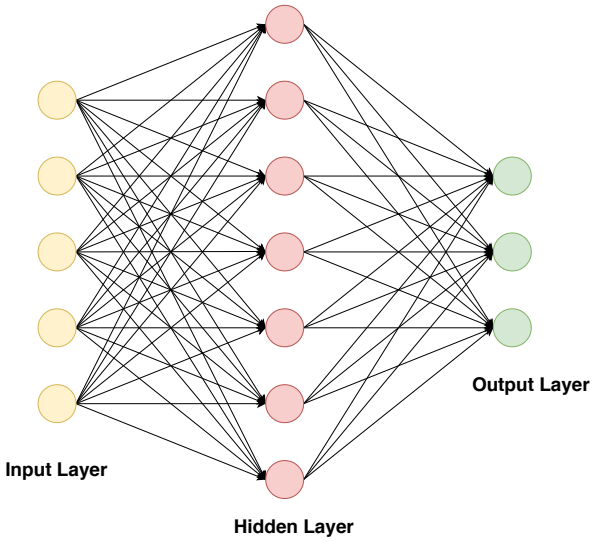

Figure 4: Representation of a basic neural network architecture structure

The input layer is designed to match the dimensions of the input data, for instance, if an image with dimensions of $255 \times 255 \times 1$, the input layer dimensions have to match, resulting in 255×255 neurons located in the input layer. Depending on the application, for convenience consider an image classification network which predicts the input image and outputs prediction in the form of an array $N \times 1$, the number of classes which the network is trained to recognize can vary where the number of classes are N. Designing the hidden layers are not a specifically straightforward method and therefore are researchers using heuristics to determine a trade-off between the number of hidden layers and the training elapse time [17]. The simplified NN represented in figure 4 is a

feedforward NN but there also exists *feedback* networks called *recurrent neural networks* which is more practical due to there ability to solve complex tasks more efficiently than the standard *feedforward* networks [17].

3.3.1 Convolutional neural network

For image classification problems, one specific NN architecture called *convolutional neural networks* (CNN) functions exceptional well over other networks. The CNN are faster to train and their structure enables the network to solve complex tasks such as image recognition. Instead of the traditional NN concept of connected layers, the CNN utilizes *local receptive fields* which takes square-formed neuron input and connects the small region to one neuron in the first hidden layer. The region is then swept over all of the input layer neurons and connected to a different hidden layer neuron. Consider figure 5 which consists of 12×12 neuron input connected to a hidden layer neuron where each of the connections from the input layer neurons contains one weight and the hidden layer neuron holds a bias.

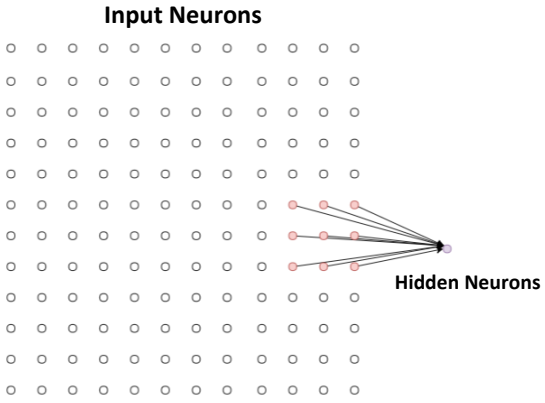

Figure 5: Representation of connection between localized input region and hidden neuron, a 3×3 localized region is connected to a hidden neuron.

As the local receptive field is swept across the input layer neurons new connections are maid to a different hidden layer neuron and these weights and biases are shared between each of the hidden layer neurons. This concept is referred to as *shared weights and biases* [17], which reduces the amount of parameters since if the parameters were not shared, the total amount would be extremely large[18]. The concept enables the CNN to create *feature maps* which are utilized to recognize specific features of the input data. This property enables the CNN to adapt to translation invariance in images [17], i.e. the location of the recognized object in the image has no consequence. The CNN usually consists of several feature maps to recognize several features of the input image, which is one key property of the CNN's performance in image classification. For instance, one feature map could be to recognize the color red in combination with the color blue, which theoretically could give an indication of the presences of a liferaft located somewhere in the input image. Due to the architecture of CNN, they are often referred to as *deep neural*

networks since they consists of multiple hidden layers, where each hidden layer holds different feature maps [17].

3.4 Haversine formula for distance, bearing and position calculations

If longitude and latitude of two points are known, it is possible to calculate distance and bearing between these points by using the Haversine formula [19, 20]. The Haversine formula gives firstly the distance between two points along a great circle as

$$\begin{aligned} a &= sin^2\left(\frac{\Delta\varphi}{2}\right) + cos\left(\varphi_1\right) \cdot cos\left(\varphi_2\right) \cdot sin^2\left(\frac{\Delta\lambda}{2}\right) \\ c &= 2 \cdot atan2\left(\sqrt{a}, \quad \sqrt{1-a}\right) \\ d &= R \cdot c \end{aligned} \tag{1}$$

where φ_n is the n^{th} point's latitude in radians, $\Delta\lambda$ (also in radians) is the difference in longitude between the points, R is the mean radius of the earth, which is 6.371km, and d is the calculated distance in meters. Secondly, the Haversine formula gives the bearing from the first point to the second as

$$\theta_b = atan2\left(sin\left(\Delta\lambda\right) \cdot cos\left(\varphi_2\right), \quad cos\left(\varphi_1\right) \cdot sin\left(\varphi_2\right) - sin\left(\varphi_1\right) \cdot cos\left(\varphi_2\right) \cdot cos\left(\Delta\lambda\right)\right) \quad . \tag{2}$$

Since both φ_n and λ_n is in radians, the calculated θ_b value will also be in radians. Equation 1 and 2 will be used when the drone travel from the liferaft to the home position and in the open-loop case when the drone travel between the calculated points.

If it is desirable to calculate the liferaft's position when the position of the drone, the drones bearing to the raft and the distance between the raft and drone is known, the following equations can be used [20]

$$\begin{aligned} \varphi_2 &= asin\left(sin\left(\varphi_1\right) \cdot cos\left(\frac{d}{R}\right) + cos\left(\varphi_1\right) \cdot sin\left(\frac{d}{R}\right) \cdot cos\left(\theta_b\right)\right) \\ \lambda_2 &= \lambda_1 + atan2\left(sin\left(\theta_b\right) \cdot sin\left(\frac{d}{R}\right) \cdot cos\left(\varphi_1\right), \quad cos\left(\frac{d}{R}\right) - sin\left(\varphi_1\right) \cdot sin\left(\varphi_2\right)\right) \end{aligned} \tag{3}$$

where θ_b is the bearing clockwise from north. This approach will be used in the Open-loop case where the drone calculates the rafts position once and then fly the calculated trajectory.

3.5 Motion equations for a quadcopter drone

The Phantom 4 PRO drone is a quadcopter drone, meaning it has four motors on separate arms, see figure 6 for an illustration. In order to understand dynamics of the drone and how it will move depending on different motor thrusts, equations 4 - 11 can be analyzed. The equations are mainly extracted from chapter 4.3 in Corke's book *Robotic, Vision and Control, The fundamental algorithm in MATLAB* [21] and modified to suit the Phantom 4 PRO. The simplified motion

equation of the drone is derived from Newton's second law as

$$m\ddot{x} = -g_z + R\left(\phi, \psi, \theta\right) \sum_{i=1}^{4} T_i \quad , \tag{4}$$

where m is the mass of the drone, $\ddot{x}$ is the acceleration in all three dimensions, g_z is the downward pointing gravity vector, $R(\phi, \psi, \theta)$ is the rotation matrix from equation 8, defined in the inertia frame where ϕ, ψ and θ represents the rotational angle of the X-, Y- and Z-axle. Equation 5 - 7 describes how the drone's axes are rotated in the global coordinate system. Combined, as in equation 8, they give the total rotation of the drone in the global coordinate system.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{5}$$

$$R_y = \begin{bmatrix} cos(\psi) & 0 & sin(\psi) \\ 0 & 1 & 0 \\ -sin(\psi) & 0 & cos(\psi) \end{bmatrix} \tag{6}$$

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$R(\phi, \psi, \theta) = R_x \cdot R_y \cdot R_z \tag{8}$$

$\sum_{i=1}^{4} T_i$ is the total upward thrust generated by the motors. This total thrust is expressed as

$$F_{tot} = \sum_{i=1}^{4} T_i = \sum_{i=1}^{4} b\omega_{m_i}^2 \quad , \tag{9}$$

where b is the propellers drag constant and ω_{m_i} is the angular velocity of each motor. In equation 10, the drones dynamics regarding the rotational accelerations is expressed. The drone's angular acceleration is expressed as

$$J\dot{\omega} = -\omega \times J\omega + \tau \quad , \tag{10}$$

where J is the diagonal form inertia matrix consisting of each axis inertia, ω is the angular velocity in three dimensions, which is not to be confused with ω_{m_i} in equation 9, and τ is the torque around each axis expressed as

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} \frac{d}{\sqrt{2}}(T_3 + T_4 - T_1 - T_2) \\ \frac{d}{\sqrt{2}}(T_2 + T_3 - T_1 - T_4) \\ \frac{k}{b}(T_1 + T_3 - T_2 - T_4) \end{bmatrix} \quad , \tag{11}$$

where d is the length from the center of the drone to each motor, k is the propellers lift constant and T is the thrust generated from each motor. Equation 4 - 11 would be used if simulations of the drone's motion was desirable, but is also useful for understanding how the drone moves

depending on how much thrust each motor gets.

By studying equation 11 and figure 6, it can be seen that in order to move the drone to the right, i.e. positive roll motion, motors M3 and M4 must get more thrust than M1 and M2. If it is desirable to go forward, i.e. positive pitch motion, M2 and M3 must get more thrust than M1 and M4. Finally, if a clockwise rotation, i.e. positive yaw motion, is to be done, M1 and M3 must get more thrust than M2 and M4.

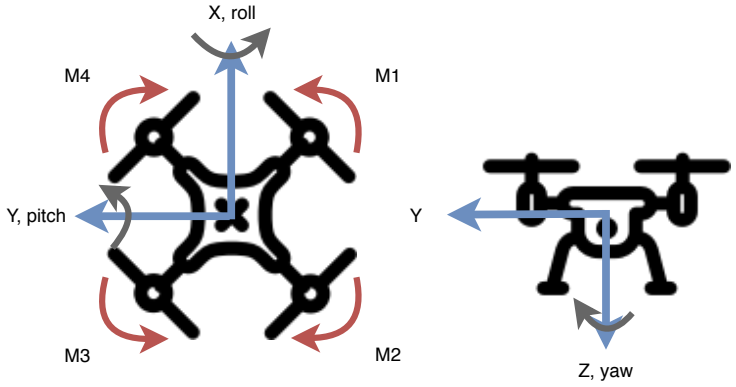

Figure 6: Defined coordinate system of the DJI Phantom 4 PRO drone.

3.6 Pitch and Roll output depending on Yaw angle and Bearing

Giving the drone the correct input on pitch and roll require consideration to the bearing to the target and actual yaw angle of the drone. Since the quadcopter drone's speed is not depending on how the yaw angle is with respect to desired flight direction, the yaw angle can be maneuvered to face in the flight direction while the drone flies towards the target. This is very effective when the drone is flying towards a known GPS coordinate, like when the drone is flying back to its home position and the drone is not yet facing the home position. In order to do this, a variable β was used which is the difference between the yaw angle and the bearing to the target,

$$\beta = bearing - yaw \quad . \tag{12}$$

The difference between the yaw and bearing angle, β, is then used to calculate how much the roll and pitch input is needed to fly towards the target,

$$\begin{aligned} pitch &= v \cdot cos\,(\beta) \\ roll &= v \cdot sin\,(\beta) \end{aligned} \quad , \tag{13}$$

where v is the drone's speed forward. This is also illustrated in figure 7 where the output from a physical joystick is shown. In this case, the bearing to target is more than 45 degrees from the yaw angle and therefore, more roll output than pitch output is needed to travel to the target. As the yaw angle is corrected towards the bearing to target, the β angle will decrees and more and more pitch output will be given.

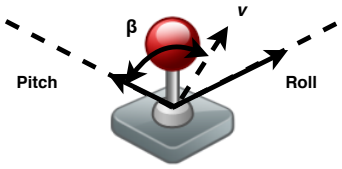

Figure 7: Illustration of how roll and pitch control signals are calculated depending on the difference between yaw angle and bearing.

3.7 Control strategy

To get the drone to fly autonomously, control of the drone's motion in upward, downward, sideways, yaw and gimbal pitch is needed in this thesis. Stabilizing the drone is not needed since it is handled by the onboard computer which is already tuned by DJI before delivery of the drone. The motion control needed can be handled by a couple of simple PD controllers which are tuned for each specific task.

3.7.1 PD regulator

A Proportional-Derivative (PD) regulator is a feedback control system with the goal to minimize the *error* between the reference signal ref and the output signal y [22]. In figure 8, an example of a PD control system is shown.

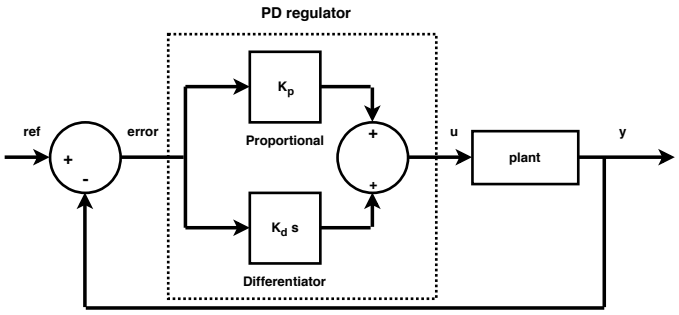

Figure 8: PD control system.

Reason for using the PD regulator is since all control of the drone's motion will use position or velocity as the reference and no integral part is needed to achieve a satisfactory behavior.

The parameters K_p and K_d in figure 8 are tuning parameters that can be tuned to achieve the desired behavior. If the plant model, in this thesis the drone, is well known, K_p and K_d can be theoretically calculated but that is not the case in this thesis. Therefore, the trial-and-error method is applied to obtain suitable parameter values for K_p and K_d.

4 Method

The Method section describes the parts solved to achieve the objective of getting the Phantom 4 PRO drone to fly autonomously and be able to distinguish a liferaft in its live feed.

4.1 Interface between Computer, Phone, and Drone

The communication between the computer, phone, and drone is set up as shown in figure 9, where the drone is connected to its RC wireless on 2.4GHz or 5.8GHz. The RC has an Android phone running the application shown in figure 11 connected and is connected to the computer via the Racer Ripsaw to get the drone's live feed to the computer. Communication between the computer and the phone is done on a private WiFi network through a router.

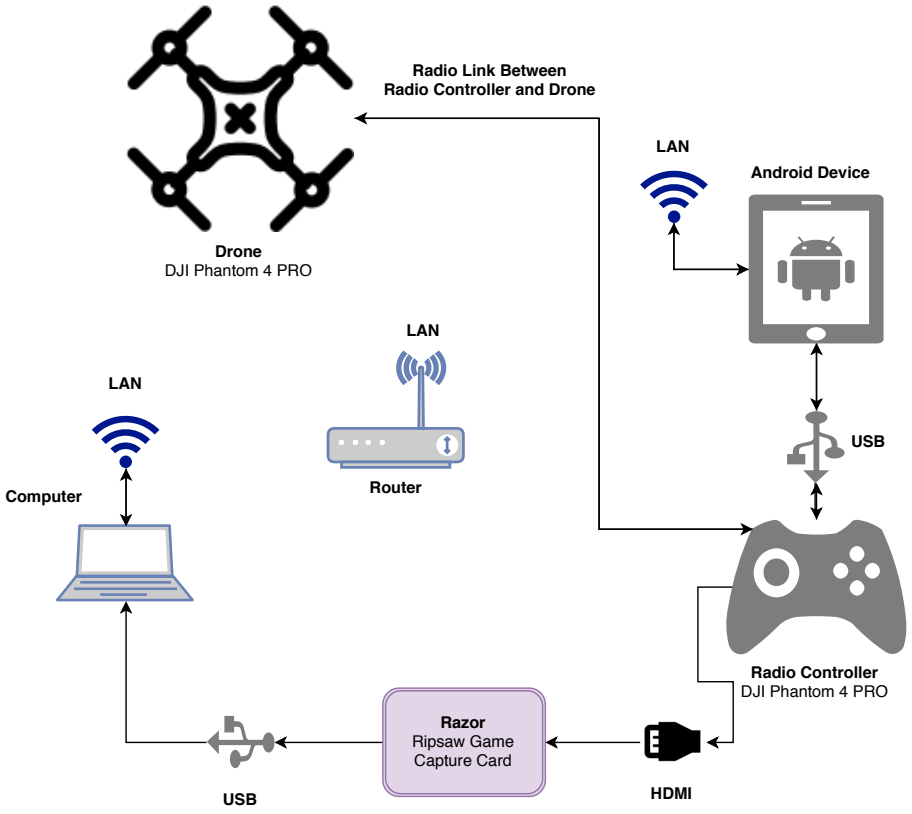

Figure 9: Visualization of all components and how they communicate with each other.

The scheme of the program to make the drone autonomous can be seen in figure 10 and the different steps are starting from the left:

1. Waiting for the operator to start the mission and to turn on the drone's motors.
2. The drone's states are collected.
3. The drone's states are sent to the computer.
4. The computer program does calculations based on the camera live feed image and the drone states received from the phone.
5. The computer sends the control signals for the motors and the gimbal pitch motor to the phone.
6. The phone receives the control signals and sends them to the drone.
7. A check to see if the operator has stopped the mission.
8. If the mission is not stopped, the loop continues from point 2.
9. If the mission is stopped, the operator gains control of the drone and can fly it with the RC or start the mission again.

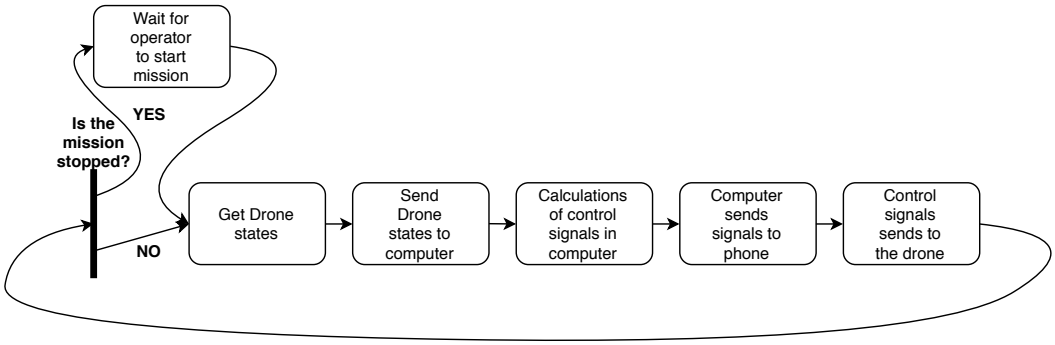

Figure 10: Program flow of the application.

To ensure that the program on the computer or phone does not get blocked while waiting for data, both programs have timeout functions which allow the program to continue after a set time. Since it is very important that the drone does not do any unwanted movements due to missed data from the computer, the phone has a shorter timeout time and if no data is received, the drone gets signals that will make it hover in place.

The features of the Android Application, as seen in figure 11, is to decide what IP-address to connect to (1), display battery percentage of the drone (2), display amount of connected satellites for the GPS (3), Start (4) or Stop (6) (changes text depending on if the mission is started or not) the mission, start or stop recording video (5), update the drone's states (6), display the relevant drone states like the latitude and longitude position, the height, the yaw angle relative to north, the gimbal pitch angle (7), display the motor signals sent to the drone (8), display the received signals from the computer (9) and to see the live feed from the camera (10).

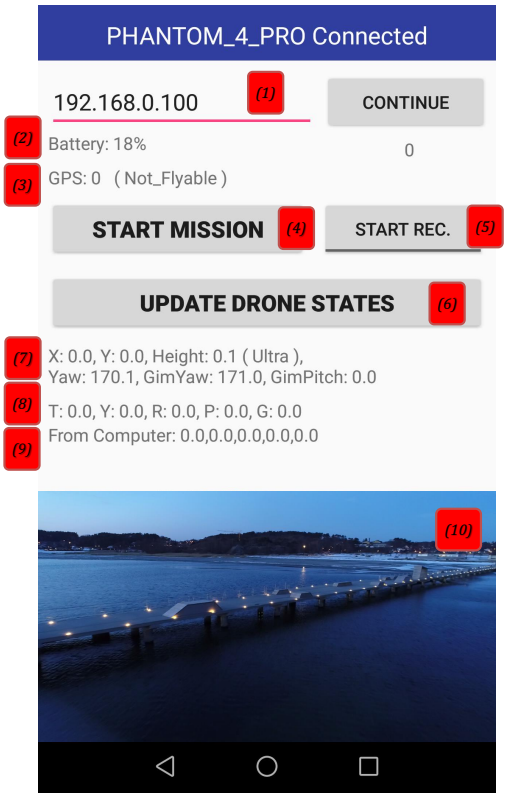

Figure 11: Android Application interface with marked features.

4.2 Convolutional Neural Network Design

Designing CNN nets is cumbersome, therefore is the proposed strategy to design a network which is based on already existing networks, *pre-trained* networks. Using the existing networks improves the design process and resources regarding different applications where the pre-trained networks already have been applied is available. Since the objective of the choice of using networks to classify objects in images, in this case, a liferaft, it is valid to pick a network specially designed for image classification. The network used in this thesis is the image classification network called *VGG-16* [23]. The *VGG-16* network scored second in the classification tracks, in the Large Scale Visual Recognition Challenge 2014 (LSVRC2014) [23, 24]. The *VGG-16* is a CNN specially designed for image classification, since the LSVRC2014 is a competition with a 1 000-class dataset, the *VGG-16* has a classifier output of 1 000. The dataset used in the competition does not include a liferaft class, hence the classifier part of the CNN needs to be redesigned. The standard *VGG-16* is procured from the *Keras* application library [25] and the last two layers of the *VGG-16* are then removed and replaced with layers that suits the thesis application. The architecture of the *VGG-16* is described in figure 12 and the input image has the dimension $224 \times 244 \times 3$ where the pixel intensities are scaled between 0 and 1. Described earlier in the theory section 3.3 each layer consists of numerous feature maps, pooling and shared weights and

biases. The last two sections of the *VGG-16* hold the classifier layers and since the standard network does not support the objective of classifying liferafts, these sections are removed and replaced.

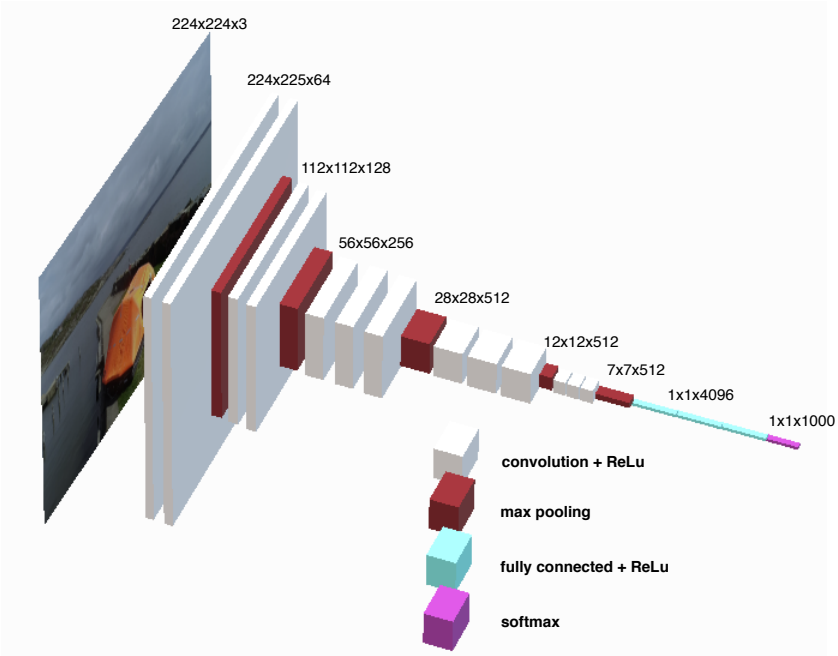

Figure 12: Visualization of the VGG-16 network.

The *VGG-16* holds 138 357 544 tunable parameters called *hyperparameters*, which essentially are weights and biases.

4.2.1 Modified pre-trained network VGG-16

In order for the VGG-16 network to work for the application in this thesis, some changes were made and figure 13 shows these changes. Here the dimensions of the last two layers are changed to $1 \times 1 \times 512$ and $1 \times 1 \times 3$ respectively, meaning the modified *VGG-16* predictions are three classes, one boat class, one liferaft class and one water class. The choice of having three classes is purely based on intuition and it could very well be proved that only two classes are needed, one liferaft class and the other one containing objects which are not liferafts. These objects could be all kinds of various things, but for simplicity, sea vessels are chosen since the intended system is to operate over open water, where, for instance, the possibility of cars present is unlikely. The choice of having a water class is to try to distinguish the boat class and liferaft class from one another, for instance, certain feature maps may obtain specific color features which affect the class prediction. By introducing the water class, the network hopefully can distinguish images where there exists no object but water, this feature may help to distinguish some sea rescue

vessels from liferafts. Since sea rescue vessels may have similar color features as the liferafts. Also, the water class exists for extra decision abilities, for example combining the presences of water together with the presences of the liferaft, i.e. only execute flight path if the drone is over water and have found a liferaft.

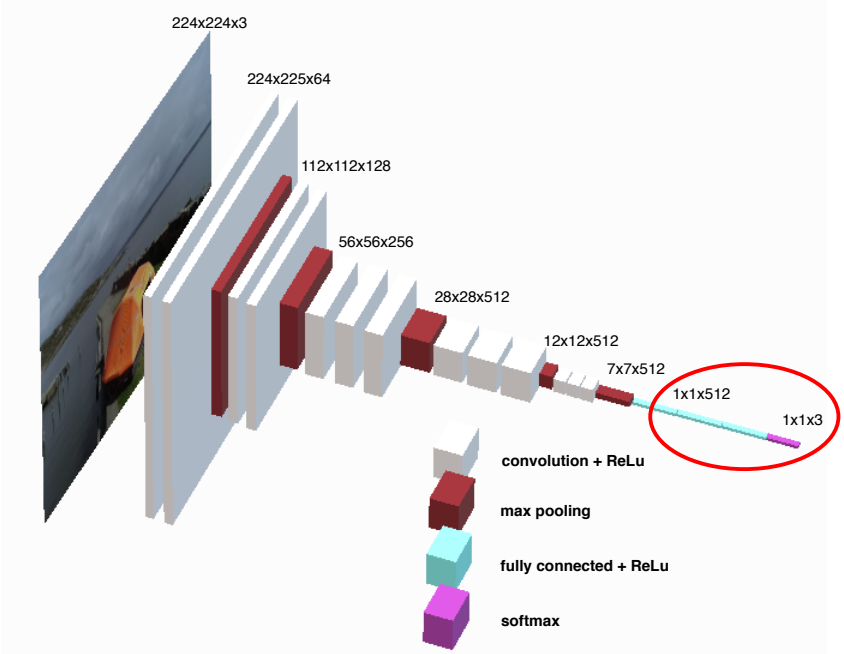

Figure 13: Visualization of the modified VGG-16 network with changes encircled by the red circle.

The number of tunable parameters in the re-designed *VGG-16* is 136 359 747 which is a considerable amount of tunable parameters. However, since the *VGG-16* is a pre-trained network, there exists no need of re-training the whole network, thus the *VGG-16* is already optimized for image recognition. The only necessary trainable part is the re-designed classifier layers and therefore is all layers except the re-designed classifier layers set to *trainable false*. By only training the classifier layers, the amount of tunable parameters decreases dramatically to 2 099 203 parameters and also decreases the training elapse time.

4.2.2 Classifier design

The added classifier layers consist of two added layers, the first added layer is constructed with *Dense* from the Keras core layer library [25], the *Dense* function connects the previous layer with the first added one. The dimension of this layer was set to $1 \times 1 \times 512$ and the activation function chosen was *(ReLu)*, short for rectified linear unit. To decrease over-fitting, *Dropout* is added to the layer. Dropout consists of randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent over-fitting [25]. The last layer added is the actual

classifier, which is connected via *dense* to the previous layer. The dimensions have to match the input classes, which is three, hence $1 \times 1 \times 3$. The selected activation function for the last classifier layer is *softmax*. The decisions of selecting the activation functions and the dimensions of the layers are based on a blog post from Keras [26]. The blog post explains how to build a powerful image classifier with little data, hence describes a valid approach for the training and design of the CNN used in this thesis. The blog post also takes advantage of using a pre-trained network, actually, the blog post uses the pre-trained network *VGG-16* and therefore is the blog post the main inspiration for using *VGG-16* in this thesis.

4.2.3 Optimizer

The Keras documentation provides insight of the specific optimizers that could be used to train NN:s, there exists quite a lot of them but how to chose specific optimizers for different tasks is not immensely clear. The choice of optimizer was determined by the blog post which uses an optimizer called *RMSprop* and this specific optimizer is recommended to only tune the learning rate. Since the tuning of CNN:s can be immensely hard due to the vast amount of tuning parameters, decreasing the amount of optimizer tuning parameters makes the tuning much easier. The chosen optimizer could very well be a bad choice, it could simply be possible to use another optimizer instead to increase the accuracy of the network. Anyhow, to make the tuning process simpler the *RMSprop* optimizer is chosen due to the simplicity of how to vary the tuning parameters.

4.2.4 Data set for training the CNN

The complete data set includes training data, validation data, and test data. The CNN model is trained on the training data and at the end of each training epoch, the optimized weights and biases are tested against the validation data set. After all training epochs are finished the optimized model is tested with the test data set to conclude if the model indeed performs as expected on the training and validation data set. Each data set includes three classes, one boat, one liferaft and one water class. The boat class is obtained by downloading some data sets of different types of boats via *ImageNet* [27]. ImageNet is a website which has a vast collection of data set used for training image recognition NN:s. Unfortunately, ImageNet did not provide any liferaft data sets so the choice fell to collect images of the provided liferaft, the Viking 25 person liferaft. By using the DJI Phantom 4 PRO to record flight videos of the liferaft on land standing on a blue tarpaulin, the data set was augmented via blue-screen in a film editor program to make the life raft appear to float in water by masking out the liferaft and adding water as a background layer. The suggested augmentation of the data could possibly decrease the performance of the image classification due to similarities in the data sets.

4.3 Liferaft prediction

For the drone to find the liferaft, the CNN is used to make a prediction if a raft is present in the image or not. This is used during the search face of the mission to ensure that the drone

does not fly to the wrong object. The decision algorithm will state that it found the liferaft in the image if the CNN prediction is higher than 70% and then move on to center the raft. The prediction is also performed while the drone is flying towards the object to avoid that the drone starts to follow something which is not the liferaft. Once the distance between the drone and the raft is smaller than five meters, most of the image will be filled with the raft and no prediction is needed and color search is only used to keep the raft centered in the image.

4.4 Search for Color in image

Since the liferaft has a very specific color, high visibility orange, it is possible to use this when centering the liferaft in the image. It has also been decided that only one liferaft will be present during the complete mission and it is very rare that objects in the surrounding have a similar color. This means that, even if it is not the most versatile approach, it will work in this case. When the prediction algorithm is certain enough that it is a liferaft in the image, it is possible to isolate the rafts color and make a black and white image with the liferaft as the white part and the rest as black. This is done with the Python library *OpenCv* [28] and the derived algorithm search for combinations of colors which the liferaft consists of, mainly red and orange, and highlights the found color. The algorithm then calculates the average distance in x- and y-direction relative to the center of the image. This way it is possible to course adjust and pitch the camera angle of the drone relative to the detected object. An illustration of the color search algorithm is displayed in figure 14.

Figure 14: Representation of the color search algorithm, the upper image illustrates the distance calculation. The green circle represents the center of the detected object, the blue circle illustrates the center of the image. In the top left corner of the upper image, the distance of the object relative to the image center is display, 35 *pixels in x-direction and* -6.0 *pixels in the y-direction. The lower image is the grey-scale image with the red/orange areas highlighted.*

4.5 Distance calculations towards the liferaft

To get the distance from the drone to the liferaft, simple trigonometry is used. By centering the liferaft in the image from the drone, you can use the altitude of the drone and the angle of the gimbal pitch expressed as α in figure 15, the distance is then calculated as

$$dist = \frac{altitude}{tan(\alpha)} \quad . \tag{14}$$

However, by looking at figure 15, it can be seen that by using equation 14, the distance will not be to the front of the liferaft but rather a bit behind, depending on the angle α. To solve this, an approximate height of the liferaft must be taken into account and equation 14 can be updated as follows

$$dist = \frac{altitude - raft\_height/2}{tan(\alpha)} \quad . \tag{15}$$

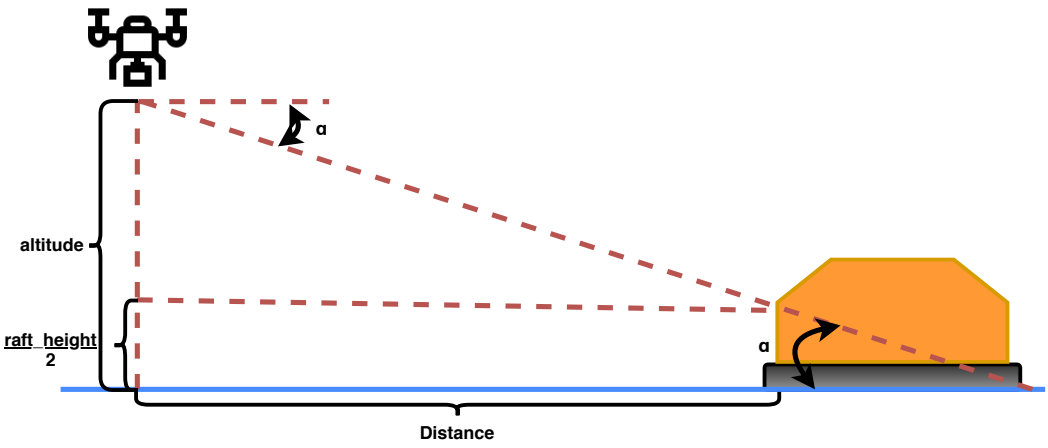

Figure 15: Distance measure to the liferaft.

4.6 Control design

Two different control schemes have been used and evaluated in this thesis. The first one is the position feedback system scheme, where the position of the liferaft is calculated once and then the drone flies the calculated trajectory without any update of the rafts position. The second scheme was camera feedback system, where the rafts position in relation to the drone is continuously calculated using the camera.

4.6.1 Position feedback system

In the position feedback system scheme, the drone centers the raft in the image in order to get a calculated distance to the raft and a bearing. To get a good calculation, the offset to the center of the image has to be very small, hence, an offset of 2 frames in x and y was used. Once the raft is centered, equation 3 will be used to calculate the rafts position and then that position is used to place eight points around the raft, as shown in figure 16, on a radius of 3 meters. These eight points around the raft will be way-points for the drones trajectory.

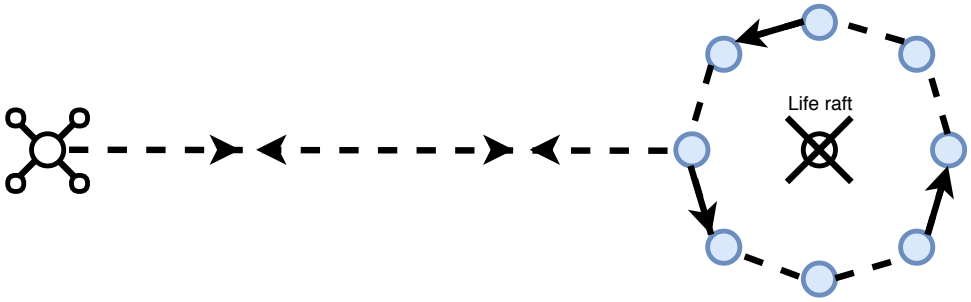

Figure 16: Flight trajectory in position feedback system.

In order to update the control signal for the drone, equation 1 and 2 is used to update the distance and bearing to the next way-point.

4.6.2 Camera feedback system

In the camera feedback system scheme, the drone continuously uses the camera to center the raft in order to get an update of the distance and bearing to the target. Since the update is done continuously, the offset to the center of the image is not as important as in the position feedback system case, hence, an offset of 50 frames is used. When the distance to the raft is 3 meters, the drone starts to circle the raft with a constant roll speed of $1.5m/s$ and keeps the distance of 3 meters to the raft. During the whole circle, the drone will keep the raft close to the center of the image in order to calculate the distance. Once the drone's bearing is the same as when it started to circle the raft, the drone flies back to its home position and land. Figure 17 shows a sketch of the flight path.

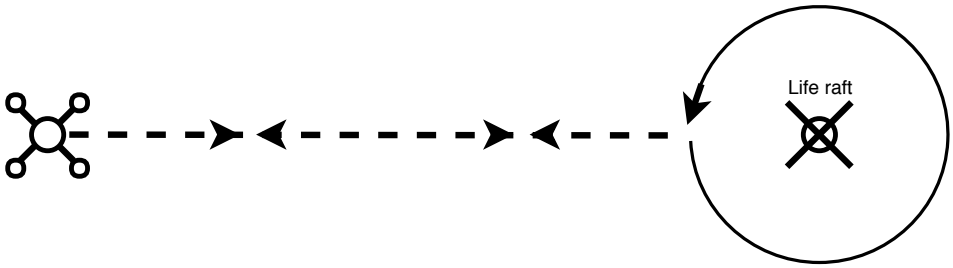

Figure 17: Flight trajectory in camera feedback system.

5 Results

This section presents the tests conducted to investigate the performance of the different parts of the application and the obtained results. First tests are to investigate if the camera feedback system or position feedback system scheme had the best performance and should be used for the complete mission. Then the GPS accuracy was investigated to see how much that could be trusted. After this, different tuned CNN was tested to see which one got the most reliable prediction of the liferaft. Finally, the complete mission was tested.

5.1 Position feedback system performance

In order to check the performance of the position feedback system system, two test scenarios were conducted. One where the object was placed at a distance of 19 meters and the other where the object was placed at 35 meters, both in the same compass direction from the start position. The object is a person wearing an orange high visibility vest and the surrounding is clear from any other orange or red objects in order to only use the color search. To ensure that the tests can be done repeatably, the target is placed on a market position and that position's latitude and longitude value is recorded by placing the drone at that position. GPS readings from the drone is noted and marked with a star in figure 18 - 21. The drone's start position is also market to ensure that the drone always starts from the same position with only a few centimeters in deviation.

The test is conducted in a way where the drone is supposed to rise to a height of 8 meters, center the target in the image, calculate the distance to the target and then use that distance together with the yaw angle to calculate at what latitude and longitude point the target is positioned at with equation 3. When the rafts position is calculated, the drone will fly to the calculated point and then back to its home position. In figure 18 a single run of the 19 meters test is shown and in figure 19 a single run of the 35 meters test is shown.

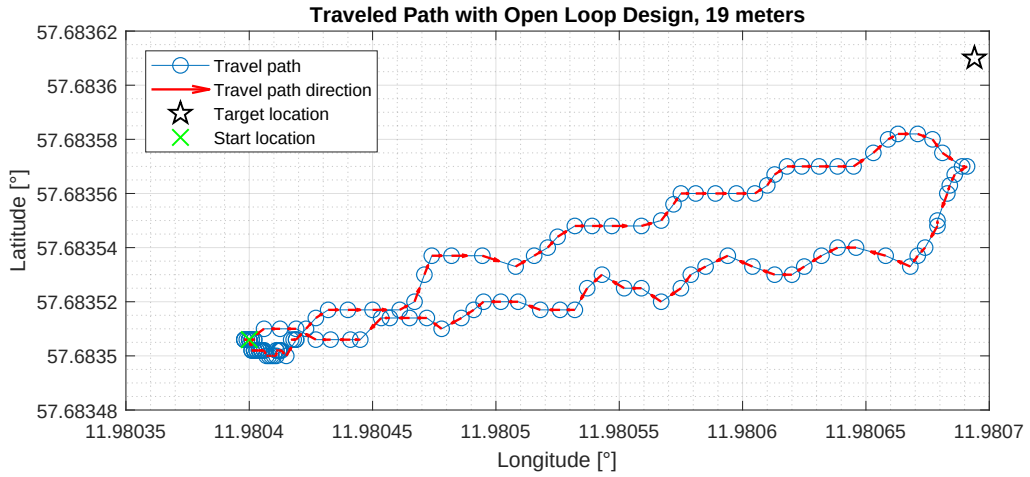

Figure 18: Traveled path with the position feedback system design on 19 meters, where start point, target point and travel direction is marked.

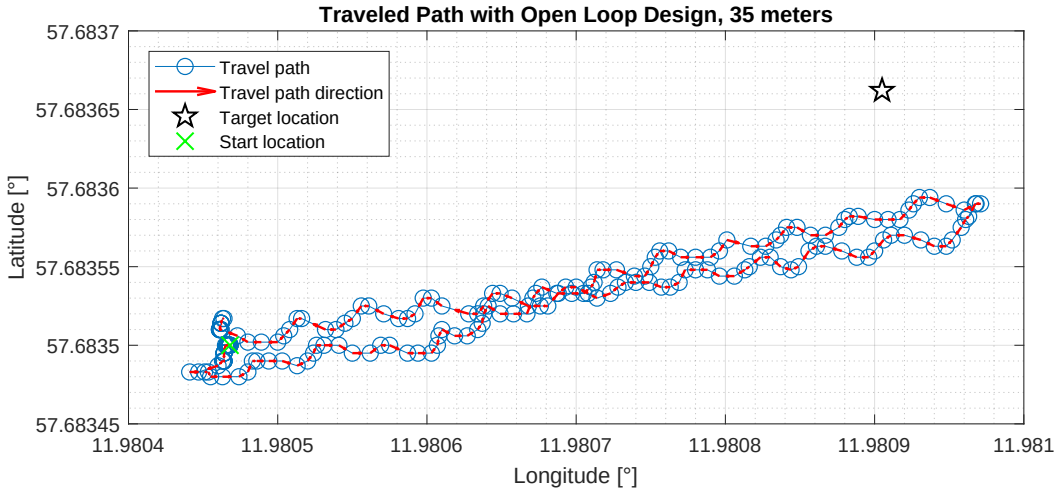

Figure 19: Traveled path with the position feedback system design on 35 meters, where start point, target point and travel direction is marked.

In order to test the repeatability of the system, five runs were done on each distance and the result is shown in figure 20 for the 19 meters test and figure 21 for the 35 meters test.

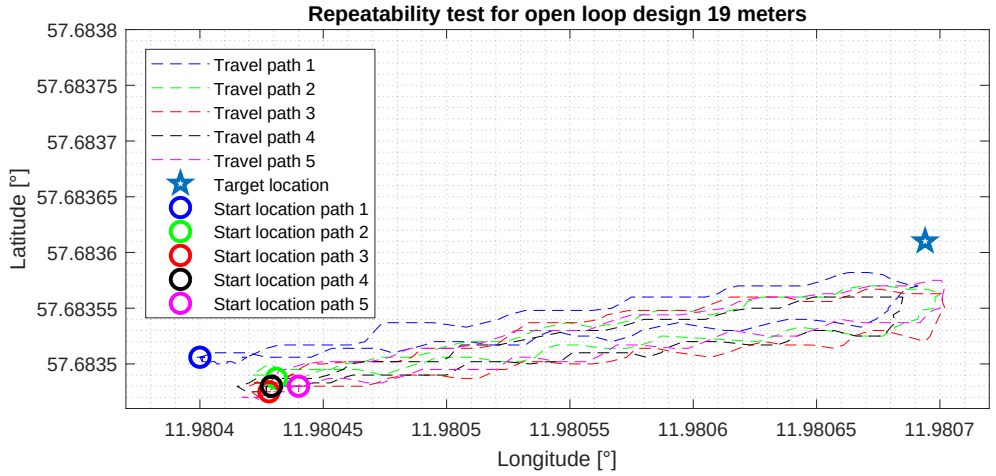

Figure 20: Repeatability test with the position feedback system design on 19 meters, where each color represents a different test flight and start and target point are marked.

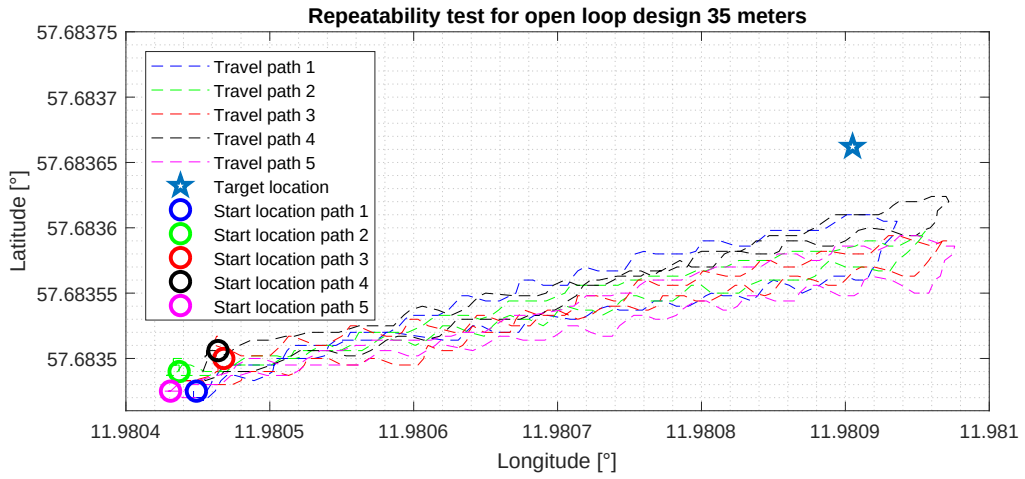

Figure 21: Repeatability test with the position feedback system design on 35 meters, where each color represents a different test flight and start and target point are marked.

As noted in the figures presented above, the drone is not calculating the position of the object correctly in any of the five tests in the two different scenarios but notably is that the calculated value is similar each time. This could either be because of the inaccuracy of the drone's GPS or that the accuracy of equation 3 is not good enough. Even though the target position is calculated using the drone's current position and is subject to GPS inaccuracy, the offset between the objects physical position and the position the drone flew to was too large. The fact that it also was not able to correct its trajectory if the object moved was not positive either. These two shortcomings could not be acceptable and therefore is the position feedback system scheme not used in the complete Mission.

5.2 Camera feedback system performance

The performance of the camera feedback system was tested in the same way as in the position feedback system case. However, since the drone's camera is used to continuously calculate the distance to the raft, it is supposed to stop at a distance of 1.5 meters from the object and then fly back to the home position. In figure 22 and 23 a single flight is shown for the 19 and 35 meters test.

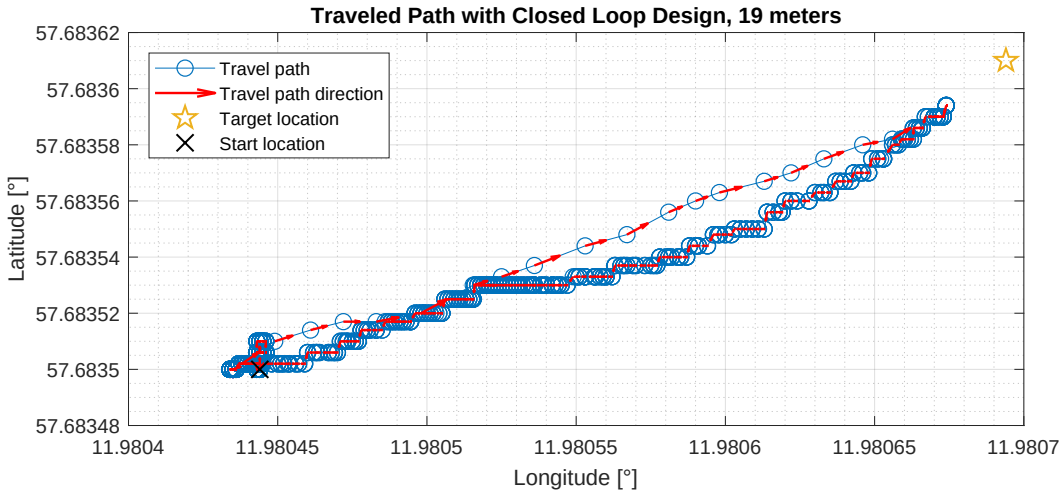

Figure 22: Traveled path with the camera feedback system design on 19 meters, where start point, target point and travel direction is marked.

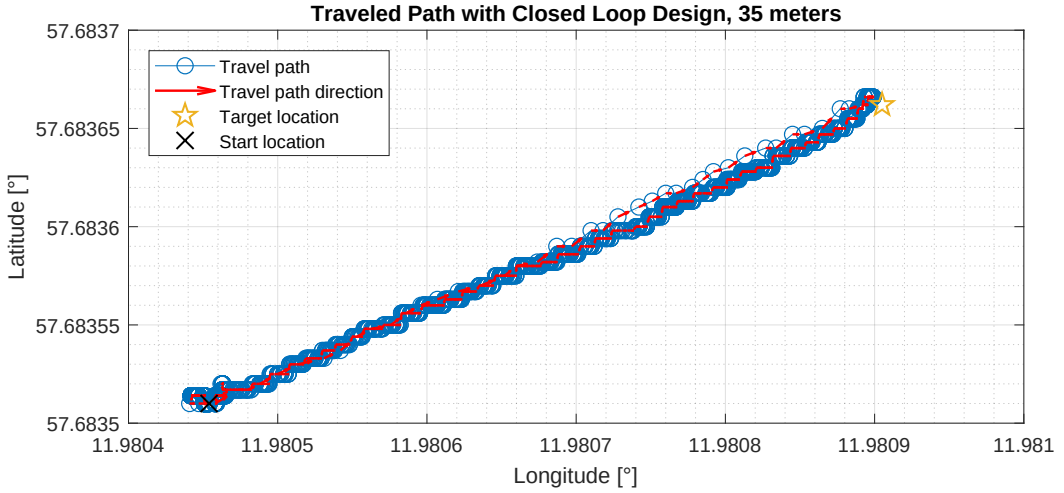

Figure 23: Traveled path with the camera feedback system design on 35 meters, where start point, target point and travel direction is marked.

A repeatability test was done for the camera feedback system scheme as well with five different flights for each scenario. The results from the 19 meters test are shown in figure 24 and for the 35 meters in figure 25.

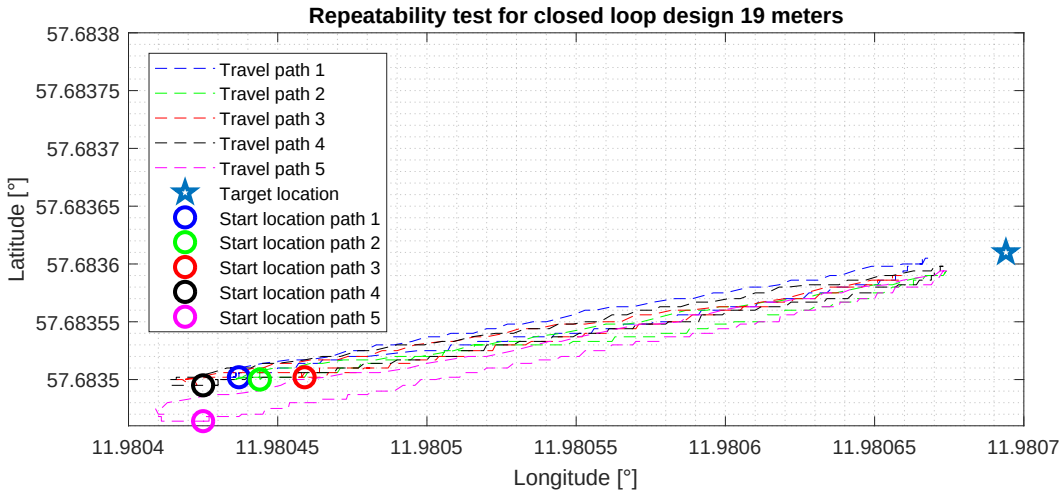

Figure 24: Repeatability test with the camera feedback system design on 19 meters, where each color represents a different test flight and start and target point is marked.

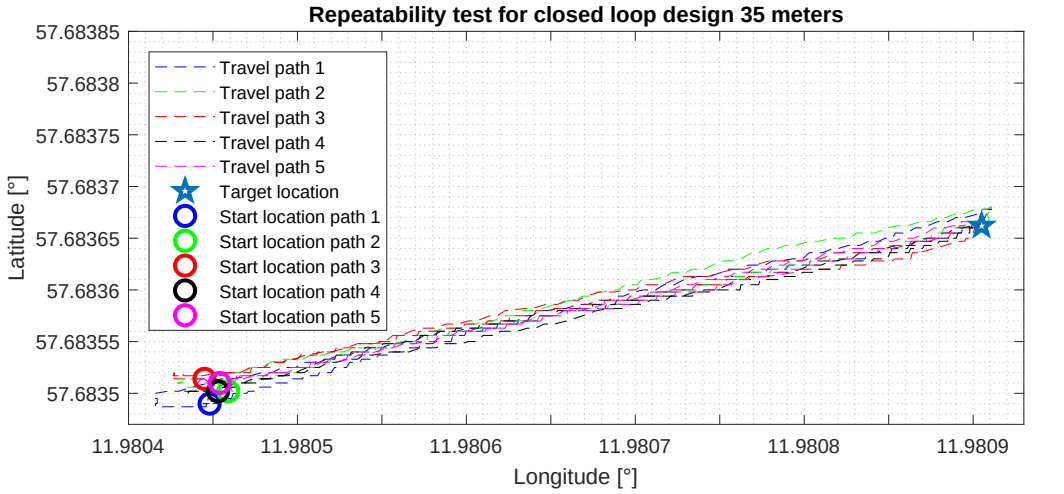

Figure 25: Repeatability test with the camera feedback system design on 35 meters, where each color represents a different test flight and start and target point is marked.

Studying the figures presented above, it is clear that the performance of the camera feedback system scheme is very satisfying. It flies very straight towards the object since it can update the position of the object continuously and it stops and turns around at about the same distance to

the object each time. The reason for that the drone seems to stop directly above the object on the 35 meters test is mainly due to the scaling of the plot and that the recorded position could be slightly off due to GPS inaccuracy. Visual observations during the two test scenarios showed very similar behavior and that the drone turned around close to the same distance to the object.

Tests were also conducted where the object was moving to investigate how good the camera feedback system scheme was to follow the object. There were no difficulties to follow the object as long as the objective was at a greater distance than 5 meters. If the drone was closer than this, it was quite easy for the object to end up outside of the image unless the movement was very slow. This will, however, not be a problem since the liferaft will not move that quickly in the complete mission.

With the results from the camera feedback system tests, it is without a doubt this scheme who has the best performance compared between the closed- and position feedback system. It is also able to follow the object if it moves when the drone flies towards or around it. Therefore is the camera feedback system scheme used during the complete mission.

5.3 GPS accuracy

To see how good the GPS accuracy of the drone was, a test was conducted. This would show how much trust could be put on the GPS and if it could explain the result from the position feedback system scheme. The test was conducted by keeping the drone at the same place on the ground with the motors turned off and then the GPS states were recorded for 300 seconds.

As can be seen in figure 26 - 28, the GPS states varies quite a lot over time with the biggest deviation being approximately 2.7 meters, marked with a red cross in figure 28. This indicates that using the GPS for position calculations in the position feedback system scheme will most likely give an inaccurate value since it is not certain that the drone's position is correct. It also shows that feedback from the camera is crucial for precise flight and for landing at the correct location.

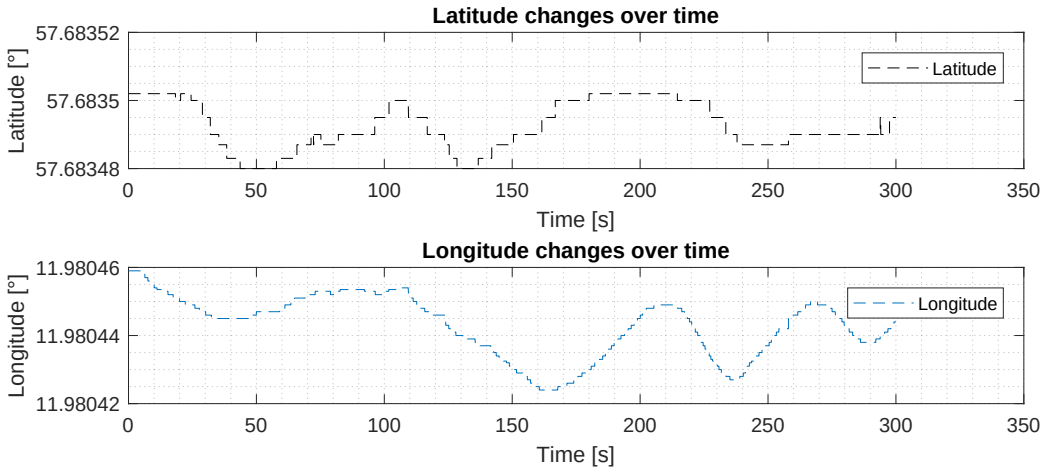

Figure 26: Plot of latitude (top plot) and longitude (bottom plot) change over time during GPS accuracy test.

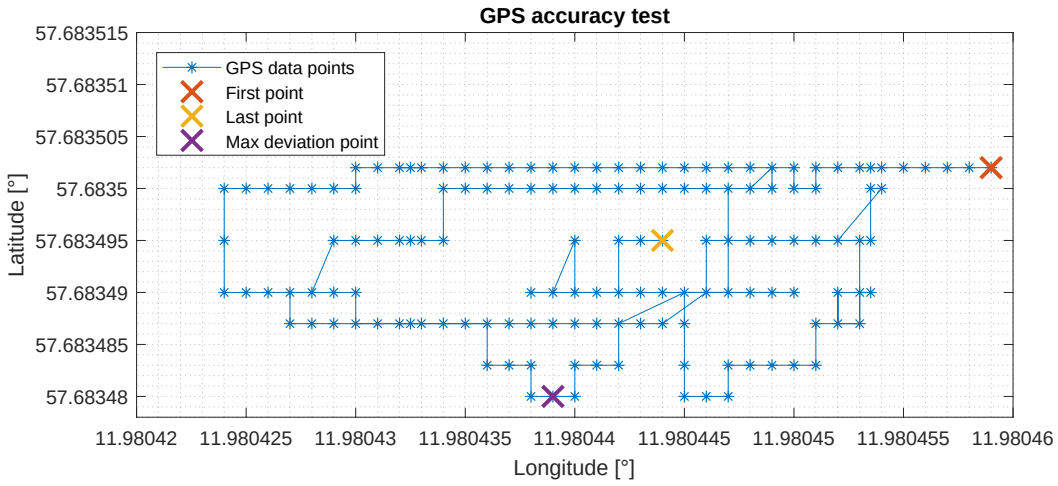

Figure 27: Plot of latitude and longitude change with respect to each other during GPS accuracy test with the first, last and max deviation value marked.

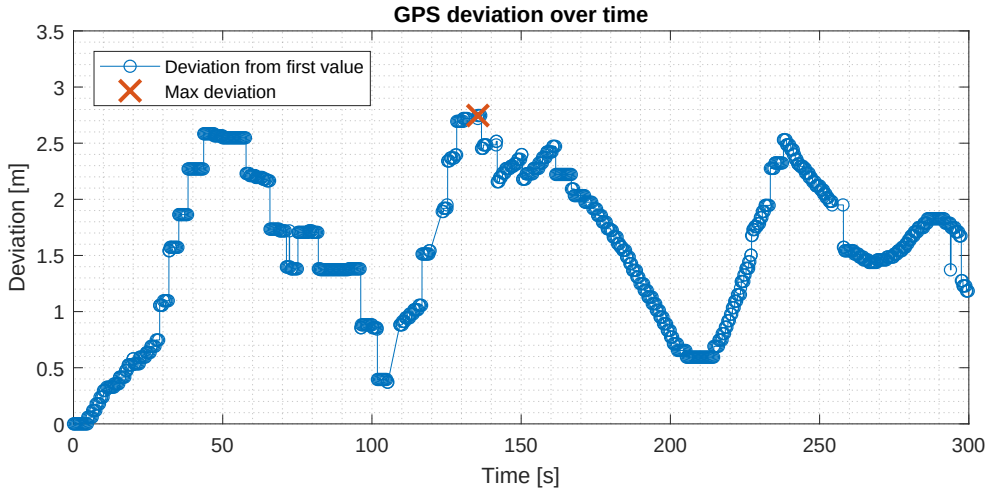

Figure 28: Deviation from the first value during GPS accuracy test with max deviation from the first value marked.

5.4 CNN tuning

In order to get some bearings for the CNN tuning, the optimizer *RMSprop*'s tuning parameters are unchanged, meaning the dropout rate is set to 0.5 and the dimensions of the classifier layers are unchanged from the proposed design in section 4.2.2. The overall training process is represented in figure 29

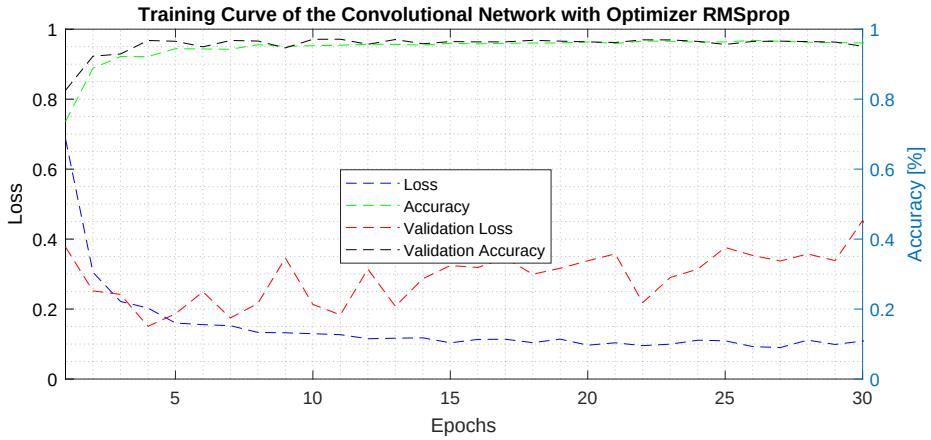

Figure 29: Representation of the initial trained model, the green curve represents the training accuracy, the blue curve representing the training loss, the red curve illustrating the validation loss, and the black curve illustrating the validation accuracy.

Analyzing the training performance of the initial model, it becomes quite clear that the validation loss is off, the goal is to achieve similar curves for the training accuracy/loss and validation accuracy/loss according to [29]. Since the accuracy curves seem to converge there is little overfitting but the learning rate is set to high, this can be illustrated by analyzing the validation loss curve, the curve should smoothly decrease until converging, which it does not. A new attempt to improve the validation loss by decreasing the learning rate to 0.0001 from the initial learning rate 0.001 and additional images taken from the test site was added to the data set. The training performance of the made changes is illustrated in figure 30.

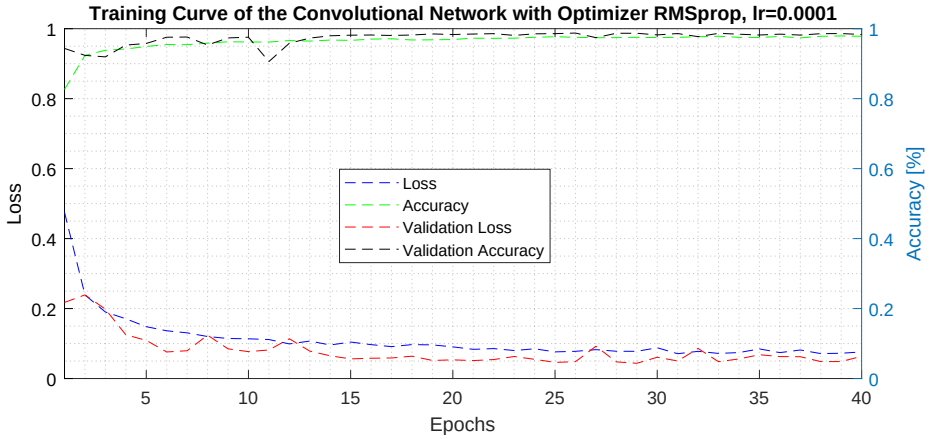

Figure 30: Representation of trained model with changed learning rate $lr = 0.0001$, the green curve represents the training accuracy, the blue curve representing the training loss, the red curve illustrating the validation loss, and the black curve illustrating the validation accuracy.

With decreased learning rate, the validation loss converges which indicates better performance. By comparison with the previous learning rate setting, the decreased learning rate achieved lower validation loss, which is preferable.

5.5 Prediction performance

The performance of the CNN was hard to evaluate since no tests were conducted with the liferaft in the water. However, the results shown in figure 31 states that the trained CNN is able to predict the presence of the liferaft rather well if a lot of water is present around the liferaft. This is expected since the data set, which the CNN is trained with, contains images of liferafts with almost only water around it.

Figure 31: Image series of three consecutive liferaft images displaying the liferaft prediction result in each image, 6.77% in the top image, 41.99% in the middle image, and 89.69% in the bottom image.

After changing the training parameters and adding images of the liferaft at the test site, new prediction tests were conducted. The result is shown in figure 32. Whit these changes made, the CNN does not require as much water to predict that a liferaft is present in the image. The result, however, was not that much improved and since the end goal of this application is to use the system out at sea, the updated data set is not really improving the system.

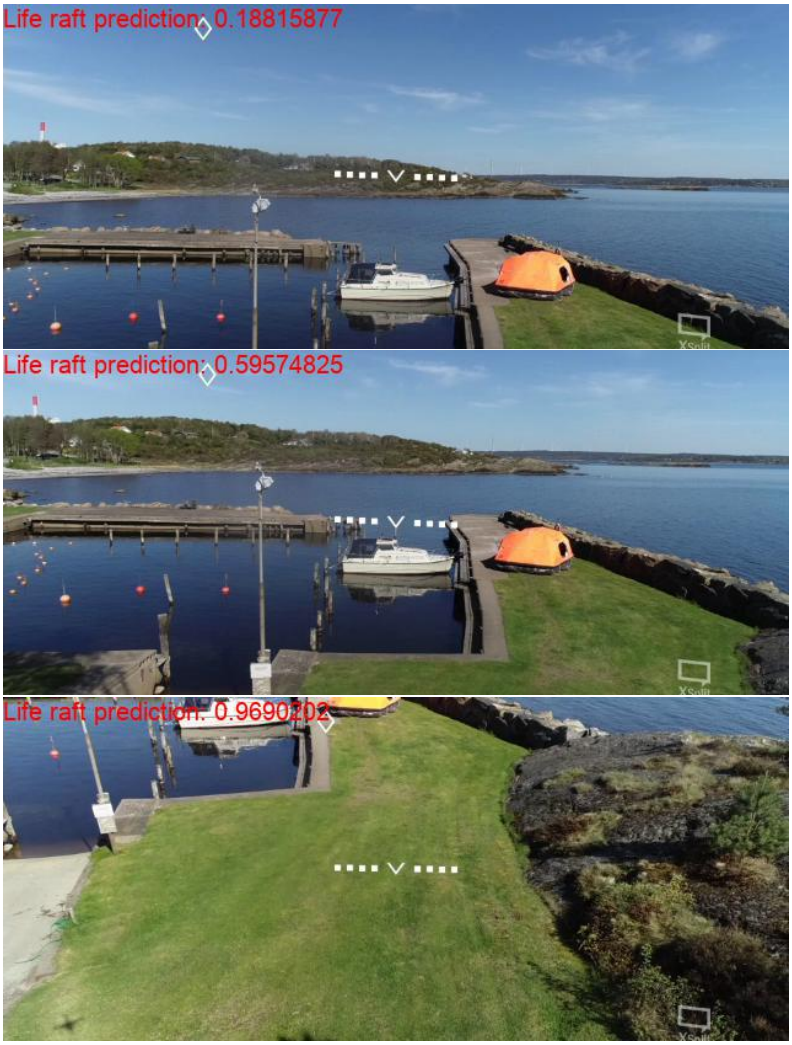

Figure 32: Image series of three consecutive liferaft images displaying the liferaft prediction result in each image with changed training parameters and added images, 18.81% in the top image, 59.57% in the middle image, and 96.90% in the bottom image.

5.6 Complete Mission performance

To test the complete mission performance, a liferaft was inflated and placed on the ground very close to the water, see figure 32. The drone's takeoff spot was not marked and the start position could vary between the different tests and it was facing in a slightly different direction each time. The distance between the drone's start position and the liferaft was approximately 16 meters and no further distances were tried due to limited space on the test site. A good test site was hard to find since the location had to be close to the water, have access to electricity for inflation of the raft and an open area to fly within was required.

The obtained test result for the repeatability test of the complete mission is presented in figure 33 and it can be seen that a very similar behavior is achieved in all five cases. This indicates that the performance of the complete mission is satisfying and the objective to make the drone autonomous and able to find a liferaft has been achieved. More tests on different distances are, however, needed to conclude if the performance of the liferaft prediction is fully satisfying.

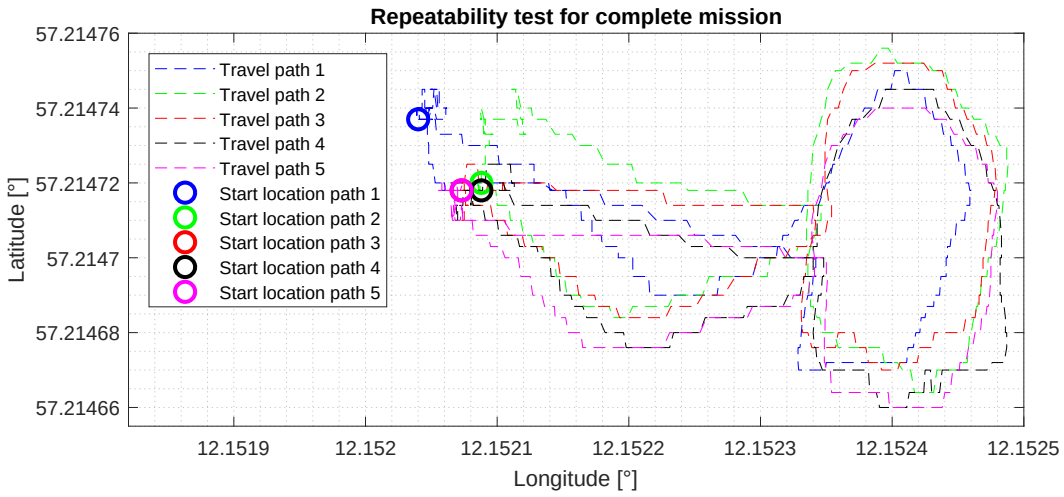

Figure 33: Repeatability test with the camera feedback system design on the complete mission, where each color represents a different test flight and each start point is marked.

6 Discussion

As mention before, the advantages of the camera feedback system system versus the position feedback system system is obvious, the camera feedback system uses the camera and the color search algorithm which enables course correction and continuous update of the distance measurements. These features make the camera feedback system system more robust, precise, and adaptive to changes, like if the liferaft starts to drift due to wind or waves. Since the position feedback system system relies completely on GPS data feedback while traveling towards the estimated liferaft location, not updating the estimated liferaft location will most likely always result in location estimation errors due to the inaccuracy of the GPS system. Considering the limitations of the position feedback system system and evaluating performance versus the camera feedback system system, it becomes clear that the camera feedback system system is preferable.

Concerning the prediction and color search algorithm, there exist some considerations that need to be mentioned. The CNN is designed to output predictions on three different classes, boats, liferafts, and water. Since one objective of this thesis is to localize and find a liferaft in an image, the CNN is specifically designed for this task, hence in-versatile. The combination of combining the prediction together with the color search algorithm provides a rough estimate of the liferaft location but problems occur if there exist numerous red/orange objects except for the liferaft in the image which the predicted is performed on. This will result in the course and distance corrections being off and the flight path towards the raft will not be straight which could result in the drone missing the life raft. The color search algorithm infuses difficulties when detecting other objects rather than the liferaft itself and in the future work section 8, actions towards a more versatile approach to solving this problem is presented.

Using the camera for distance measurements works good for long estimated distances but will not be accurate on small distances. Consideration of the height of the raft is also needed to get a good distance measure which requires knowledge of the raft height in advance. For the tests done in this thesis, the distance measurements were accurate enough to show the functionality of the system. However, if a more accurate flight path is needed in order to ensure that the line gets connected properly, a more precise way to measure the distance between the drone and raft would be needed.

Since the circumstances of the rescue procedure consider liferafts floating at sea, the data set were constructed with images of the specific circumstances but the testing of the proposed system design is mainly done on land close to water. The CNN is predicting on images with altered environmental circumstances which result in the trained CNN having difficulties detecting the liferaft during the tests. Due to the nature of the test scenarios, the validation of the prediction accuracy is cumbersome, some predictions are accurate and others are inaccurate. However, the predictions are accurate if there exists a lot of water in the background of the images which is one drawback of using the augmented data set where most of the collected data set contains images with a liferaft surrounded by water.

7 Conclusions

The objective to lay the foundation for using a DJI Phantom 4 PRO as an autonomous drone and to find a life raft using a deep learning algorithm has been achieved. Without inputs from the pilot, the drone can take off from the ground, search for a life raft, fly to and circle the raft once the prediction is high enough, and then fly back to its starting position and land. The CNN is able to produce accurate liferaft-predictions with reasonable environmental circumstances to the augmented data set, which is expected since the tests were conducted with the liferaft placed on land close to the sea. However, the proposed system design is fairly simple in detail, considering the search for color algorithm, distance approximations, PD regulators, and communication interface, but the overall developed system performs well in simplified scenarios. Simplified scenarios consist of low wind speeds, good visual sight, and the approximated distance to the liferaft is below 50 meters, hence, even due to the simplicity of the system it enables an autonomous option for controlling the DJI Phantom 4 PRO. The proposed system is designed for search and rescue missions but could easily be modified to suit other applications and the developed communication interface lay a foundation of external control by bypassing the drone's RC.

8 Future work

To get this application closer to the desired end goal to attach a line to the liferaft using the drone, more tasks need to be solved. No investigation of how much load the drone can carry has been done. This is needed in order to decide what type of connection device will be used to connect the line to the life raft or if the Phantom 4 PRO is a suitable drone for the mission. How the line will affect the drone during flight depending on how much line is extracted and at what speed the drone is flying must also be investigated.

The limitations of the color search must also be handled to achieve a more versatile solution. One solution would be to use a Fast RCNN[30] where the predicted object gets boxed and then the box can be tracked and centered. This would improve the overall performance of the system and ensure that the drone flies to the correct object.

If the tasks done on the computer could be done on the drone instead would simplify the setup of the system. This could result in a faster system since all calculations are done on the drone's onboard computer and no data has to be sent. This would, however, mean that a different drone than the Phantom 4 PRO would have to be used since the onboard computer is not accessible for us to program. A possible drone to use would be the DJI Matrice 100 [31], which is a developer platform with fully programmable onboard computer and a lot of possibilities to customize which sensors and camera to use.

References

[1] D. Joshi, "Exploring the latest drone technology for commercial, industrial and military drone uses," [accessed June 21, 2018]. [Online]. Available: http://www.businessinsider.com/drone-technology-uses-2017-7?r=US&IR=T&IR=T

[2] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," University of Oxford, Computing Laboratory Oxford, United Kingdom, Tech. Rep., unknown, [accessed June 21, 2018]. [Online]. Available: https://www.cs.ox.ac.uk/files/3198/submission_waharte.pdf

[3] N. S. Foundation, "Small, unmanned aircraft search for survivors in katrina wreckage," [accessed June 21, 2018]. [Online]. Available: https://www.nsf.gov/news/news_summ.jsp?cntn_id=104453

[4] DJI, "Phantom 4 Pro," [accessed June 21, 2018]. [Online]. Available: https://www.dji.com/phantom-4-pro

[5] ——, "DJI Developer Mobile SDK," [accessed June 21, 2018]. [Online]. Available: https://developer.dji.com/mobile-sdk/

[6] Transport Styrelsen, "Drone laws in Sweden," [accessed June 21, 2018]. [Online]. Available: https://www.transportstyrelsen.se/dronare#159819

[7] Razer Inc., "Razer Ripsaw video capture card," [accessed June 21, 2018]. [Online]. Available: https://www.razer.com/gaming-broadcaster/razer-ripsaw

[8] DJI, "DJI HDMI Output Module," [accessed June 21, 2018]. [Online]. Available: https://store.dji.com/product/phantom-3-hdmi-output-module

[9] Viking, life saving equipment, "Viking throw overboard liferaft, type DK+, 10-25 Persons," [accessed June 21, 2018]. [Online]. Available: https://www.viking-life.com/en/throw-overboard/liferafts-/liferafts-/4856-l000d0002-viking-throw-overboard-liferaft-type-dk-10-25-persons-orange-10-persons

[10] DJI, "DJI Developer Mobile App Tutorials," [accessed June 21, 2018]. [Online]. Available: https://developer.dji.com/mobile-sdk/documentation/introduction/index.html

[11] Djacob502, "DJIDemo Ground Station," [accessed June 21, 2018]. [Online]. Available: https://github.com/Djacob502/DjiDemo

[12] Sanduo007, "DJIplus Get GPS signals from drone," [accessed June 21, 2018]. [Online]. Available: https://github.com/Sanduo007/DJIplus

[13] Pisarik, "DJI-phantom-pc-sdk Communication between phone and computer," [accessed June 21, 2018]. [Online]. Available: https://github.com/pisarik/dji-phantom-pc-sdk

[14] DJI, "DJI Simulator Tutorial for Android," [accessed June 21, 2018]. [Online]. Available: https://developer.dji.com/mobile-sdk/documentation/android-tutorials/SimulatorDemo.html

[15] Oracle, "What is a socket?" [accessed June 21, 2018]. [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

[16] D. Witkowska, "Applying artificial neural networks to bank-decision simulations," *International Advances in Economic Research*, vol. 5, no. 3, pp. 350–368, August 1999.

[17] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[18] karpathy@cs.stanford.edu, "Convolutional Neural Networks (CNNs / ConvNets)," [accessed June 21, 2018]. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[19] N. R.Chopde and M. K. Nichat, "Landmark Based Shortest Path Detection by Using A* and Haversine Formula," *International Journal of Innovative Research in Computer and Communication Engineering*, pp. 298–302, 2013.

[20] C. Veness, "Calculate distance, bearing and more between Latitude/Longitude points," [accessed June 21, 2018]. [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html

[21] P. Corke, *Robotic, Vision and Control, The fundamental algorithms in* MATLAB, 1st ed. Springer-Verlag Berlin Heidelberg, 2011.

[22] P. R. C. Panda, *Introduction to PID Controllers - Theory, Tuning and Application to Frontier Areas*. InTech, 2012.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[24] ImageNet, "Results of ILSVRC2014," [accessed June 21, 2018]. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2014/results

[25] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[26] F. Chollet, "Building powerful image classification models using very little data," [access]June 21, 2018. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.htm

[27] 2016 Stanford Vision Lab, Stanford University, Princeton University, "ImageNet," [accessed June 21, 2018]. [Online]. Available: http://image-net.org/index

[28] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[29] karpathy@cs.stanford.edu, "Convolutional Neural Networks for Visual Recognition," [accessed June 21, 2018]. [Online]. Available: http://cs231n.github.io/neural-networks-3/

[30] Fizyr, "Keras RetinaNet," [accessed June 21, 2018]. [Online]. Available: https://github.com/fizyr/keras-retinanet

[31] DJI, "Matrice 100," [access]June 21, 2018. [Online]. Available: https://store.dji.com/product/matrice-100