Autonomous and Accurate Landing of an Unmanned Aerial Vehicle

Louise Tylstedt
Simon Ågren

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis TFRT-6100
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2020

Abstract

During recent years, advancements in mobile technology have enabled the development of quadcopters and unmanned aerial vehicles (UAV) to the point where their size and maneuverability enable indoor flight. In such GPS-denied environments, accurate positioning is a challenge which poses difficulties during flight, navigation and landing. Such positioning systems exist, but are often expensive or rely on environment encroachment. This thesis implements a landing procedure with hardware physically limited to a UAV and its landing platform. This is done by implementing an inner position controller utilizing visual odometry for pose estimation, and an outer sequence controller for discrete event handling. The results show ability to control the UAV from within line of sight to a completed landing. When landing on a flat surface, the results show an average positional error of <12 mm and angular error of <1°. In comparison, when landing on the designed landing platform, the results show less accurate control and a resulting 40 % success rate.

Acknowledgements

First of all we would like to acknowledge our two supervisors, Alexander Isrealsson at Verisure and Anders Robertsson at the Department of Automatic Control. Alexander has been invaluable to the progression of the project as we have received valuable feedback on our weekly meetings and been able to approach him with questions at any time at the Verisure office. Additionally, he, in particular, made sure we became a part of the team at Verisure which we greatly appreciate. Anders has given us input on our progress and we appreciate his commitment and involvement in the project, despite having probably one of the busiest schedules at the Faculty of Engineering.

We would like to thank Sofie Olsson, student at Engineering Mathematics at the Faculty of Engineering at Lund University, who has been working on her master thesis in parallel with us at Verisure. In addition to being great company during coffee breaks, we have been able to discuss difficulties and tackled common obstacles together. We would also like to thank her for helping us with camera calibration using Kalibr.

We would also like to thank the team at Verisure for our time there. Unfortunately, due to the global pandemic of 2020, we were unable to spend a large portion of the spring at the office which we much had looked forward too. In particular, we would like to thank Johannes Book and Noroz Akhlagi for being key persons in making this master thesis project practically and financially possible.

We would like to thank the people at Bitcraze for an excellent development platform and great support. Kristoffer Richardsson, who became our main contact, helped us get started and was there to answer any question and solder abused hardware.

Finally, a big thank you to the group of people who took their time to review this report: Isabelle Larsson, Gustaf von Plomgren, Johan Ågren, Lisa Rygård, and Claudio Mandrioli.

Contents

Acronyms

BOM	Bill of Materials
CF	Crazyflie
FOV	Field of View
FPV	First-Person View
IBVS	Image-Based Visual Servoing
IGE	In Ground Effect
IMU	Inertial Measurement Unit
LOS	Line of Sight
MAV	Micro Aerial Vehicle
MRAC	Model Reference Adaptive Control
OGE	Out of Ground Effect
PBVS	Pose-Based Visual Servoing
PID	Proportional-Integral-Derivative (controller)
RX	Receiver (coil)
ToF	Time of Flight (sensor)
TX	Transceiver (coil)
UAV	Unmanned Aerial Vehicle
VRX	Video Receiver
VO	Visual Odometry
VS	Visual Servoing, vision-based robot control
VTX	Video Transmitter

1

Introduction

1.1 Problem Statement

In the home alarm business, installation, maintenance, and staff are among the major costs. Since such companies promise to respond to every alarm, equal measures are applied to both true alarms (true positives) and false alarms (false positive). To minimize response costs of false alarms a possibility would be to install cameras at all angles in every room of the house. However, this approach is problematic because it entails higher system complexity with a higher level of home environment encroachment by cameras, higher installation and maintenance costs, and a decreased sense of privacy by the home owner.

Technological advancement in the mobile and smartphone industries during recent years has enabled the development of two relatively new markets: smart homes, and smaller hobbyist commercial unmanned aerial vehicles (UAVs, often referred to as quadcopters or drones). Smaller and more powerful processors and batteries lie at the heart of the evolution of these products. Many markets benefit from this development and entirely new ones have emerged. Among them are the agriculture, toy and photo-/videography markets, drone racing, and drone delivery markets.

An alternative to the installation of cameras in every room is to have a mobile camera. An autonomously controlled mobile device could in case of an alarm power on, reach the location where the alarm triggered and send a real-time video stream to an alarm provider's service center. Taking advantage of the recent technological development within the field, the camera could be mounted on a UAV that is small enough to enable effective navigation indoors. Such a platform could require minimal installation and home environment encroachment.

The conceptual solution of a mobile, UAV-mounted camera poses a list of challenges. A trade-off between flight time, and camera weight and power consumption is introduced. Another problem is precise and accurate landing on a base station as required for charging and storage during inactivity. In addition to stable flight control, landing at a stationary platform requires real-time information regarding the relative position of UAV and platform. Positioning systems for indoor use exist but are often expensive, rely on advanced sensors and/or require extensive environment

intervention with multiple anchors around the workspace. Accordingly, the problem faced in this thesis is how to achieve minimal encroachment positioning using low cost, conventional sensors to enable precise and accurate landing.

1.2 Purposes and Goals

The purposes of the project are:

1. To identify the challenges of precise, accurate, and autonomous landing,
2. To propose a functional solution to the problem, and
3. To begin development of a UAV for domestic use.

Originating from named objectives is a list of tasks which have been set up as goals, namely:

- Design a hardware and software setup for positioning of the UAV, physically limited to the UAV and landing platform,
- Design and implement a position controller using input from above mentioned setup,
- Propose an implementation of a landing algorithm for above controller,
- Investigate performance of proposed controller and algorithms,
- Design a landing platform, henceforth called base station, for the UAV containing required hardware for the proposed solution.

There are several reasons why precise and accurate landing is of importance. First, it is desirable to construct a base station with smallest possible (physical) footprint. Second, the feature of automatic charging of the UAV is an important design consideration. In other words, an inaccurate landing could result in a crash or failed charging, requiring human intervention and deeming the UAV non-autonomous.

A phenomenon which occurs during ground proximity flight is the so-called In Ground Effect (IGE) which aerial vehicles such as airplanes, helicopters and quadcopters are subject to. The effect is a change in thrust due to change in aerodynamics when close to solid obstacles [He et al., 2019]. In this work, this effect will be evaluated and included as a design consideration.

1.3 Thesis Outline

In Chapter 2, an insight into the area is provided through a study of previous research.

In Chapter 3, relevant theory is presented. The chain of theory that is required to achieve autonomous landing is built by presenting its links: rigid-body transformation, visual odometry and control, quadcopter mechanics and control, and sequencing control. The ground effect and wireless charging are explored as they are important design considerations. Following this structure, the method and proposed solution is presented in Chapter 4.

In Chapter 5, the experiments to test the proposed solution are described. This is done by evaluating the links individually to an extent as large as possible. The results of the tests are presented in Chapter 6. A discussion of the results is then conducted in Chapter 7, along with possible improvements of the solution. Finally, Chapter 8 concludes the thesis and suggests future work.

1.4 Delimitation

The extent of this thesis has been limited to define a feasible scope completable within the set time frame. The delimitations are:

- The Crazyflie 2.1 by Bitcraze and its surrounding infrastructure will be used as a development platform. It is chosen for being open source, having an active community and is sold ready-to-fly. This allows us to focus on the core problem, rather than building a quadcopter and developing a flight controller.
- The UAV will be equipped with a camera and a video transmitter. The final Verisure product will need a camera to detect false alarms as previously described. Arguably, it is therefore advantageous to use a camera for automatic control in this project and not any other type of sensor.
- The possible effects of a change of platform will not be investigated, only discussed. Similarly, the effects of different hardware models other than the ones chosen will not be investigated. This is due to the fact that the project has a limited budget.
- The autonomous landing developed in this thesis assumes the base station is within line of sight (LOS) of the UAV. Although related, indoor localization and navigation are deemed non-essential for landing. Rather, it is a project in and of itself. Appropriately, this is a topic currently being investigated at the Centre for Mathematical Sciences at Lund University and Verisure by our co-master-student Sofie Olsson.

1.5 Division of Labor

Here, a list of roughly what was done by whom of the two thesis students is provided.

Simon
Report: Introduction, Background (2.1), Theory (3.1-3.3, 3.7), Implementation (4.1, 4.3-4.4), Experiment, Results, Discussion and Conclusion
Implementation: Code (Python) implementation, control & controllers, testing, camera mount design, figures, and graphs.

Louise
Report: Background (2.2-2.4), Theory (3.2, 3.4-3.6, 3.8), Implementation (4.2), Discussion and Conclusion
Implementation: Base station design & testing, and camera mount design

2

Background

In this chapter some of the background to the project is presented. Firstly, some previous research relevant to this study is presented. Further on, the thoughts and arguments that led up to the final concept are described. The final concept presented is carried out throughout this study. Then some possible applications of this solution are discussed. Lastly a short description of the company Bitcraze AB that lay the foundation for the UAV platform used in this project is provided.

2.1 Previous Research

As quadcopters and their development have become more readily available a lot of research is being done on the subject of UAV and quadcopter control. Due to that the area is exciting and challenging, it attracts researchers, enthusiast and students alike. Several papers deal with visual servoing (VS), or visual-based robot control, which is where visual information used in automatic control applications. More specifically, pose-based visual servoing (PBVS) where the complete 3D pose is estimated using a camera.

In [Karlsson, 2019], a method similar to the one presented in this thesis is proposed. A camera is mounted to a micro aerial vehicle (MAV) and computer vision is used to identify an ArUco marker to calculate the quadcopter's relative pose. The quadcopter is able to land on a marker but has insufficient landing accuracy to be used as-is in this project. The method does not consider that the camera and inertial measurement unit (IMU) frames have a relative translation and rotation. Including this would be an evident improvement.

Similarly, [Goeller, 2018] proposes a method for precision landing of a quadcopter with a *theoretical* landing accuracy of 5 cm. A conclusion which is drawn in the paper is that the implemented marker detection has insufficient trustworthiness. It utilizes color masking which is sensitive to changes in light conditions.

In [Acuña and Willert, 2018] a proof of concept which achieves landing us-

ing dynamic fiducial markers[1] is presented. An increased range and accuracy is achieved by varying the type and size of marker. Centimeter precision landing is achieved with the particular quadcopter. A common factor of the above papers is that they do not take the ground effect into consideration. Numerous research is being done to investigate the ground effect, see Section 3.6 for theory on this topic.

In [Del Cont Bernard et al., 2017], the ground effect is investigated from a dynamical approach, as opposed to the conventional, static method of observing fixed rotors and constant set-points. The paper concludes that in addition to static effects, the distance to ground impacts the attitude (pitch and roll) dynamics.

Finally, in [Wei et al., 2019] the proposed control architecture of model reference adaptive control (MRAC) improves the tracking performance of the altitude command when the UAV is in the ground effect region. The controller is based on a linear, height-varying empirical model obtained in the paper. A common factor of the above papers is that they use external positioning systems such as motion capture systems, rendering them reliant on environment intervention beyond a single base station.

2.2 Concept Selection

Before the final concept was chosen various other solutions were discussed and investigated. Below follows motivations for the concept that was continued with and some arguments for discarding other solutions.

The final concept is placing a camera on the drone pointing downwards, using fiducial markers and computer vision (through the OpenCV libraries) for positioning relative to the marker. The camera images are sent though radio communication to an external processing unit for computations and then controller commands are sent back to the drone through another radio frequency.

A base station is also a part of the final concept. The base station is designed to passively improve the landing accuracy, allow for wireless charging and housing for a processing unit. In Figure 2.1 the base station and the Crazyflie 2.1 with a camera and a video transmitter can be seen.

Other possible concepts were also generated before the final concept selection. Among these were placing a camera on the base station and through computer vision identifying the drone and its relative position. This would give more stable pictures since the base station is fixed. However, the marker placed on the drone for computer vision identification would have to be small in order to fit on and be carried by the mini drone. This would make identification of the marker at long distances difficult.

[1] A *fiducial marker* is an object of known dimension placed in the environment to be used as reference for later image analysis. In robotics applications they are often used to estimate the pose of an object relative to a camera.

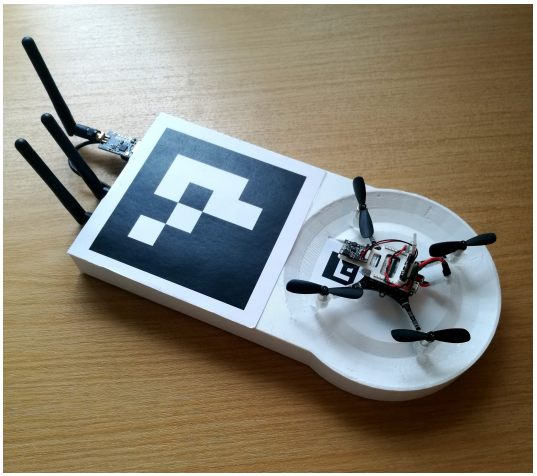


Figure 2.1 The final concept

As concluded in the previously mentioned study, [Acuña and Willert, 2018], fiducial markers are reliable in the terms of precise pose estimation. There are several advantages and among these is the possibility of error correction and adjustments from information loss. The fiducial markers, also referred to as ArUco tags in this report, are explained further in Section 3.2.

There are other markers apart from the ArUco markers that could be used along with computer vision for positioning. As mentioned in Section 2.1, in the study of [Goeller, 2018] markers in a specific color and placed in a specific order were used. However, that solution resulted in not being so precise and sensitive to changes in light conditions. To avoid light condition issues, a solution could be positioning LED:s in a pattern. This way it would be hard to detect angle deviations as the relative size of the LED:s in the image would have to be estimated. Another great argument for using ArUco tags is the simplicity of using the existing library for them with ready-to-use functions. New markers positioned in a new pattern would require new algorithms and would be time consuming.

Other kinds of positioning systems were also discussed such like HTC Vive, motion capture (MOCAP) and Radar. These systems require installation of multiple anchors preferably positioned with distances separating them more than a few decimeter. As the purpose of the project is a simple implementation in a home it is not appropriate with additional installations, and therefore were all of these positioning systems discarded.

The design of the base station was based on the idea of implementing wireless charging. Contactless charging was considered vital as a solution with contacts would require additional force in order to plug in the contact, for example from several motors holding the quadcopter in place as the contact is inserted. An al-

ternative to traditional contacts could be implementing contact pads to charge the battery through, e.g., the legs of the Crazyflie. However, this idea was not taken further as it would require design of the electric circuit along with a lot of testing and furthermore, a higher landing precision compared to a wireless solution. Therefore, a wireless solution seemed to be the most suitable for this project. As described further in Section 3.8, magnetic induction is a wireless charging method well suited for this kind of application and was therefore chosen as the best charging concept.

2.3 Other Possible Applications

As drones are becoming more popular new applications for them are found. For example they can be used in search and rescue operations as they could enter spaces that would be too dangerous for humans to enter. Also deliveries to dangerous or non accessible areas can efficiently be made by these small aircrafts. A more precise landing could be of use for these applications as they commonly use GPS for localization which can deviate up to a few meters or more in accuracy. If the aircrafts are to travel far distances the need for recharging the batteries must also be fulfilled. Therefore, a wireless landing station on the way to the final goal could be crucial. An efficient power transfer also requires a precise landing.

2.4 Bitcraze AB

The quadcopter platform used in this project is developed by Bitcraze AB. Bitcraze AB is a Swedish company founded in 2011 that develops and manufactures a small quadcopter called Crazyflie. To add extra features and functions to the quadcopter they offer different kinds of expansion decks that are easily mounted and integrated to the Crazyflie system. They also develop and maintain an open source software infrastructure with development environments, debuggers and clients.

The initial idea that lay the foundation for the company was to make an electronic board fly. Three embedded engineers from Sweden wanted to make a simple and small flying device that could operate indoors. The result was the first Crazyflie, a small quadcopter with a PCB as the main mechanical frame and with small motors glued on to it. At that time, in 2009, it was the smallest quadcopter in the world [Bitcraze, 2020b].

3

Theory

In this chapter, relevant theory is presented in order to give an understanding of the topics covered further on in this project.

First, the algebra of rigid-body transformation is described to enable transformation between frames of reference, e.g., camera and world frame. Visual odometry (VO) including camera models, camera calibration and pose estimation is then presented to determine the relative transformation between image, camera, and world frame. Visual servo control is then described to provide a foundation for controller implementation using the camera image feed. The basic understanding of quadcopter dynamics and the Crazyflie control structure are later explained. Further, the fundamentals of the ground effect are explained with suggestions on controllers that can handle this effect. In the following section, sequencing control is described. It is the system of using finite-state machine automata for automatic control, applicable to this thesis landing algorithm. Lastly the basic fundamentals of wireless charging in general and inductive charging in particular are described.

3.1 Rigid-Body Transformation

This section describes how points in different reference frames, e.g., camera and world frame, can be related to each other. Any 3D rigid-body transformation can be described as the result of a rotation and translation, described below according to work in [LaValle, 2006], [Kumar, 2018], [Rodrigues, 1840], and [Mason, 2006]. Using the theory of this section, the translational and rotational 6D pose of the UAV may be determined given an arbitrary translation and rotation expressed in different ways.

Rotation

Rotation is the description of the change in orientation of an object. It can easily be seen that a rotation of α in two dimensions can be described as

$$R(\alpha) = \begin{bmatrix} c_\alpha & -s_\alpha \\ s_\alpha & c_\alpha \end{bmatrix}, \tag{3.1}$$

where, $c_\alpha = \cos(\alpha)$ and $s_\alpha = \sin(\alpha)$, a notation which will be used throughout this thesis.

Similarly, a rotation in three dimensions (using a right-handed coordinate system) around an axis, e.g., z, can be seen as a 2D rotation applied to two axes, e.g., x and y, where the coordinate remains constant for the axis in question. These matrices become

$$R_z(\psi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.2}$$

$$R_y(\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix}, \tag{3.3}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}, \tag{3.4}$$

where the notation of roll ϕ, pitch θ, and yaw ψ is borrowed from aviation as rotation around the x, y, and z-axis respectively. Equations (3.2)-(3.4) together form the compete rotation matrix

$$\begin{aligned} R(\phi,\theta,\psi) &= R_z(\psi)R_y(\theta)R_x(\phi) \\ &= \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}. \end{aligned} \tag{3.5}$$

Note that the order of rotation matters. Here, R in Equation (3.5) applies rotation around the x, y, and z axis in that order. The rotation matrix satisfies a few conditions and characteristics:

- Transpose is the inverse, $R^T = R^{-1}$
- Determinant is 1, $\det(R) = 1$
- Rotations preserve cross products, $Ru \times Rv = R(u \times v)$
- Rotation of a skew symmetric matrix $A^\wedge$ ($-A = A^T$) becomes $RA^\wedge R^T = (RA)^\wedge$

Rotations may also be described as occurring around an arbitrary axis, rather than the world frame coordinate system as described above. One way of such representation is through the Rodriguez rotation formula [Rodrigues, 1840] where an arbitrary vector v is rotated about an arbitrary axis n

$$v' = n(n \cdot v) + \sin\theta(n \times v) - \cos\theta n \times (n \times v), \tag{3.6}$$

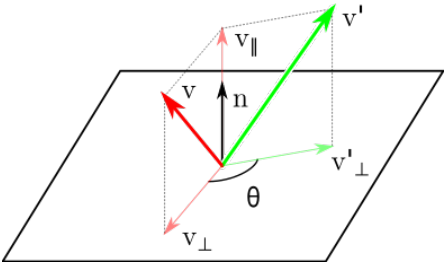


Figure 3.1 Illustration of Rodriguez rotation formula.

where n is a unit vector and θ the angle of rotation perpendicular to this axis in counter clockwise direction. This rotation is illustrated in Figure 3.1 using the Equations (3.7)-(3.9).

$$v = v_\perp + v_\parallel \tag{3.7}$$

$$v_\parallel = n(n \cdot v) = v'_\parallel \tag{3.8}$$

$$v_\perp = v - v_\parallel = v - n(n \cdot v) = -n \times (n \times v) \tag{3.9}$$

where $v_\parallel$ is the vector projection of v on n, $v_\perp$ is the vector rejection of v from n.

The Rodriguez rotation formula can be related to a rotation matrix on the form of Equation (3.5) by defining a matrix N

$$N = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}, \tag{3.10}$$

so that $Nv = n \times v$ and rewriting Equation (3.6) as

$$\begin{aligned} v' &= v + \sin\theta(n \times v) + (1 - \cos\theta)n \times (n \times v) \\ &= v + \sin\theta Nv + (1 - \cos\theta)N^2 v \end{aligned} \quad . \tag{3.11}$$

This leaves the rotation matrix R after factorizing v and expanding:

$$\begin{aligned} R(\theta, N) &= I + \sin\theta N + (1 - \cos\theta)N^2 \\ &= \begin{bmatrix} n_1^2 + (1 - n_1^2)c_\theta & n_1 n_2(1 - c_\theta) - n_3 s_\theta & n_1 n_3(1 - c\theta) + n_2 s_\theta \\ n_1 n_2(1 - c_\theta) + n_3 s_\theta & n_2^2 + (1 - n_2^2)c_\theta & n_2 n_3(1 - c_\theta) - n_1 s_\theta \\ n_1 n_3(1 - c_\theta) - n_2 s_\theta & n_2 n_3(1 - c_\theta) + n_1 s_\theta & n_3^2 + (1 - n_3^2)c_\theta \end{bmatrix} \end{aligned} \tag{3.12}$$

A 3x3 rotation matrix may be used to determine the rotational angles of the pose, i.e., the roll, pitch, and yaw [LaValle, 2006]. Consider an arbitrary rotation matrix R:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{3.13}$$

Setting this matrix equal to Equation (3.5), using the relations $r_{21}/r_{11} = \tan\psi$, $r_{32}/r_{33} = \tan\phi$, $r_{31} = -\sin\theta$, and $\sqrt{r_{32}^2 + r_{33}^2} = \cos\theta$ and solving for roll, pitch and yaw:

$$\phi = \tan^{-1}(r_{32}/r_{33}), \tag{3.14}$$

$$\theta = \tan^{-1}\left(\frac{-r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right), \tag{3.15}$$

$$\psi = \tan^{-1}(r_{21}/r_{11}), \tag{3.16}$$

assuming $r_{11} \neq 0$, $r_{33} \neq 0$.

Mathematically, the quadrants of the angles of the arctan function are not uniquely defined. However, by considering the sign of the numerator and denominator this can be determined. This function is called atan2.

Translation

A translation is the description of the change in position of an object. The change can be described as the addition of a vector $\mathbf{t}$

$$\mathbf{t} = \begin{bmatrix} \Delta x & \Delta y & \Delta z \end{bmatrix}^T. \tag{3.17}$$

By introducing a translation matrix t_v and homogeneous coordinates the position change of a point $\mathbf{p}$ may be described using matrix multiplication as

$$\mathbf{p}' = \mathbf{p} + \mathbf{t} = \begin{bmatrix} p_x + \Delta x \\ p_y + \Delta y \\ p_z + \Delta z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = T\mathbf{p}. \tag{3.18}$$

Naturally, the translation matrix may also be used to represent the translation relating two coordinate systems.

Transformation

Transformation is the combination of an objects translation and orientation and can describe an arbitrary rigid-body displacement.

The rotation and translation matrices of Equations (3.5) and (3.18) may be assembled to form a transformation matrix

$$T = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{t}_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix}. \tag{3.19}$$

3.2 Visual Odometry

Visual odometry (VO) is the process of estimating the pose of a camera utilizing its visual information. The method addresses the relation between a digital camera's inherent 2D pixel coordinate system to the world 3D coordinate system. A camera and an image distortion model are necessary to express how points are projected through the lens and onto the image plane. Camera calibration is the process of estimating the model's parameters. The model and features in an image are then used for pose estimation.

Camera Modeling

Modelling the camera's properties is at the core of VO. To model the first person view (FPV) camera used in this project (see Section 4.2), different types of models have been evaluated. The pinhole camera model [Zhang, 2000] is a simple and very commonly used model. In its simplest form it cannot model lens distortion and is therefore often supplemented with a radial-tangential (often written "rad-tan") distortion model [Ma et al., 2003]. However, the rad-tan model is not capable of modelling extreme distortions which may be present in wide angle cameras such as FPV cameras. Therefore, the fisheye camera models presented in [Scaramuzza et al., 2006] and [Kannala and Brandt, 2006] are also evaluated.

Pinhole Camera Model Denoting a camera 2D point in homogeneous coordinates as $p = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ and, similarly, a world 3D point as $P = \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T$. The relation between them using the pinhole camera model is described as

$$w\mathbf{p} = C\mathbf{P} = \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \underbrace{\begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \mathbf{P}, \tag{3.20}$$

where w is a scaling factor, C the camera matrix consisting of its extrinsic parameters (rotation R and translation t) and its intrinsic parameters K (optical center u_0, v_0, focal length α, β in u, v axes respectively, skewness factor γ between axes). The relation may be simplified as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x X_c / Z_c + u_0 \\ f_y Y_c / Z_c + v_0 \end{bmatrix}, \tag{3.21}$$

if $Z_c = 0$ where $\begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T = \begin{bmatrix} \mathbf{R}|\mathbf{t} \end{bmatrix} \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T$ are camera coordinates.

Radial-Tangential Distortion Model The relation of Equation (3.20) does not model distortions as a result of a camera lens. Especially, a wide angle/field of view (FOV) camera adds considerable radial distortion. This can be described by the relations

$$\begin{aligned} x_{dist} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \\ y_{dist} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned} \tag{3.22}$$

where $r^2 = x^2 + y^2$, k_i is the radial distortion coefficients, x_{dist}, y_{dist} are the distorted pixel locations and x, y are the undistorted pixel locations. Tangential distortion is the result of misalignment of camera sensor and camera lens planes. This can be described by the relations

$$\begin{aligned} x_{dist} &= x + (2p_1 xy + p_2(r^2 + 2x^2)), \\ y_{dist} &= y + (2p_2 xy + p_1(r^2 + 2y^2)), \end{aligned} \tag{3.23}$$

using above definitions and where p_1, p_2 are the tangential distortion coefficients.

Fisheye Distortion Model As previously mentioned, the rad-tan model is unable to model extreme distortions introduced by a fisheye lens. In both [Scaramuzza et al., 2006] and [Kannala and Brandt, 2006], the fisheye distortion model is written as:

$$\theta_d = k_1\theta + k_2\theta^3 + ... + k_5\theta^9 + ... \tag{3.24}$$

where $\theta = \arctan(r)$ is the angle of the incoming ray, and $r = \sqrt{x^2 + y^2}$ is the distance between image point and principal point. In [Bradski, 2000], Equation (3.24) of the 9th power is used and, similar to Equation (3.22), the distorted coordinates become

$$\begin{aligned} x_{dist} &= (\theta_d / r) X_c / Z_c, \\ y_{dist} &= (\theta_d / r) Y_c / Z_c, \end{aligned} \tag{3.25}$$

using previous variable notations.

Regardless of distortion model presented above (rad-tan or fisheye), the undistorted image coordinates may be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x_{dist} + u_0 \\ f_y y_{dist} + v_0 \end{bmatrix}, \tag{3.26}$$

analogous to Equation (3.21). The complete chain of transformation from (to) the display frame O_D to (from) the world frame O_W, i.e., Equations (3.20) to (3.23), is illustrated in Figure 3.2.

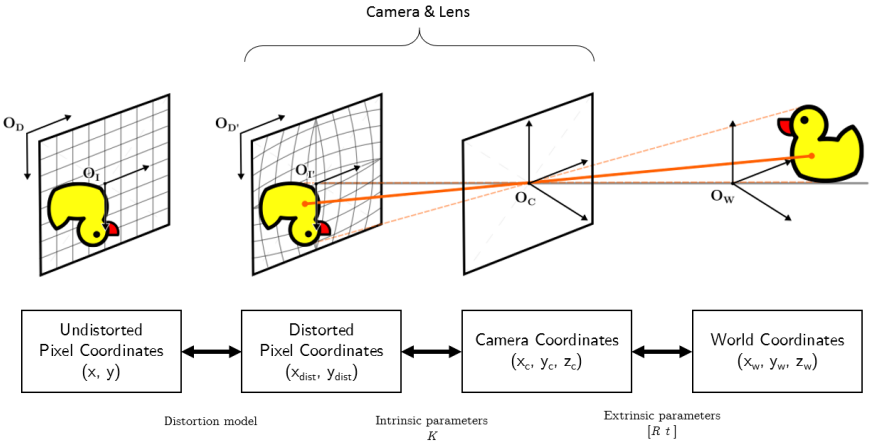


Figure 3.2 Transformation chain from display frame (left) to world frame (right). The two middle sections represent the physical pinhole camera and lens.

Camera Calibration

Camera calibration is the process of calculating the camera intrinsic parameters and distortion coefficients, i.e., moving right to left in Figure 3.2. The work flow of camera calibration as described in [Zhang, 2000] and implemented in the ArUco library in [Bradski, 2000] can be summarized as follows: First, a set of real world coordinates is defined. This is commonly done using a flat checkerboard pattern of known dimensions, see Figure 3.3. Second, multiple snapshots of this set is captured using the camera. An algorithm is then used to detect the set of points in pixel coordinates for the image batch, proving the utility of a checkerboard pattern. Third, the pixel and world coordinates are used to estimate the intrinsic and extrinsic parameters. In the case of [Zhang, 2000], this is done analytically by solving for B in the equation

$$B = C^{-T}C^{-1}, \tag{3.27}$$

where C is the camera matrix. The interested reader is referred to [Zhang, 2000] for the full solution. Fourth, a Levenberg-Marquardt optimization problem is solved to minimize the reprojection error. This reprojection error is the euclidean distance between the reprojected image point and the measured image point when using a certain model and parameters. It is often used to quantify the quality of a calibration because a good model should be able to reproject points in their correct locations. Once again referring to Figure 3.2, reprojection is the process of going from left to right and back.

This project utilizes a set of software for camera calibration such the one described above and fisheye distortion calibration. For references to these, see Section 5.2.

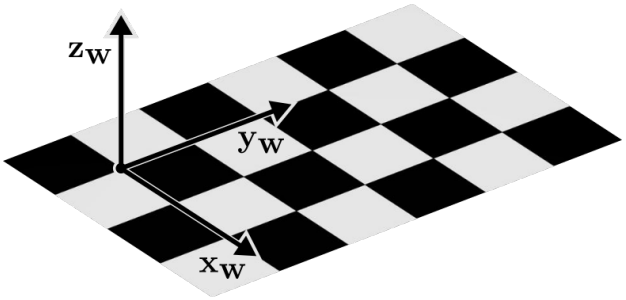


Figure 3.3 Checkerboard pattern in world coordinates. Here, the origin has been placed in an arbitrary corner and the board placed in the XY plane, i.e., $z_w = 0$.

Pose Estimation

Pose estimation, sometimes referred to as localization, in the context of computer vision and robotics is the process of determining the transformation between camera/robot and world frames. Multiple such methods exist and may be divided into analytic/geometric, genetic algorithms, and learning-based methods. For this real-time, on-line application an analytic method formulation will be used. A popular, well researched, and commonplace such formulation is the Perspective-n-Point (PnP) problem where a number of points n is used to determine the pose. Using the theory of this section, the UAV may be positioned relative to a 2D coded marker.

Perspective-n-Point The PnP problem is to determine the rigid-body transformation (see Section 3.1) from identified, observed features of a known, rigid-body. The letter n is the number of points used for setting up the solution, e.g., determining the pose of a feature using three points is called P3P. In fact, three is the minimum amount of points in order to extract a limited number of solutions, as illustrated in Figure 3.4 [Lu, 2018; LaValle, 2019].

There are numerous ways of setting up the solution of the PnP problem. The solver implemented in the OpenCV library when using four points (as required by the function to return a unique solution) is the one proposed in [Gao et al., 2003]. A requirement of PnP is that camera intrinsics are known or estimated.

Pose Ambiguity As presented above, the pose of a camera with respect to a planar marker can be be estimated using four points. However, due to finite resolution, image noise and inaccuracies in point detection, situations occurs where the pose is not uniquely defined. This is referred to as pose ambiguity or the ambiguity problem. The phenomenon is illustrated in Figure 3.5.

In most of these situations, the reprojection error of one solution is smaller than the other and the former can simply be used as the true solution. However, in some of the situations the reprojection errors are very similar, for example because the camera is far away from the features. Then, the solution to the problem is not trivial

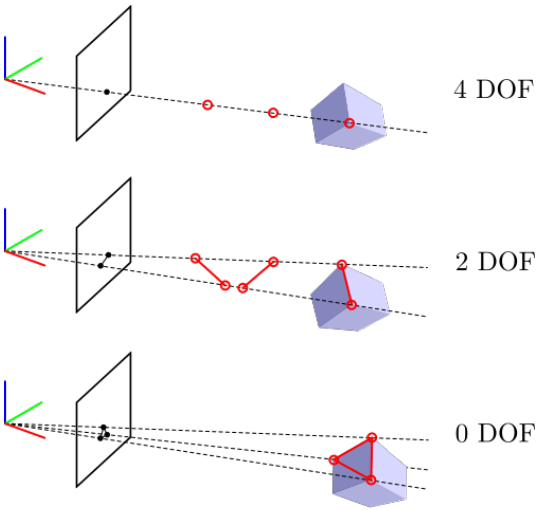


Figure 3.4 Illustration of degrees of freedom (DOF) in the PnP problem. Illustrated are the camera coordinate system, image plane and feature(s) as the corner(s) of a cube. In the top most example, $n = 1$ and the cube is free to rotate around the point in all axes and translation is limited to one axis (along the line). In the middle example, $n = 2$ and the features are free to revolve around the axis common to the two points and translate back and forth constrained by the lines. In the bottom example, $n = 3$ and a finite number of solutions are possible.

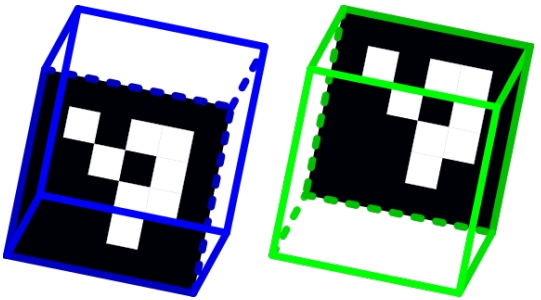


Figure 3.5 Pose ambiguity illustrated using an ArUco marker. The detected four corners may be the result of a camera pose towards the bottom left of the marker (blue), or top right (green).

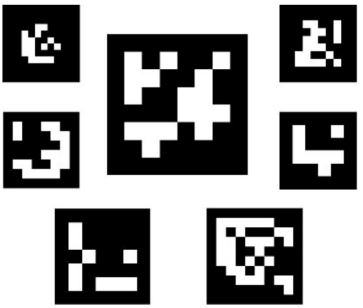


Figure 3.6 Examples of ArUco markers

but has been studied and circumvented by, e.g., marker tracking [Muñoz-Salinas et al., 2018] and motion models [Wu et al., 2012].

Computer Vision using OpenCV

The software tools used for image processing in this project are from OpenCV (Open Source Computer Vision Library). OpenCV is an open source library with tools that can be used in computer vision and machine learning [OpenCV, 2020a]. Since it is open source it is easily accessible and free. It has various functions suitable for the aim of this project. Among these is the ArUco library used for pose estimation.

The ArUco Library

ArUco markers are a type of binary squared fiducial markers commonly used in pose estimation. The ArUco library was developed in 2014 by Rafael Muñoz and Sergio Garrido and has since then been frequently used. An ArUco marker is a square with a black border and an inner binary matrix that determines the id of the marker, see Figure 3.6. The inner matrix can for example consist of 4x4 squares that results in an id composed by 16 bits. The ArUco-marker can be used to estimate the pose of the camera relative to the marker. The pose can be estimated only based on the four outer corners of the marker [OpenCV, 2020b].

3.3 Visual Servo Control

Visual servo (VS) control, or visual servoing, is the use of visual data, e.g., provnded by a camera, in automatic control applications. Historically, it has been used to control robotic arms and their servo motors, hence the name. However, the theory is equally applicable to other robots such as mobile robots or, as in this case, UAVs.

VS may be divided into two categories depending on where the visual information is gathered. Eye-in-hand configuration is where the camera is mounted to the

robot, as in this case when the camera is on the drone. Eye-to-hand configuration is where the camera is mounted in the environment and is able to observe the robot's movement, e.g., upward mounting of the camera to the base station.

Similar to other types of control, the methods associated with VS aim to minimize the norm of an error e as a function of time t which may be expressed as

$$e(t) = s(m(t), a) - s^*, \tag{3.28}$$

where $m(t)$ are the image measurements such as pixel coordinates, a is a set of system parameters such as camera intrinsics, and s^* is the reference, i.e., the target values of s [Chaumette et al., 2016]. There are two main approaches of determining s and its space, Image-Based Visual Servoing (IBVS) and Pose-Based Visual Servoing (PBVS). In IBVS the error is calculated by measurement of features directly available in the image. Such features may be lines, regions or pixel coordinates. Hence, control is often executed in 2D (the inherent dimensions of an image frame) or "2.5D" space, e.g., utilizing length of features to determine distance to them. On the other hand, in PBVS the error is calculated by interpretation of the visual information, i.e., estimating the complete 3D pose of a feature relative the camera by utilizing camera intrinsics. Control is therefore executed in 3D space.

3.4 Quadcopter Dynamics

The concept of a quadcopter consists of four motors with propellers and some circuitry to control them and provide power. The center of gravity should coincide with the origin of the body fixed frame. Two of the rotors rotate in clockwise (CW) direction and the other two in counter-clockwise (CCW) direction. The rotors are placed so that there is one CW rotor between each CCW rotor. This is to negate moment generated by their rotation so the quadcopter doesn't spin about it's own axis. The forces and moments affecting a quadcopter can be seen in Figure 3.7. The orientation of the coordinate system and the direction of roll (ϕ), pitch (θ) and yaw (ψ) can be seen in the same figure.

Each individual rotor n generates upward thrust, denoted f, along the z-axis of the quadcopter body. The moment generated by each rotor is denoted τ and actuates in the opposite direction of the rotors.

$$\begin{aligned} f_n &= k_n \Omega_n^2, \\ \tau_n &= b_n \Omega_n^2, \end{aligned} \tag{3.29}$$

where Ω is rotational velocity of the rotor given in rad/s, k and b are constants that depend on drone mechanics such like torque proportionality, propeller diameter, back-EMF, density of surrounding air etc. These constants are found empirically and the ones for the Crazyflie are given in the study of [Förster, 2015].

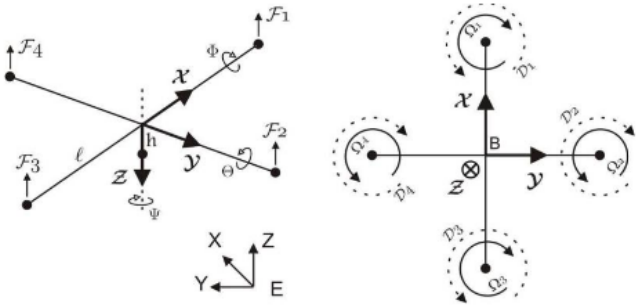


Figure 3.7 Orientation of the coordinate system and direction of the angles. The forces and moments acting upon a quadcopter when the propellers are rotating is also illustrated. Image from [Gopalakrishnan, 2017]

The total upward thrust, T_B, is given by summing all the thrust forces generated by each rotor

$$T_B = \sum_{n=1}^{4} f_n. \tag{3.30}$$

When a drone is hovering at a fixed position the force generated by the rotors are equal to, and aligned with, the gravitational force. To move the quadcopter upwards along the z-axis of the world frame it simply has to increase thrust. To move in x- or y- direction of the world frame it first has to orient itself in required roll and pitch angle. The forces along the x- and y- axis of the world frame is previously derived in [Gopalakrishnan, 2017] and can be described as,

$$\begin{aligned} F_x &= (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\sum_{n=1}^{4} f_n, \\ F_y &= (-\cos\psi\sin\phi + \sin\psi\sin\theta\sin\phi)\sum_{n=1}^{4} f_n. \end{aligned} \tag{3.31}$$

Rotor Transfer Function

The non-linear rotor model is derived in [Greiff, 2017] and can be described as a single-input single-output system according to Equations (3.32) and (3.33).

$$\begin{aligned} \dot{\mathbf{x}}_n^r(t) &= \mathbf{A}_n^r\mathbf{x}_n^r(t) + \mathbf{B}_n^r u_n^r(t), \\ \mathbf{y}_n^r(t) &= \mathbf{C}_n^r\mathbf{x}_n^r(t) \end{aligned} \tag{3.32}$$

Table 3.1 Rotor parameters for the Crazyflie from a study made in [Förster, 2015]

Parameter	K_t	K_e	b	J^+	J^-	R	L
Value	580	0.0011	0.10	0.031	0.13	2.3	0.12

$$\mathbf{A}_n^r = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -b/J^{\pm} & K_t/J^{\pm} \\ 0 & -K_e/L & -T/L \end{bmatrix}, \quad \mathbf{B}_n^r = \begin{bmatrix} 0 \\ 0 \\ 1/L \end{bmatrix}, \quad \mathbf{C}_n^r = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \tag{3.33}$$

The state vector, $\mathbf{x}_n^r = \begin{bmatrix} \mu_n(t) & \dot{\mu}_n(t) & i_n(t) \end{bmatrix}^T$, consists of the rotor position, $\mu_n(t)$, the rotational velocity, $\dot{\mu}_n(t)$ and the current, $i_n(t)$ is calculated from the input voltage, u_n^r. The rotor parameters $K_t, K_e, b, J^{\pm}$, R and L are related to motor current draw, drag, friction, inertia, resistance and inductance of the motor-rotor combination. The parameters have previously been identified in the study of [Förster, 2015] and can be seen in Table 3.1. The inertia, $J^{\pm}$ depends on whether the rotor velocity, $|\Omega|$ is increasing or decreasing according to Equation (3.34).

$$J = \begin{cases} J^+ & if \quad sign(\dot{\mu}(t) \times \ddot{\mu}(t)) > 0 \\ J^- & if \quad sign(\dot{\mu}(t) \times \ddot{\mu}(t)) < 0 \end{cases} \tag{3.34}$$

3.5 Crazyflie Control

In order to implement an external controller for the Crazyflie 2.1 it is important to understand how the internal control structure works. The information below is based on an article in [Richardsson, 2016].

Control Structure

The control structure in the firmware of the Crazyflie is visualized in Figure 3.8. The position of the quadcopter is estimated in the firmware by sensor inputs. The sensors are gyroscope, accelerometer and pressure sensor. With an additional Flow Deck two more sensors are added, a time of flight (ToF) sensor and an optical sensor used to measure movement in X, Y and Z-direction. The sensor data is then used in the State Estimator to estimate the state of the quadcopter. The state includes the orientation (roll, pitch, yaw), position and speed of the Crazyflie. The state is then used in the State Controller along with a Setpoint sent from the Commander. The commander can be a gaming controller, e.g., a PS3-controller or a mobile phone, it can also be a script running on a computer. The commander can communicate to the Crazyflie over BLE (Bluetooth Low Energy) or Crazyradio (radio protocol specific for Crazyflie, explained further in Section 4.2). The state controller finally sends a control signal to control the power to the motors [Richardsson, 2016].

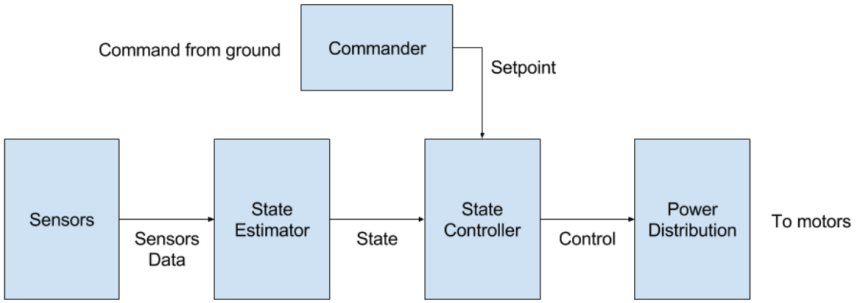


Figure 3.8 Position control structure in the firmware of the Crazyflie. Schematic from [Richardsson, 2016]

Extended Kalman Filter

The default state estimator of the Crazyflie (when the flow deck is on) is an extended kalman filter (EKF). It is a recursive filter that estimates the current state of the Crazyflie based on the incoming measurements (with a predicted standard deviation of the noise), model of the measurements and model of the system. An EKF is used for estimation of a dynamic system that lacks data due to, e.g., noise. Such system includes autonomous and assisted navigation systems, [Chadaporn et al., 2014], and is therefore suitable for the Crazyflie control structure. The incoming measurements are sensor-data from gyroscope, accelerometer and flow deck. The output from the Kalman Filter is attitude (roll, pitch, yaw), position (x, y, z) and velocity (x, y, z) [McGuire, 2020].

There are two models for EKF,

$$\begin{aligned} \text{State Model} \quad \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k + \mathbf{w}_k) \\ \text{Measurement Model} \quad \mathbf{z}_k &= h(\mathbf{x}_k + \mathbf{v}_k) \end{aligned} \tag{3.35}$$

where $\mathbf{x}_k$ is the state vector, $\mathbf{u}_k$ is the control data which is optional, $\mathbf{w}_k$ is the process noise vector, $\mathbf{z}_k$ i the measurement model vector, $\mathbf{v}_k$ is the measurement noise vector, $f(\cdot)$ is the process nonlinear vector function and $h(\cdot)$ is the observation nonlinear vector function [Chadaporn et al., 2014].

Cascaded PID Controller

The default state controller in the Crazyflie is Proportional-Integral-Derivative (PID) control for each desired state aspect. The control structure can be seen in Figure 3.9. The commander sends desired position or velocity set-points to the Position/Velocity controller. The position/velocity-mode of this first controller can be changed in the firmware. The output from that controller is a desired pitch and roll angle used as the input to the next PID controller, the attitude controller. These result in desired angle rates that are used as the input to the angle rate PID controller.

The result is the desired thrust for each motor that will be handled by the power distribution. This type of control structure is called cascaded PID control [Bitcraze, 2020c].

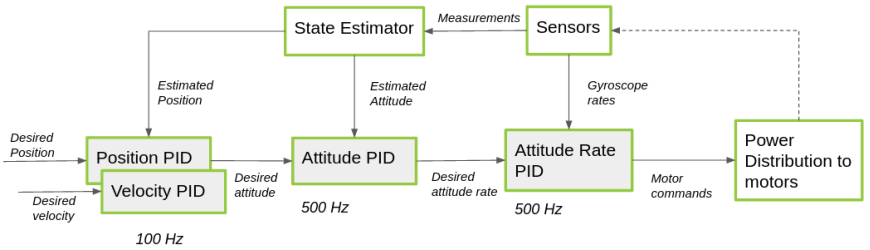


Figure 3.9 PID control structure in the firmware of the Crazyflie. Schematic from [Bitcraze, 2020c].

3.6 Ground Effect

Ground effect is defined as the increase in thrust when a rotor operates at constant power close to the ground. The change in thrust drastically affects the flight behaviour and limits the aircraft's ability for precision flight control and landing. The ground effect is related to rotor diameter and distance to ground, in this thesis referred to as height. The effect of it also varies between flights due to other factors such as type of ground and the speed of the vehicle [Wei et al., 2019]. It is a well investigated subject for conventional helicopters but not as commonly discussed among small-scale multi-rotor aircrafts. However, it has to be taken into account for multi-rotors in order to operate stably near ground.

Ground effect is also known as in-ground-effect (IGE) and the normal state when the aircraft is not in the IGE-zone is called out-of-ground-effect (OGE).

The ground effect appear due to the change of airflow under the rotor blades. When the quadcopter flies outside of the IGE-zone the airflow under the rotor blades is vertical. When the vehicle approaches the ground the airflow direction under the blades changes and is parallel to the ground. As the exit area for the airflow is reduced, the flow velocity is increased. The increased velocity is translated to an increase in thrust for the rotor blades [Wei et al., 2019].

In a study made at the University of California, [Wei et al., 2019], a model for the ground effect for a Crazyflie 2.0 was investigated. It showed that, close to the ground, the thrust generated by the rotors has a linear relation with the distance to the ground. This is different from the single-rotor model that indicates a quadratic relation between distance to ground and thrust. The study also shows that the model

switches from a linear to a quadratic function when the distance between the rotor blades is large enough.

There are different ways of compensating for the ground effect. In the same study as mentioned above a model reference adaptive controller (MRAC) was used for position control in the z-axis. A study from [He et al., 2019] showed through simulations that a nonlinear-disturbance observer can be used effectively to mitigate the ground effect. In another study, [Del Cont Bernard et al., 2017], where a dynamic analysis of IGE was made, they propose a gain scheduled attitude controller as a way of compensating for the ground effect.

3.7 Sequencing Control

In systems where the output should depend on the history of the system, *states* are introduced to implement sequencing controllers. In practice, these systems are often formalized using finite-state machines and visualized using their accompanying state graph. The states are linked through conditional transitions coupled to events or time. A Discrete Event System (DES) is typically event-driven, e.g., the push of a button or a measurement exceeding a value. Continuous-state systems are time-driven, e.g., the solution of a differential Equation [Årzén, 2014].

The states and transitions form a sequence net which can have serial and/or parallel structure. In a serial sequence one state is active at a time, and conversely, more than one is active in a parallel sequence. As mentioned, a sequence net can be visualized using a *state graph* (also referred to as state chart or diagram). The simplest example of a finite-state machine and its graph is presented in the form of a Moore Machine in Figure 3.10.

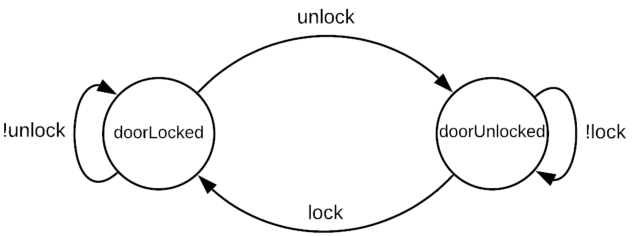


Figure 3.10 A Moore finite-state machine example of a door lock. The state of the door is either locked or unlocked. Events/conditions which trigger a transition is illustrated as an arrow with accompanying name.

Sequencing control in beyond trivial applications such the door lock example need to be extended in order to be practically useful. These extensions include hi-

erarchy, concurrency and history/memory. UML state diagrams solve these issues, amongst others, by using hierarchically nested states, orthogonal regions and extended states respectively.

A sequencing controller is necessary in this project as the landing procedure is a sequence of events, e.g., taking off, descending, and landing, whilst executing feedback control. Additionally, state machines are excellent ways to structure code and has proven most useful in the implementation. An example code skeleton in Python can be seen in Listing 3.1 where three states, e.g., controllers, make some "calculations" at a fixed frequency.

```
from random import random
from time import sleep
from time import time

class FiniteStateMachine():

    def state0(self):
        print ("state0")
        sleep(random()) # some calculations...
        if random()>.5:
            return self.state1
        else:
            return self.state2

    def state1(self):
        print ("state1")
        sleep(random()) # some calculations...
        if random()>.5:
            return self.state0
        else:
            return self.state2

    def state2(self):
        print ("state2")
        sleep(random()) # some calculations...
        if random()>.5:
            return self.state0
        else:
            return None

    def __init__(self):
        T = 1.0
        state=self.state0()     # initial state
        while state:            # state machine loop
            t0 = time()
            state = state()
            duration = time() - t0
            if (duration < T):
                sleep(T - duration)

        print ("Done with states")

fsm = FiniteStateMachine()
```

Listing 3.1 Finite-state machine code skeleton example in Python based on [Klaffenbach, 2010]. The machine is initialized and run in the `__init__` method, the states execute at a frequency of 1 Hz as made sure by the measurement of execution time (given no calculation exceeds the period $T = 1.0$ s).

3.8 Wireless Charging

There are some different kinds of wireless charging systems, that are all well described in the article of [TDK, 2017]. The non-radiative type transferring energy through an electrical or magnetic field and the radiative type, transferring energy through radio waves or laser. The non-radiative type has higher efficiency but has a limited transfer distance. The radiative type, on the other hand, allows for longer transfer distance but lower efficiency due to energy loss from environmental conditions.

There are two kinds of non-radiative wireless energy transferring, magnetic field coupling and magnetic resonance. The latter one is more complicated and costly compared to magnetic field coupling and is therefore not as common in consumer devices.

Magnetic Field Coupling

Magnetic field coupling works by transferring electric energy from a transmitter coil (the charger) to a receiver coil. A current in the transmitter coil generates a magnetic field that induces a current in the receiver coil. The process is illustrated in Figure 3.11. The induced current is used to charge a battery or a power load device.

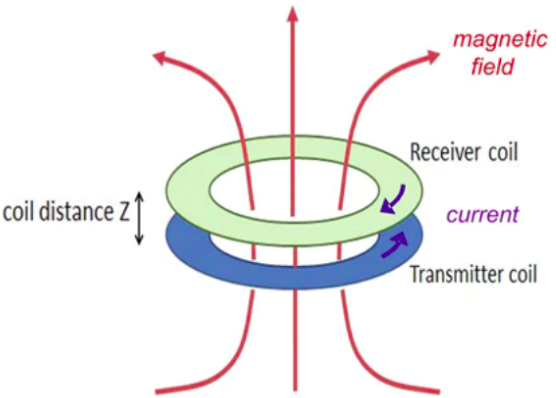


Figure 3.11 How inductive charging works. Illustration from [Wireless Power Consortium, 2018a]

The process of how this magnetic field appears is well described in a study about wireless inductive charging in [Macharia, 2017]. The Biot-Savart Law, in Equation (3.36), describes the magnetic field generated by a current-carrying wire and it used to calculate the strength at different points.

$$B = -\frac{\mu_0}{4\pi}\oint_l \frac{Idl \times e_r}{r^2} \tag{3.36}$$

The magnetic flux is represented by B, μ_0 is the permeability, Idl is the linear-current-element in the wire, r i the full displacement vector from the wire to the point, in which the field is being computed and e_r is the unit vector of r.

In most hand-held wireless power transfer devices the transmitter coil is circular as illustrated in Figure 3.12.

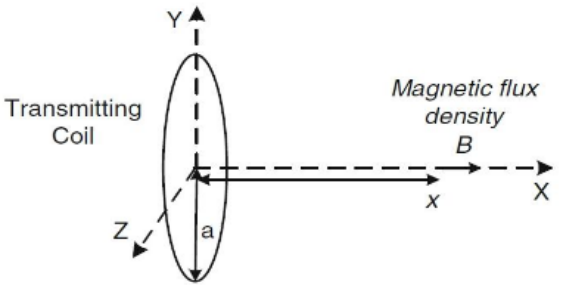


Figure 3.12 Magnetic flux generated by a circular transmitter coil. Illustration from [Macharia, 2017]

The magnetic flux generate by a circular coil at point x in Figure 3.13 can be expressed by Equation (3.37).

$$B_x = \frac{\mu_0 NIa^2}{2(a^2+x^2)^{3/2}} e_x \tag{3.37}$$

Where N is the number of wire turns in the coil, I is the current in each individual turn, a is the radius of the coil, x is the distance from the center of the coil to the point x and e_x is the unit vector between point a and point x.

The total magnetic flux captured by the receiver coil can be expressed by Equation (3.38).

$$\Phi_m = \int_S B \cdot dS, \tag{3.38}$$

where B is the magnetic flux density generated by the transmitter coil and S is the surface area of the receiver coil. In accordance to Faraday's law of induction the induced voltage, $V(t)$, in the secondary coil is given by Equation (3.39).

$$V(t) = -\frac{d\Phi_m(t)}{dt} \tag{3.39}$$

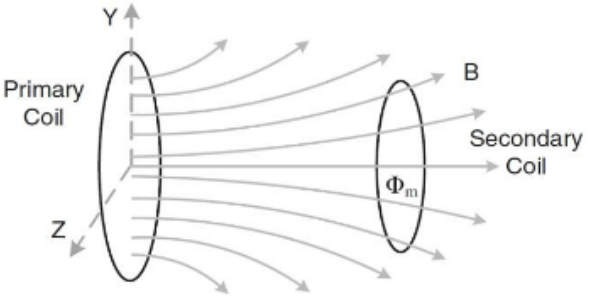


Figure 3.13 Magnetic flux captured by the secondary coil. Illustration from [Macharia, 2017]

where $\Phi_m(t)$ is the total magnetic flux crossing the receiver coil. The induced voltage generates a current and also a magnetic field with a polarity that opposes the magnetic field from the transceiver coil.

As understood from the equations above, the magnetic flux and therefore the coupling efficiency is affected by coil geometry, coil material and design and also by the orientation of the two coils relative each other. Figures 3.14 and 3.15 shows how the transfer efficiency drops as the distance between a transceiver and a receiver coil increases.

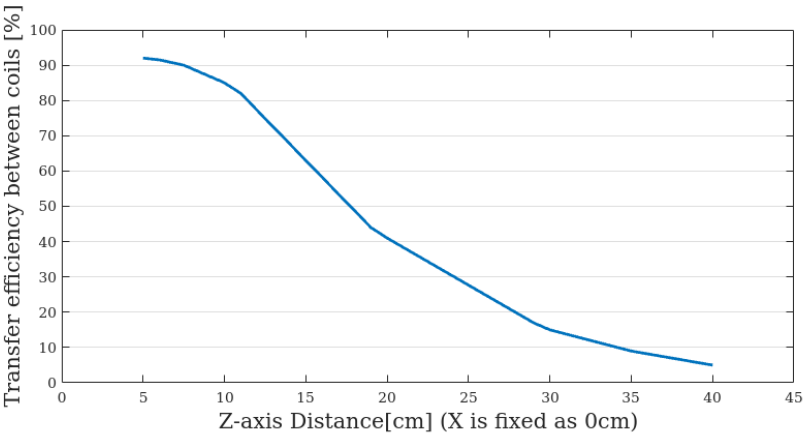


Figure 3.14 How increased vertical displacement between the TX ad RX coils affects the transfer efficiency. The graph is deducted from [TDK, 2017]

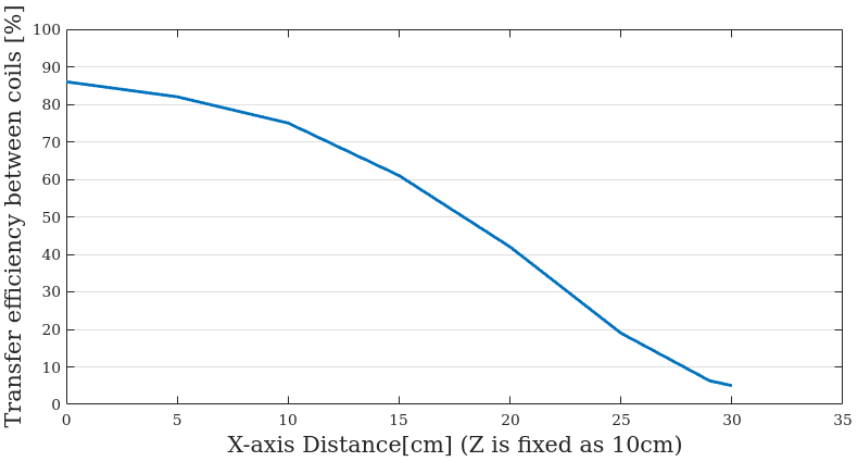


Figure 3.15 How increased horizontal displacement affects the transfer efficiency. The graph is deducted from [TDK, 2017]

Qi-standard

There are different kinds of standards for inductive charging products. The most common one is the Qi-(pronounced "chee") standard created by WPC (Wireless Power Consortium) [Mearian, 2018]. The WPC is an open collaborative development group with more than 650 member companies from around the world. Qi standard enables pad-style charging for power of 5-15W delivered to small personal electronics [Wireless Power Consortium, 2018b]. It is mainly used to charge smartphones but the number of compatible consumer devices are growing. The Qi standard involves different tests to make sure the product is safe to use and reliable.

4

Implementation

In this chapter the implementation of the final system setup is presented. Firstly, all of the included hardware parts are presented followed by a more thorough explanation of each part. The implementation of wireless charging and how that sets a target for the landing precision is then explained. Afterwards, the design requirements and choices of the base station is described. Further, the vision pipeline and camera setup is described and motivated.

After the presentation of the hardware, the implementation of the control is explained. Firstly the position controller and secondly the sequencing controller. Lastly, there is a reference to the Git repository where all of the code for the controllers can be found.

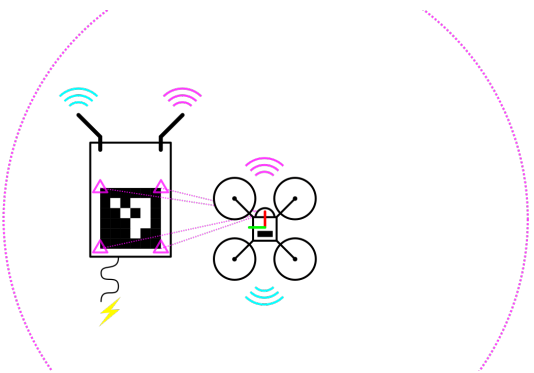


Figure 4.1 Simplified system overview. Object to the left is the base station containing marker, processing unit, video receiver and radio communication device. Object to the right is the drone. Base station and drone communicate via two links: the 5.8 GHz video feed is sent from UAV to base station (magenta) and control signals are sent from base station to UAV (cyan). The outer circle represents the camera FOV.

4.1 System Overview

A system overview of UAV and base station may be seen in Figure 4.1. The included components are further described in succeeding sections.

4.2 Hardware

The specific components used in this project can be seen in the Bill of Materials (BOM) in Table 4.1. They are explained more thoroughly further down in this section.

Table 4.1 BOM

	Part name	**Weight** [g]
UAV		
Quadcopter	Bitcraze Crazyflie 2.1	22
Battery	Bitcraze LiPo 0.925 Wh	8.4
Expansion deck	Bitcraze Flow deck v2	1.6
Camera	RunCam Nano3	1.1
Video transmitter (VTX)	RunCam TX25	2.5
Camera & VTX mount	(3D printed)	1.2
Battery splitter cable	(Self made)	0.6
		37.4
Base station		
Processing unit	Lenovo ThinkPad†	-
Radio communication dongle	Bitcraze Crazyradio PA	6
Video receiver (VRX)	Eachine ROTG02	35
Case	(3D printed)	-
Light absorbing paint	BLK 3.0	-

† 64-bit Linux Ubuntu 18.04 LTS, 8 GB, Intel® Core i5-6200U @ 2.30 GHz ×4

Crazyflie 2.1

The Crazyflie 2.1 (Figure 4.2) is, at the time of writing, the latest version of the mini quadcopter developed by Bitcraze. It weighs 27 g and is equipped with low latency and long-range radio and also Bluetooth LE. This allows for control through a mobile device or through the Crazyradio PA dongle plugged in to a computer [Bitcraze, 2020d].

Crazyradio PA

The Crazyradio PA (seen in Figure 4.2) is used for communication to and control of the Crazyflie. It is a long range open USB radio dongle based on the nRF24LU1+

(a) Image from [Bitcraze, 2020d] (b) Image from [Bitcraze, 2020e]

Figure 4.2 Crazyflie 2.1 and Crazyradio PA radio dongle

from Nordic Semiconductor. It has a 20 dBm power amplifier, a low-noise amplifier (LNA) and is pre-programmed with a Crazyflie compatible firmware. The power amplifier allows for communication in a range up to 1 km to the Crazyflie 2.1 [Bitcraze, 2020e].

Flow deck

The quadcopter in this project is equipped with the expansion deck Flow deck V2, seen in Figure 4.3. It can detect movement in any direction through its optical flow sensor and Time of Flight (ToF) distance sensor. The ToF sensor measures the distance to the ground, in this thesis synonymous to *height*, and the optical flow sensor measures movement along the ground in the X and Y direction. Due to the fact that it is able to detect movement, it is used to reduce positional drift relative to the ground [Bitcraze, 2020f].

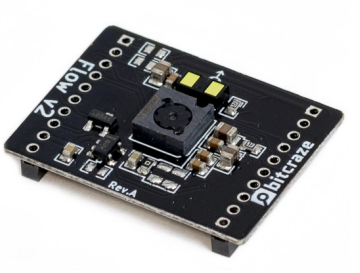
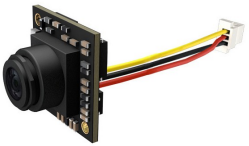


Figure 4.3 Flow deck V2. Image from [Bitcraze, 2020f].



(a) FPV camera RunCam Nano 3. Image from [RunCam, 2016].

(b) Video transmitter RunCam TX25. Image from [RunCam, 2019].

(c) Video receiver Eachine ROTG02. Image from [Eachine, 2020].

Figure 4.4 RunCam FPV camera and video transmitter.

Camera and Video Stream

Requirements The camera and VTX are carried by the Crazyflie and supplied with power by its 240 mAh LiPo battery. The battery has a nominal voltage of 3.7 V [Bitcraze, 2020a]. This requires lightweight components that doesn't require too much power and that can operate at 3.7 V. The Crazyflie can carry a load up to 15 g excluding expansion deck(s) [Bitcraze, 2020d]. The Flow deck weighs 1.6 g [Bitcraze, 2020f] which leaves a maximum weight of 13.4 g for the camera and the VTX. These weights are aimed for being minimized in order to maximize the flight time.

Camera The camera, RunCam Nano3 seen in Figure 4.4, is a FPV camera especially designed for drone racing. It has a 1/3" CMOS image sensor, a horizontal resolution of 800 TVL and a 160°FOV. It has a net weight of 1.1 g and requires 3-5.5 V DC and 110 mA@5V [RunCam, 2016]. This camera is chosen due to its low weight, high resolution and low power demand.

Video Stream Both the video transmitter (VTX) and video receiver (VRX) are also developed especially for drone racing. They can both operate at 5.8 GHz and have low latency [RunCam, 2020; Eachine, 2020]. The VTX, RunCam TX25 (seen in Figure 4.4), is chosen due to its low weight, streaming frequency and low power demand. The VRX, Eachine ROTG02 (seen in Figure 4.4), is chosen due to its bandwidth, low latency and USB video device class compatibility (UVC).

Wireless Charging

Bitcraze offers a wireless charging expansion deck ("Qi charger deck" [Bitcraze, 2020g]) with a RX coil that is mounted under the drone body. However, it is not possible to have both the flow deck and the Qi charger deck mounted on the Crazyflie

at the same time. Due to this limitation no charging pad nor receiver coil will be implemented in this project. However, the geometry of the base station is designed as if a transceiver coil was implemented.

As explained in Section 3.8, the performance of a wireless power transfer system depends on the distance from the TX coil to the RX coil and the performance decreases as the distance increases. The performance also depends on the coils' sizing, geometry and quality so a good estimate of the possible operating distances requires testing of the circuitry. Since there are no actual power transfer implemented in the station no such testing was performed. Instead, it is possible to rely on typical operating distances for inductive charging coils. According to [Voler Systems, 2020] a wireless charger operates typically over distances up to 4 cm. Therefore, the Crazyflie's centerpoint, which with the Qi charger deck mounted would have been the centerpoint of the RX coil, would have to land within a circular area of 4cm in diameter in order to charge. However, the closer to the centerpoint, the better efficiency of the power transfer.

Base Station

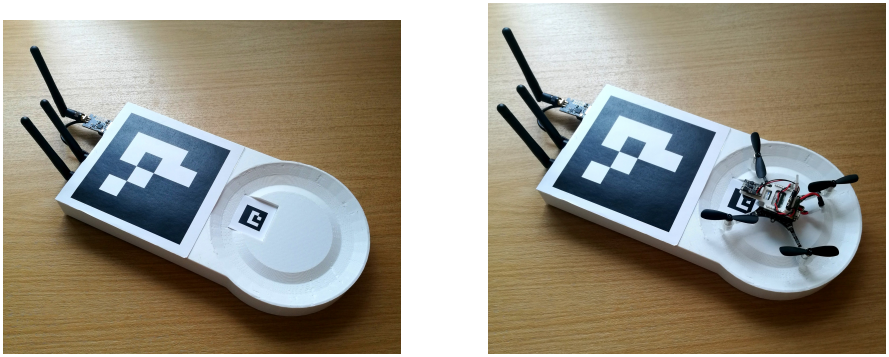
The main function of a base station is to give commands to the quadcopter and to charge it when it is in standby mode. A base station would therefore require a computer with some communication to the drone (a Crazyradio dongle if a Crazyflie is used) and some circuitry for charging.

The design of the landing pad can be seen in Figure 4.5, this prototype is 3D printed in ABS plastic. The round part is the landing area and under the larger ArUco tag there is enough space to fit a small-scale computer and a VRX. Such computer could for example be a Raspberry Pi that the design was adapted to for demonstration. Note that the Raspberry Pi was not implemented as the processing unit in this project but a laptop was used instead.

To improve the landing accuracy there is a slot with tilted sides in the landing area that will make the quad slide into the right position if the landing attempt was a bit off. Inspiration for the design was found in the article [Richardsson, 2018].

The circular surface in the middle of the landing area represents an inductive charger and is in this design adapted to fit the dimensions for the IKEA charger "Nordmärke" [IKEA, 2020]. If the drone would have the Qi charger deck mounted the idea is that this should touch the charging surface to minimize the vertical distance between the coils. In this design there is a clearance between the bottom of the Crazyflie to the charging surface when it is landed. This is to avoid damage of the flow sensor since the flow deck is mounted instead of the Qi charger deck.

The small tag is lowered to the same level as the slot to make it possible for the camera to detect the tag even when it is landed on the station. The size selection of the two ArUco tags is explained in Section 4.3.



(a) (b)

Figure 4.5 The 3D-printed landing pad with ArUco-tags, the Eachine VRX, the Crazyradio dongle and a Raspberry Pi placed under the large tag.

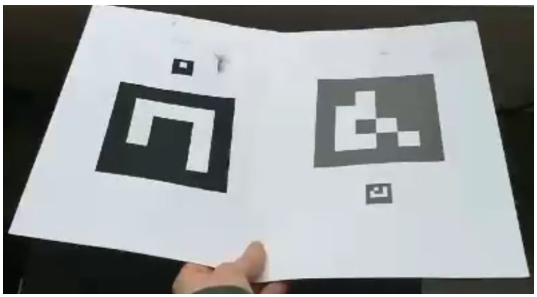


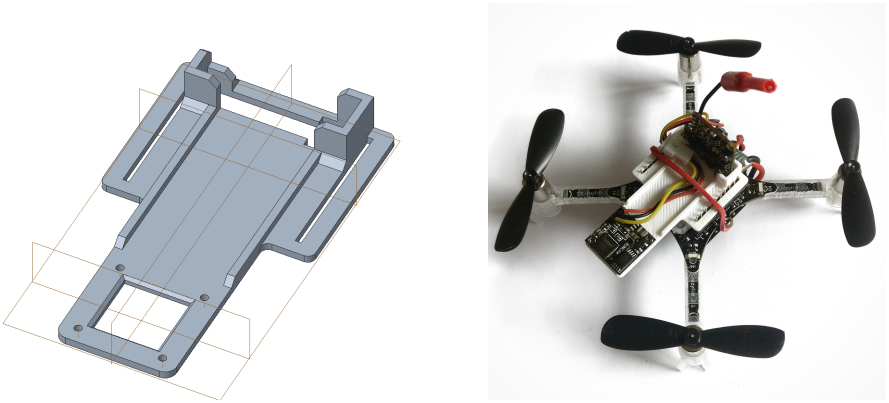
Figure 4.6 The marker to the left painted with BLK 3.0 and the marker to the right printed with a normal printer. The photo is taken as the tags are directed towards a light source.

BLK 3.0

To reduce light glare in the ArUco tags the light absorbing paint BLK 3.0 was used. It was painted with a brush on top of the ArUco tags that were printed on a white paper. The matte black paint absorbs up to 99% of visible light according to the producer [Semple, 2020]. The paint reduced the glare significantly as can be seen in Figure 4.6

4.3 Vision Pipeline

The FPV camera and video transmitter was fixed to the drone in a downwards facing fashion using a custom made 3D printed mount in ABS plastic, see Figure 4.7.



(a) CAD model

(b) Photo of complete setup

Figure 4.7 Camera and VTX mount.

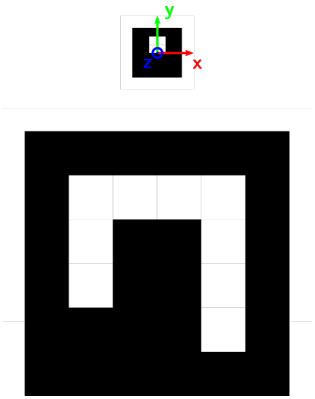


Figure 4.8 Custom marker board, here using marker id number 10 (bottom) and 17 (top) of the ArUco marker library *DICT_4X4_50*. The origin of the board is placed in the middle of the small marker.

The video feed is analysed at an average frequency of 30 FPS (Hz) using the UVC VRX. Using the ArUco library [Garrido-Jurado et al., 2014] with calibrated camera intrinsics as input, an ArUco marker is detected and its corners extracted. The pose of the marker can then be estimated as described in Section 3.2. Using the transformations described in Section 3.1, the pose of the camera with respect to the world frame, i.e., the marker frame, can be calculated. This pose is then fed to the controller, described in Section 4.4.

Two ArUco markers from the marker library *DICT_4X4_50* of different sizes

were used to create a custom marker board with origin placed in the middle of the small marker, as seen in Figure 4.8. The side lengths were determined from a preliminary test. The small marker with side length 21 mm can be detected when the UAV has landed within a 10 mm radius, approximately. The big marker with side length 112 mm can be detected reliably at a distance of 1 m. The displacement between the markers in x,y direction was arbitrarily chosen as 0 and -90 mm, respectively.

As described in Section 3.2, the problem of pose ambiguity is present at far distances. To circumvent this, pose estimation is only activated at close distance flight. Position control outside this half sphere is implemented by marker identification and using its corner coordinates $u_{x,y}^{i=1..4}$ in the FOV as reference, further described in Section 4.4.

4.4 Control Implementation

Position Controller

A necessity for autonomous landing is position control. In this section, the implemented position controller is described. As described in Section 3.5, the Crazyflie implements a velocity controller. Due to the natural, integrating relation between velocity and position, the proposed position controller is a P controller. This is sufficient if the velocity controller is assumed to be well-performing. This decision is further discussed in Section 7.1.

Reference signal generation of the position controller was implemented using a hybrid approach of IBVS and PBVS. As explained in Section 3.2, pose estimation becomes uncertain far away from the marker while the marker identification was stable. Therefore, IBVS was used by defining s in Equation (3.28) as the average of the coordinate values of the marker corners and using the image center u^c as reference, yielding the error:

$$e_{x,y} = u_{x,y}^c - \frac{\sum_{i=1}^{4} u_{x,y}^i}{4}. \tag{4.1}$$

Where pose estimation was stable, PBVS was used by estimating the pose as described in Section 3.2. The state variable s was defined by extracting the x and y coordinates of the pose of the camera with respect to the marker. In other words, the position reference was simply the distance between the marker origin and camera in the horizontal plane:

$$e_{x,y} = 0 - s_{x,y}. \tag{4.2}$$

The reference height z_{ref}, i.e., vertical distance to the ground, is determined by the sequence controller described below as a function of state. A simple block diagram of the controller illustrating its relation to the velocity controller described in Section 3.5 can be seen in Figure 4.9.

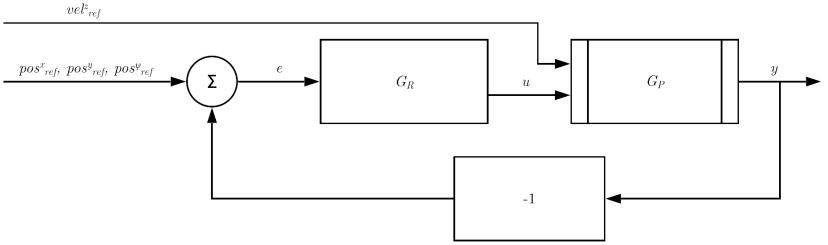


Figure 4.9 Block diagram of position controller where G_R is the P controller, G_P is the pre-implemented velocity controller.

Sequencing Controller

To implement a complete landing procedure, from initial connection and take off to position control and landing, sequencing control was used to generate reference signals of the above mentioned position controller. The controller consists of a finite-state machine with the following states:

- S0: **Initializing** - Connection to the UAV and camera module is established.
- S1: **Taking off** - The UAV ascends with a specified velocity. A transition is triggered when a specific height is reached.
- S2: **Seeking** - The UAV "searches" for a marker. No search algorithm is implemented as the marker is assumed to be within LOS, see Section 1.4.
- S3: **Nearing** - IBVS position control is activated. X and Y position reference is the image center, Z velocity reference is proportional to the radial distance.
- S4: **Approaching XY** - PBVS position control is activated. X and Y position reference is the marker origin, Z velocity reference is zero, and yaw reference angle is 0°.
- S5: **Approaching XYZ** - Like S4 but Z velocity reference is a constant value (down).
- S6: **Descending** - Like S5 but with higher gain controller(s) and different vertical speed.
- S7: **Landing** - Like S6 but with higher gain controller(s) and different vertical speed. The UAV lands straight down when a specified height is reached.
- S7X: **Retrying** - The UAV ascends with a specified velocity. A transition to state S5 is triggered when a specific height is reached.
- S8: **Terminating** - Connection to the UAV and camera module is terminated.

Transitions between states are driven by the height of the UAV, except in states S0 (not yet taken off), S4 (yaw correction), and S8 (landed). The higher gains are motivated by the diminishing noise as the distance to the marker decreases. If state S7

Table 4.2 Position control signal values of state machine states.

State	Controller		
	x,y	z	ψ
S0	N/A	N/A	N/A
S1	$u = 0.0$	$u =$`TAKEOFF_VEL`	$u = 0.0$
S2	$u = 0.0$	$u =$`SEEK_VEL`	$u = 0.0$
S3	$u = K_x e$	$u = -K_z/\sqrt{x^2+y^2}$	$u = 0.0$
S4	$u = K_x e$	$u = 0.0$	$u = K_\psi e$
S5	$u = K_x e$	u=`APPROACH_VEL`	$u = K_\psi e$
S6	$u = 2K_x e$	u=`DESCEND_VEL`	$u = K_\psi e$
S7	$u = 4K_x e$	u=`LANDING_VEL`	$u = 2K_\psi e$
S7X	$u = 0.0$	$u =$`TAKEOFF_VEL`	$u = 0.0$
S8	N/A	N/A	N/A

is reached and the landing cannot be autonomously confirmed, i.e., camera cannot detect the marker, a retry-protocol is run where the UAV takes off to a low height and proceeds directly to state S5. The controller control signal as a function of state is described in Table 4.2.

The state machine was written integrated in a multi-threaded Python program where one thread handles image analysis and one executes the state machine and control. The script is visualized in Figure 4.10 in the form of a UML diagram.

Git Repository

All code of above implementation can be seen in the Github repository found at `https://github.com/agrensimon/ugly-dockling`.

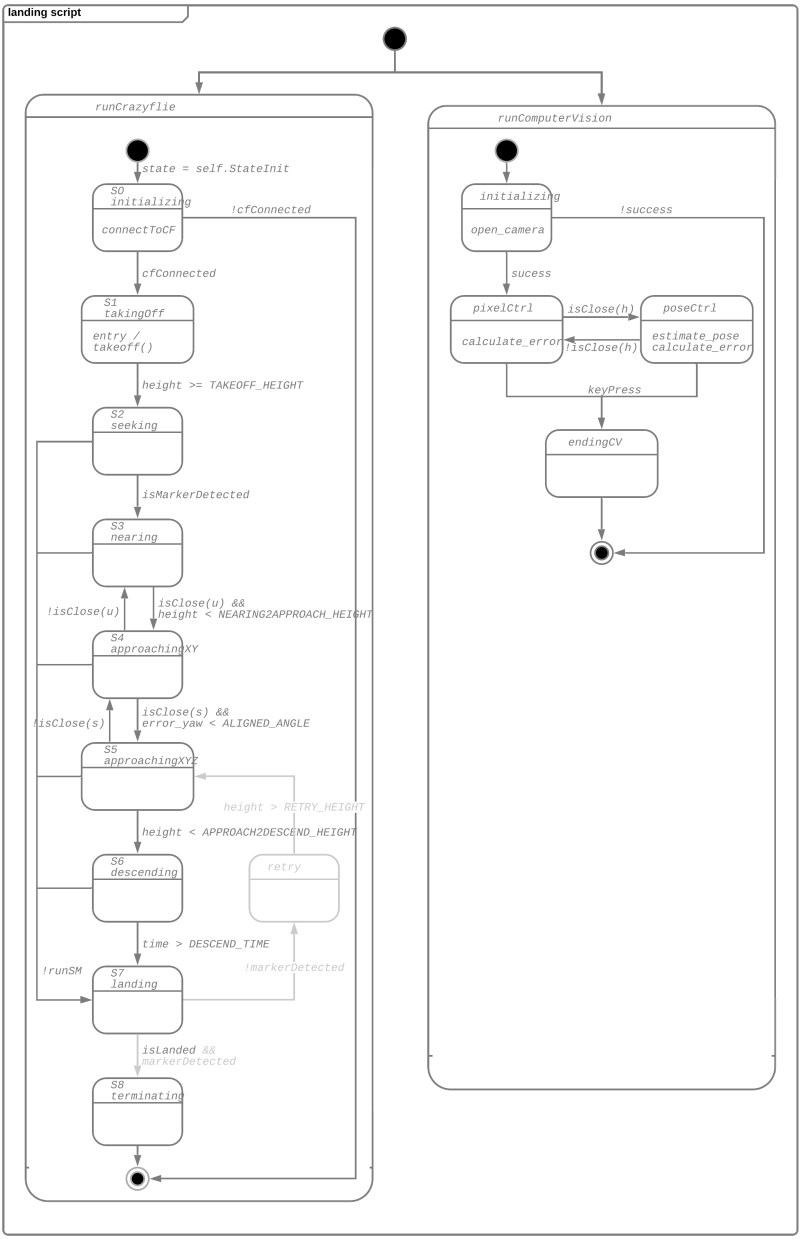


Figure 4.10 UML state diagram of sequence controller Python implementation.

5

Experiment

This chapter presents the approach of the experiments performed for system evaluation. The results of these experiments are presented in Chapter 6.

The experiments are presented in the following order: Battery life, camera calibration, pose estimation, ground effect characterization, controller performance, landing precision & accuracy and base station landing. Lastly there is a reference to the Git repository where the code used in the experiments can be found.

5.1 Battery Life

The impact of the camera module on battery life was tested in a short experiment. A unused battery was equipped and the flight time was measured during stationary hover at height $h = 1.0$ meters for three different cases: (1) Crazyflie and Flow deck, (2) like 1 but with camera module mounted and turned off, (3) like 2 but with camera module turned on. Before each test the battery was fully charged and the UAV was flown until the voltage reached below 3.0 V.

5.2 Camera Calibration

The camera was calibrated as described in Section 3.2. All evaluated models are presented in Table 5.1. The software used was the Matlab Camera Calibrator App (pin hole, rad-tan & fisheye model) [The MathWorks Inc., 2020], and Kalibr (equidistant fisheye model) [Furgale et al., 2012; Furgale et al., 2013].

The image batch was the same for all models (except in Kalibr), contained 175 images and was taken of the checkered calibration board at varying areas in the field of view and at varying angles. Images where the point detection algorithm failed were excluded, as exemplified in Figure 5.1. In the case of Kalibr, the process described in [Furgale et al., 2020] was followed. The camera calibration was evaluated by studying reprojection errors of the models. As model E was not calibrated using the same software, the image batch and results are not directly comparable (Kalibr defines the reprojection error in non-absolute x,y terms). To be able to compare all models, a visual inspection of undistorted images was made.

Table 5.1 Evaluated camera models

Model	Description
A	Pinhole camera with 2 radial distortion coefficients
B	Pinhole camera with 3 radial distortion coefficients
C	Pinhole camera with 3 radial distortion coefficients including skew and tangential distortion estimation
D	Fisheye camera (Matlab)
E	Equidistant fisheye model (Kalibr)

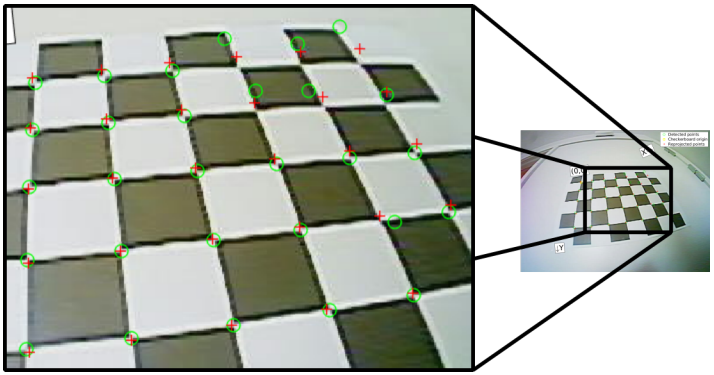


Figure 5.1 Incorrect detection of checkerboard corners (green circles), resulting in incorrect parameter estimation.

5.3 Pose Estimation

A simple test was conducted to determine the limitations of the pose estimation. The safe operation distance was found by studying measurement noise of the estimated pose. The camera was placed at a specified distance l to a marker in a fixed fashion and the 6 DOF pose estimation of the camera with respect to the marker (executed at 30 Hz) was logged during 10 seconds, see Figure 5.2. The test was then repeated for the distances $l = [0.07, 0.1, 0.2, 0.3, \ldots, 1.4]$ m.

5.4 Ground Effect Characterization

An experiment was conducted to analyze the presence of the ground effect and to validate the linear thrust-height model proposed in [Wei et al., 2019]. The setup was setup in a similar fashion to cited work and can be seen in Figure 5.3 which includes schematic and photo of UAV stand.

The Crazyflie 2.1 was connected to an external power supply to eliminate the

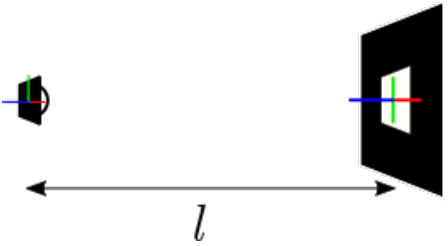
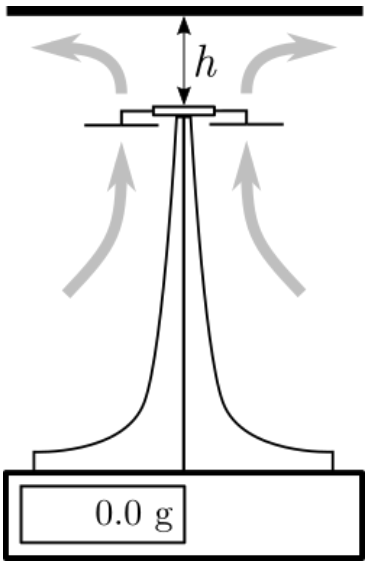
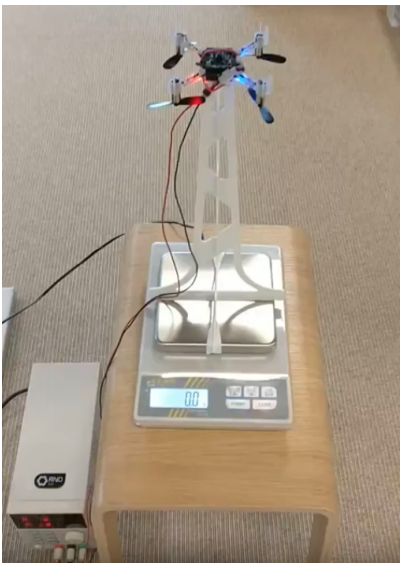


Figure 5.2 Pose estimation experiment setup. Left: camera and camera frame, right: marker and marker frame.



(a) Experiment setup schematic showing height h as distance measured by Flow deck ToF distance sensor between UAV and surface, stand allowing air to flow freely, and scale to measure thrust in grams.



(b) Experiment setup photo. External power supply is seen in bottom left.

Figure 5.3 Thrust test experiment setup.

effect of varying battery voltage. The firmware emergency stop which detects "tumbled state" was disabled to allow upside-down mounting without strange rate controller behavior. A surface was placed a distance h from the UAV. A PWM reference command signal from 0 to 100% in steps of 10% was sent to the PWM controller of all motors every 3 seconds. The PWM and height values were logged using the Crazyflie library logging framework and the thrust was logged manually using video playback. Then, the height h was changed and the process repeated for $h \approx [300, 280, 160, 120, 70, 20, 8]$ mm.

5.5 Controller Performance

The position controller performance was tested by conducting a series of step response tests. First, the UAV was flown at a height $h = 0.5$ m and a horizontal, radial distance $r = 0.5$ m away from the marker origin. The reference was then changed to $r = 0.0$ m and the step response was logged. The gain was set to $K_p = 0.8$.

Second, the UAV was flow at $h = 0.5$ m $r = 0.0$ m, this time with an angular yaw error of $\psi = 135°$. The reference was then changed to $\psi = 0°$ and the step response was logged. The gain was set to $K_\psi = 1.0$.

5.6 Landing Accuracy

An experiment was conducted to test the landing accuracy of the implemented control method. The UAV was placed on the ground approximately 0.5 m from the target landing spot with an angular error of 90°. The sequence controller was executed as described in Section 4.4. After completion the translation error in x,y and rotational error in yaw of the camera relative to the small marker was recorded, see $\overline{OC}$ in Figure 5.4. Due to the camera-UAV center offset, these measurements were also related to how close the UAV was to a hypothetical charging coil, see $\overline{AD}$. This distance was calculated according to Equation (5.1). The overall landing accuracy was evaluated by calculating the average and standard deviation of the radial and yaw errors.

$$|\overline{AD}| = |(\overline{OA} \begin{bmatrix} c_\psi & -s_\psi \\ s_\psi & c_\psi \end{bmatrix} + \overline{OC}) - \overline{OA}|, \tag{5.1}$$

where point $A = [0.0, -28.0]$ mm.

5.7 Base Station Landing

An experiment was conducted to test if the base station influences the landing precision and accuracy test described previously in Section 5.6. The experiment setup was identical of that of the test above, only difference being the addition of the base station. The sequence controller was executed and whether the landing was successful or failed was documented.

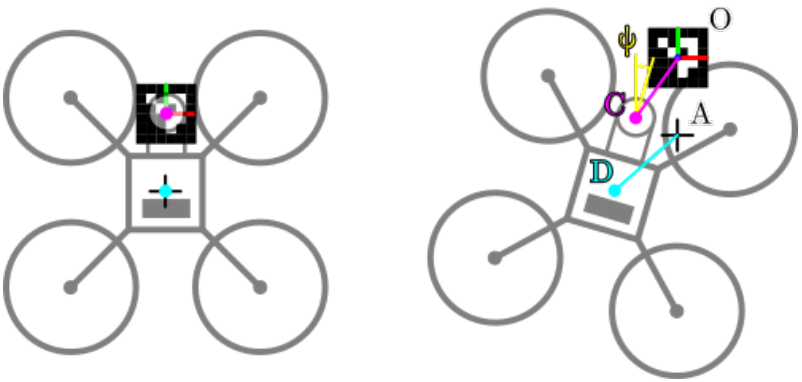


Figure 5.4 Left: Defintion of perfect landing. Right: Imperfect landing where $|\overline{OC}|$ is the radial error of the camera and $|\overline{AD}|$ is the radial error of the UAV. These differ unless the yaw $\psi = 0$ because of the camera-UAV offset $\overline{CD}$.

Git Repository

All code of above experiments can be seen in the Github repository found at `https://github.com/agrensimon/ugly-dockling`.

6

Results

In this chapter the results from the experiments in Chapter 5 are presented. These results are later discussed in Chapter 7.

6.1 Battery Life

The result of the battery life test explained in Section 5.1 of cases (1) Crazyflie and Flow deck, (2) as 1 but with camera module mounted and turned off, and (3) like 2 but with camera module turned on were 07:13, 05:50, and 04:48 [mm:ss] respectively. A graph of the battery voltages over time can be seen in Figure 6.1.

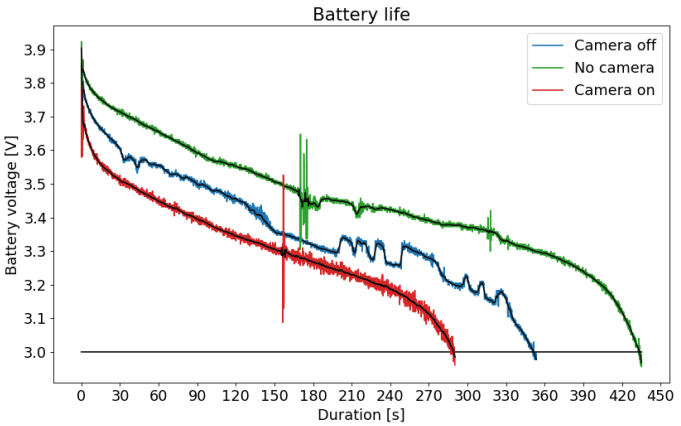


Figure 6.1 Battery life tests where the UAV was hovering (1) in standard configuration (green), (2) with camera module mounted and powered off (blue) and (3) with camera module mounted and powered on (red). Black lines show filtered (moving average) values.

6.2 Camera Calibration

As can be seen in Table 6.1 and Figure 6.2, the overall mean reprojection error was the least for models B and C (refer to Table 5.1 for model descriptions). Note the insignificant difference between including skew and tangential distortion in the model. Model A was not able to compensate for the considerable distortion of the wide FOV, as can be seen in Figure 6.3.

Table 6.1 Reprojection errors of models A-D.

Model	Mean overall error [pixels]	Standard deviation σ [pixels]
A	0.20	0.14
B	0.16	0.09
C	0.16	0.09
D	0.40	0.29

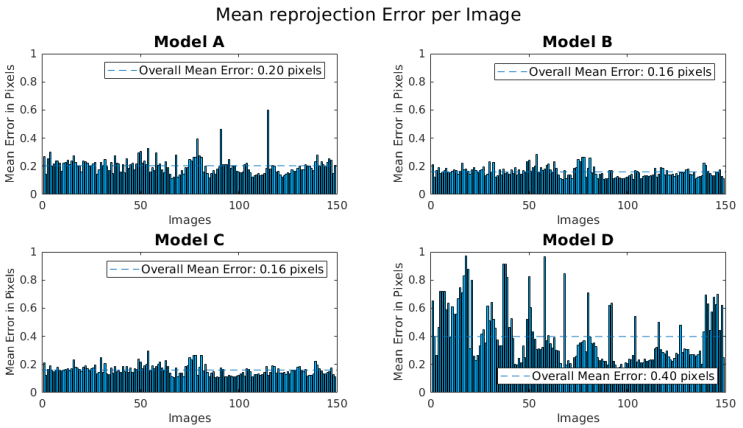


Figure 6.2 Mean reprojection errors per image of models A-D. To clarify, every bar along the x axis is the mean reprojection error of checkerboard corners (35 points) in one image.

The reprojection error of model E from Kalibr can be seen in Table 6.2. The visual inspection of undistorted images may be seen in Figure 6.4. Note the similarity in undistortion line straightness, but the much larger FOV of model E. The resulting camera matrix (see Equation (3.20)) and distortion coefficients (see Equation (3.24)) are presented in Equations (6.1) and (6.2).

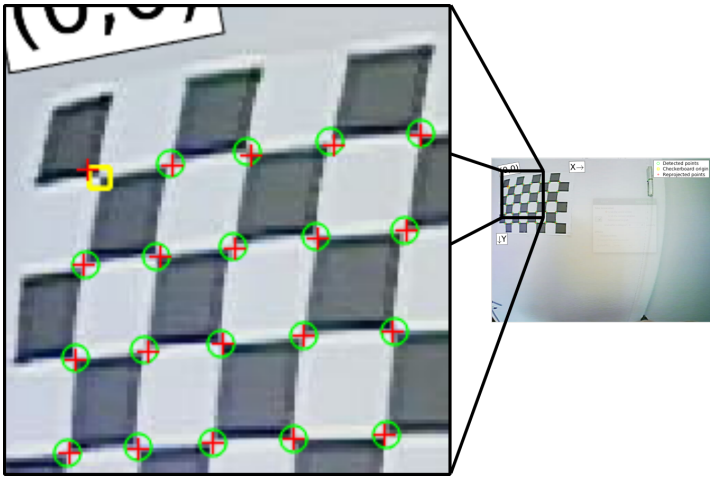


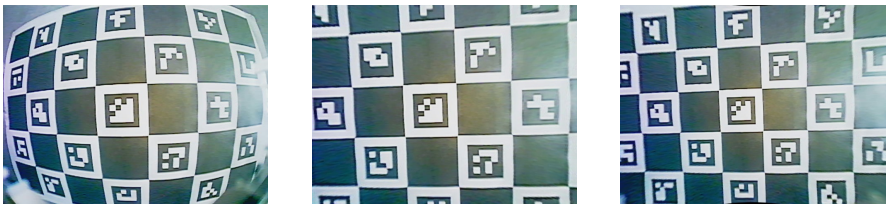
Figure 6.3 Example of an insufficient model. The reprojected point (red cross) in the top left deviates from the true point (yellow square) due to the considerable distortion in the corners of the field of view.

Table 6.2 Reprojection errors of model E.

Model	Error [pixels]	Standard deviation σ [pixels]
E	[0.000014, -0.000003]	+-[0.67, 0.95]

$$K = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 274.472 & 0 & 318.806 \\ 0 & 269.652 & 245.493 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

$$\begin{aligned} \theta_d &= k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9 \\ &= \theta + 0.13514225\theta^3 - 0.0147262\theta^5 - 0.04347712\theta^7 + 0.02299276\theta^9 \end{aligned} \tag{6.2}$$



(a) Raw image (b) Model B: Rad-tan (c) Model E: Equidistant

(d) Raw image (e) Model B: Rad-tan (f) Model E: Equidistant

Figure 6.4 Visual comparison of raw image, radial-tangential model and equidistant model.

6.3 Pose Estimation

The test described in Section 5.3 was conducted and the safe parameter of pose estimation was found to be when the marker sides were ≥ 50 pixels long. This equates to a distance between marker and camera of approximately ≤ 0.6 meters with the current camera and lens. Beyond this distance, the pose ambiguity problem became increasingly prevalent. Figures 6.5 and 6.6 show the pose tests of $l = 0.5$ m and $l = 1.2$ m where the marker side length was 58.76 and 24.15 pixels long, respectively.

In addition to the above result, it became evident that marker detection and succeeding algorithms, e.g., corner detection and pose estimation, were highly sensitive to light reflection. The interfering reflections occur when a light source is placed approximately behind the camera. This means ceiling light fixtures which are common in homes and office environments become a problem since they shine light at the marker paper which is reflected into the downward-facing camera on the UAV. This effect is mitigated by using a light absorbing paint as described in Section 4.2.

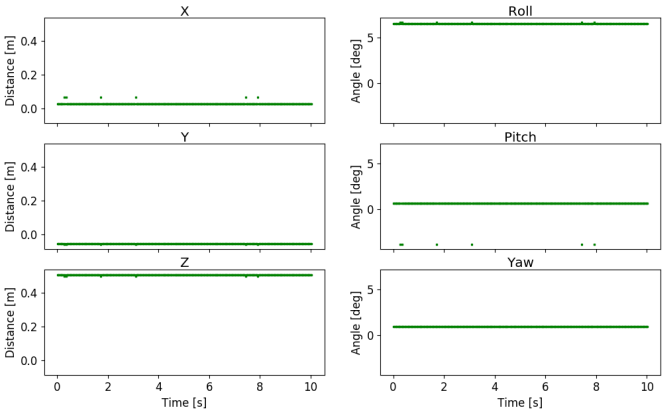


Figure 6.5 Pose test at $l = 0.5$ m where pose estimation is stable. Outliers may easily be filtered.

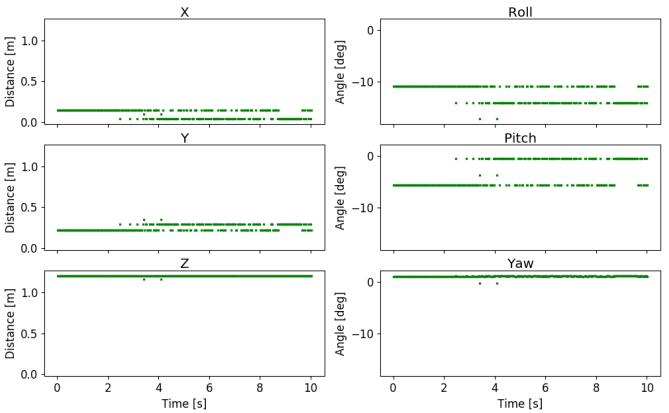


Figure 6.6 Pose test at $l = 1.2$ m where pose estimation is subject to ambiguity. From only studying the graph, it is impossible to conclude what the real pose is. The difference between the dominating 6D pose of the first half and second half of the test is [0.10, -0.08, 0.0, 3.18, -5.22, 0.0] m. This difference is considerable considering it may occur from frame-to-frame.

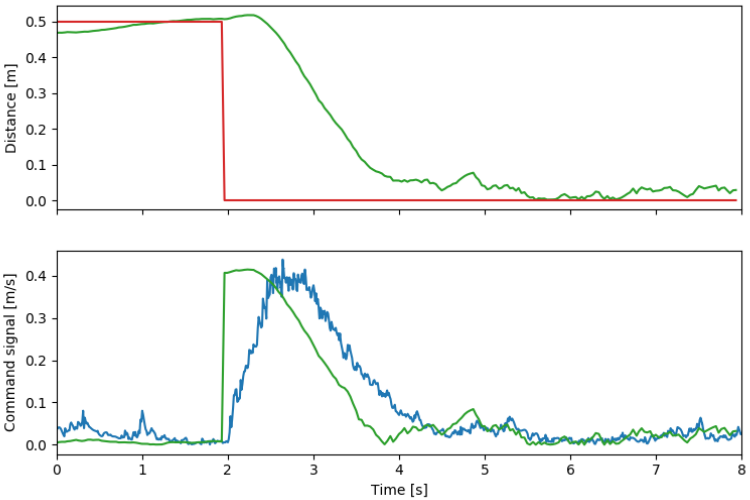


Figure 6.7 Step response of position controller (radial error). Top: reference (red) and estimated distance (green), bottom: control signal u (green) and UAV state velocity (blue).

6.4 Controller Performance

The result of the experiment described in Section 5.5 can be seen in Figures 6.7 and 6.8. The residual radial error of Figure 6.7 suggests the proposed P controller is unable to eliminate steady-state error but this is misleading. As *radial* distance is measured, the graph is unable to depict direction. In fact, it is movement and noise *around* the reference which is seen. The step response is therefore complemented with Figure 6.9 for a clearer picture.

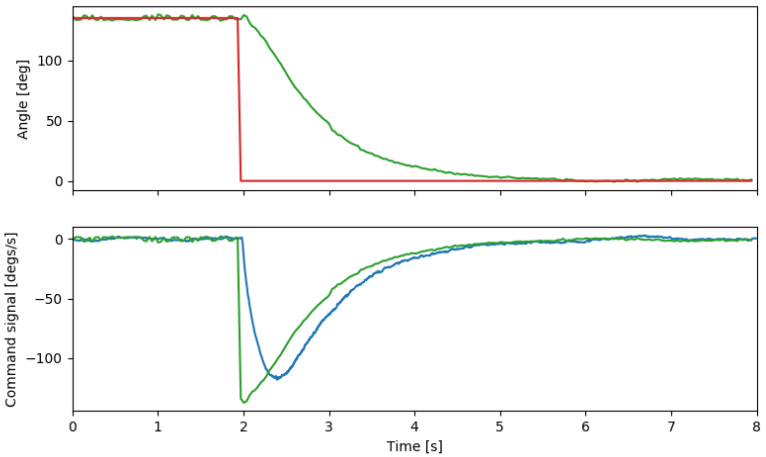


Figure 6.8 Step response of position controller (angular error). Top: reference (red) and estimated angle (green), bottom: control signal u (green) and UAV state velocity (blue).

6.5 Ground Effect Characterization

The experiment conducted as described in Section 5.4 brought unintended results. It became evident that the height measurement of the Flow deck ToF sensor used is unreliable under distances of approximately 4 cm. After further investigation, this is also found in the documentation [STMicroelectronics, 2018].

Measurement *over* this height is accurate and the result can be seen in Figure 6.10. The thrust-PWM relation seems to be independent of height which suggests the IGE zone is not observed. This is in accordance to [Wei et al., 2019] where IGE was first observed under approximately 60 mm. However, this test is unable to confirm neither the linear model nor a traditional exponential model.

It was decided to not extend testing of the ground effect further due to time constraints. The current implementation is dependent on height measurement of the Flow deck and due to the fact that the sensor is insufficient a workaround using another sensor(s) is first needed to be found. For further discussion see Section 7.2.

6.6 Landing Accuracy

The landings are visualized in Figure 6.11 and the average errors of the camera pose are found in Table 6.3. Additionally, the marker was detected in 76.5 % of landings. This motivates a "retry"-procedure as proposed in Section 4.4.

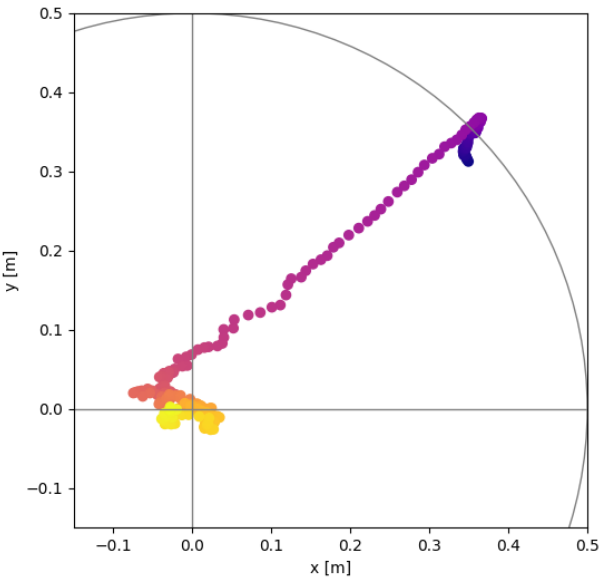


Figure 6.9 Step response scatter plot of pose estimation position in x,y. Blue: $t = 0$ s, yellow: $t = 8$ s.

Table 6.3 Landing experiment errors.

Error	Average	Standard deviation σ
Radial distance [mm]	11.22	6.59
Yaw [°]	0.973	2.150

In order to determine the quality of a wireless coupling the distance between the center of the drone body and the center of the landing pad on the base station was computed. The minimum, maximum and average distance can be seen in Table 6.4. As explained in Section 4.2, the typical operational distance for inductive charging is up to 4 cm. As seen in the table the maximum distance measured from the experiment is about 2.6 cm which is less than 4 cm and the landing accuracy is therefore showed to be satisfactory.

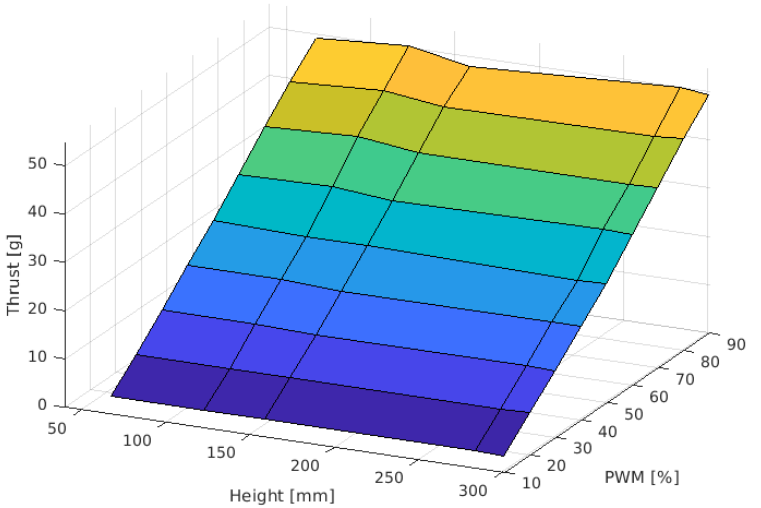


Figure 6.10 Graph of empirically extracted relation between height, thrust and PWM. A near linear correlation between PWM and thrust independent of height is observed.

Table 6.4 The minimum, average and maximum distance between the RX and the TX coil.

	Min	Average	Max
Distance RX to TX coil [mm]	3.6017	11.2029	25.6759

6.7 Base Station Landing

In 4 of 10 tests (40 %), the UAV landed in the designated groove. Using the base station measurements, this equates to lower accuracy than the previously conducted experiment without base station. Thus, it can be concluded that the current base station design affects the sequence controller negatively. This result is further discussed in Section 7.2.

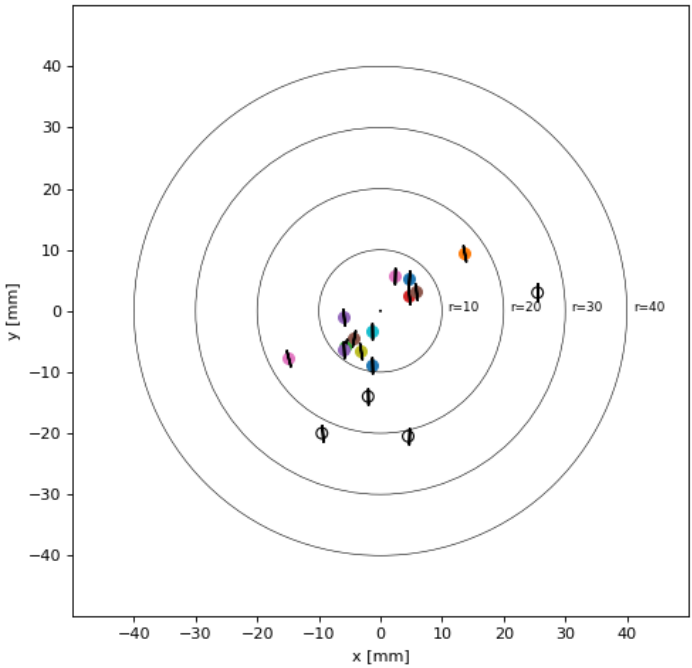


Figure 6.11 Landing accuracy experiment. Each circle represents a landing. Unfilled circles represent landings where the ArUco marker is undetected after termination. Lines represent landing yaw angle where reference (no error) is vertical. Angles are exaggerated by a factor 2 for clarity.

7

Discussion

In this chapter, the results of the previous chapter are discussed. Then, strengths and weaknesses of the project are identified along with the improvement that could be made. Further, future work on the topic is discussed.

7.1 Result Discussion

Battery Life

The test confirms that flight is possible with the camera module without any additional power source but at the cost of decreased flight time. For the purpose of having an indoor drone as a surveillance camera, flight time longer than 04:48 [mm:ss] is most likely needed. However, a bigger battery would also require more powerful motors in order to carry the increased weight. This is further discussed in Section 7.2.

Camera Calibration

The insignificant difference between excluding and including skew and tangential distortion in models B and C (refer to Table 5.1 for model descriptions) suggests that there is no skew and that the lens is aligned with the camera sensor. Visually comparing model B and E, it is evident that the undistortion performs similarly, i.e., straight lines in the real world are straight lines in the photos. However, the equidistant model is able to extract a larger FOV and is therefore deemed to perform better.

The calibration through Kalibr proved to be more time consuming compared to the other calibration methods. However, as a well performed camera calibration is crucial for a good pose estimation and as the resulting model is outperforming the others it is considered worth the time.

Pose Estimation

The pose estimation resulted in being reliable at distances $\leq$0.6 m which for the purpose of this study works well. However, this distance could be improved as is further discussed in Section 7.2.

Controller Performance

The step responses of radial and angular error show a satisfactory controller performance. The controllers were manually tuned and could therefore be further improved by more vigorous tuning. The inner velocity controller is initially unable to follow the control signal u, indicating an aggressive controller. Due to the fact that the Crazyflie is very stable, this is tolerated and flight stability is not compromised. As a result of the aggressive control, a time delay is also introduced between control and actual velocity.

As described in Section 4.4, the controller is a P-controller. In retrospect, we may conclude this is sufficient to eliminate the residual error with the current gains. However, if the velocity controller would not have performed as well as it does, an I-part would be needed to eliminate the error. Controller and measurement improvements are further discussed in Section 7.2.

Ground Effect Characterization

The conclusion regarding the ToF sensor from the ground effect experiment were somewhat disappointing. However, the method of the experiments is still considered reliable based on the study of [Wei et al., 2019]. Even though the results couldn't indicate any impact on thrust from the ground effect it is still considered an issue that should be taken into account for operation near ground. However, in this study the quadcopter does not operate for an extended period of time in the IGE-zone and the landing accuracy is therefore not severely affected.

Landing Accuracy

The landing precision resulted in being satisfactory in means of fulfilling the charging accuracy requirement of 4 cm. However, as mentioned in Section 3.8 the charging performance depends on more than the distance between the two coils. It is therefore not possible to conclude what efficiency would be possible with this landing accuracy without any power transfer measurements.

Base Station Landing

Landing on the base station resulted in being less accurate than landing on a flat surface. The reason behind this could be that the ToF sensor experiences more fluctuations in height over the base station. It could also be due to the fact that the base station is mainly white and lacks distinguishing features, reducing the performance of the flow sensor. Furthermore, the laser from the ToF sensor could also be re-

flected unusually due to the change in elevation. These results does not motivate for using a base station design similar to this as a mean of improving landing accuracy.

7.2 Improvements

Hardware

The visual pipeline hardware used in this project, including camera, transmitter and receiver, is inexpensive and unsophisticated. We believe better hardware could improve the pose estimation, and consequently the control implementation. An example is the camera resolution, here 640x480 pixels, which in today's standard is quite low. With higher image resolution, marker corners could be identified more accurately and thus produce less noise in the pose estimation. Alternatively, higher resolution could enable the use of smaller markers. This could both decrease base station footprint and permit the placement of a larger number of markers providing more points for pose estimation, both having obvious advantages. However, such cameras are digital and would rely on another type of video stream standard which could increase latency.

A height sensor with higher precision at short distances is crucial in order to investigate the ground effect. As described previously, the study of [Wei et al., 2019], concludes that IGE are detected for distances of approximately ≤ 60 mm. This sets the requirements for the height sensor.

As mentioned in Section 6.7, controller performance diminished after introducing the base station. The reason to this was further discussed in the previous section. An improvement to the current base station design could be a flat surface rather than the current groove for error correction. Furthermore, the size of the base station could be investigated. If the area of the base station would be increased the height sensor would get reading from outside of the station borders less frequently. This would decrease the times of readings from the edges of the base station. That could result in a height reading from the base station followed by a reading of the surface next to it which could affect the flight performance.

The ArUco markers have served their purpose well. The issue of light conditions affecting the tags was solved successfully by the BLK 3.0 paint. However, a further improvement to this could be displaying the markers on a LED screen. This would also, with some communication, allow for change to a smaller tag as the drone approaches the screen. In the study of [Acuña and Willert, 2018] a similar solution is presented. Moreover, lights could be added to the base station for constant light setting on the markers.

Software

The visual pipeline software could be improved in a number of ways. First, ArUco allows the user to define custom marker boards. The current implementation uses one marker at a time to determine the pose. A custom marker board would allow a

more stable pose estimation as more points (corners) are used for the PnP algorithm. This would improve accuracy when both markers are visible, i.e., during medium height flight.

Second, the pose estimation could benefit from marker tracking. Marker tracking, i.e., remembering previous poses, could hopefully mitigate the ambiguity problem as has been shown in [Muñoz-Salinas and Medina-Carnicer, 2020]. Pose ambiguity was a challenge which was faced throughout the project and limited the distance from which pose estimation could be used for control.

Third, as previously mentioned the Flow deck ToF distance sensor is unable to measure distances under 4 cm. A possible and promising improvement would therefore be to use the height measurement from the visual pipeline. This value, along with the yaw angle ψ, was perceived to be less noisy than that of e.g., roll ϕ and pitch θ.

Fourth, the controller could be improved by eliminating or compensating for the time delay by considering UAV acceleration limitations or implementing a Smith predictor, and improving signal filtering, e.g., using a Kalman filter.

Finally, a major software improvement would be to feed position data from the visual pipeline to the Crazyflie's on-board Kalman filter. By doing so the internal position controller of the UAV could be utilized. This could also render the Flow deck redundant hardware, resulting in a lower takeoff weight or enabling a Qi charger coil to be mounted.

We believe the above hardware and software improvements could be made based on the current state of implementation. These would likely improve the landing accuracy but to what extent is unknown.

8

Conclusions and Future Work

8.1 Conclusions

The concept presented in this study showed good results in landing accuracy. The controller showed satisfactory performance and the landing accuracy fulfilled the charging accuracy requirements.

Performing a thorough camera calibration through Kalibr resulted being worth the extra time required. The visual pipeline allowed for reliable pose estimation at distances ≤ 0.6 m. The experiment performed for ground effect characterization was unsatisfactory as it could not state any conclusions due to the ToF distance sensor that could not operate at distances under 4 cm. However, since the drone does not operate near ground for an extended period of time in this project the landing accuracy is not severely affected. The base station design decreased the landing accuracy even though it was designed to improve it through tilted sides and a designated groove.

The landing performance could be improved though better hardware as in camera, VTX and VRX. A ground effect characterization could improve flight performance and for that a better distance sensor is required. The base station design should also be overseen for a higher landing accuracy. Furthermore, a custom ArUco marker board would allow for a more stable pose estimation and marker tracking could mitigate the ambiguity problem.

8.2 Future Work

Platform Reevaluation

The choice to use the Bitcraze Crazyflie 2.1 platform for development has been beneficial to the progress of the project. It has enabled us to quickly build a working prototype as we intended, find related work by researchers using the same platform,

and easily find help from the community. However, we do believe the CF 2.1 is limited in a few ways. First, due to its size it cannot fly for an extended period of time as would be a requirement moving forward in the product development phase considering the goal application. Second, it is incapable of running real-time image analysis on-board. This limits its use to being within the range of the base station.

Full Autonomy

As mentioned in Section 1.4, a delimitation of the project is that the marker is within LOS at the instant our sequence controller is run. Thus, future work to achieve full autonomy includes mapping, localization and path planning. Fittingly, this is currently being investigated as previously mentioned.

Bibliography

Acuña, R. and V. Willert (2018). "Dynamic Markers: UAV Landing Proof of Concept". In: pp. 496–502. DOI: 10.1109/LARS/SBR/WRE.2018.00093.

Årzén, K.-E. (2014). *Real-Time Control Systems*. Department of Automatic Control, Lund University, Lund, Sweden.

Bitcraze (2020a). *240 mAh LiPo battery including 500 mA USB charger*. URL: https://store.bitcraze.io/products/240mah-lipo-battery-including-500ma-usb-charger (visited on 2020-04-28).

Bitcraze (2020b). *About*. URL: https://www.bitcraze.io/about/ (visited on 2020-04-28).

Bitcraze (2020c). *Controllers in the Crazyflie*. URL: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/controllers/ (visited on 2020-05-09).

Bitcraze (2020d). *Crazyflie 2.1*. URL: https://store.bitcraze.io/products/crazyflie-2-1 (visited on 2020-04-28).

Bitcraze (2020e). *Crazyradio PA*. URL: https://www.bitcraze.io/crazyradio-pa/ (visited on 2020-04-28).

Bitcraze (2020f). *Flow Deck v2*. URL: https://www.bitcraze.io/flow-deck-v2/ (visited on 2020-04-28).

Bitcraze (2020g). *Qi charger deck*. URL: https://www.bitcraze.io/products/old-products/qi-charger-deck/ (visited on 2020-04-28).

Bradski, G. (2000). "The OpenCV Library". *Dr. Dobb's Journal of Software Tools*. URL: https://opencv.org/ (visited on 2020-07-04).

Chadaporn, K., J. Baber, and M. Bakhtyar (2014). "Simple Example of Applying Extended Kalman Filter". In: *1st International Electrical Engineering Congress (iEECON2013)*. Chiangmai City, Thailand.

Chaumette, F., S. Hutchinson, and P. Corke (2016). *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer International Publishing. DOI: 10.1007/978-3-319-32552-1. URL: https://doi.org/10.1007%5C%2F978-3-319-32552-1.

Del Cont Bernard, D., F. Riccardi, M. Giurato, and M. Lovera (2017). "A dynamic analysis of ground effect for a quadrotor platform". In: vol. 50. DOI: 10.1016/j.ifacol.2017.08.1500.

Eachine (2020). *Eachine ROTG02*. URL: https://www.eachine.com/Eachine-ROTG02-UVC-OTG-5_8G-150CH-Dual-Antenna-Audio-FPV-Receiver-for-Android-Tablet-Smartphone-p-1063.html (visited on 2020-04-28).

Förster, J. (2015). *System Identification of the Crazyflie 2.0 Nano Quadrocopter*. eng. BA Thesis. ETH, Zürich, Switzerland. DOI: 10.3929/ethz-b-000214143.

Furgale, P., T. Barfoot, and G. Sibley (2012). "Continuous-time batch estimation using temporal basis functions". *The International Journal of Robotics Research* **34**, pp. 2088–2095. DOI: 10.1109/ICRA.2012.6225005.

Furgale, P., J. Maye, J. Rehder, and T. Schneider (2020). *kalibr*. URL: https://github.com/ethz-asl/kalibr (visited on 2020-07-04).

Furgale, P., J. Rehder, and R. Siegwart (2013). "Unified temporal and spatial calibration for multi-sensor systems". In: pp. 1280–1286. DOI: 10.1109/IROS.2013.6696514.

Gao, X.-S., X.-R. Hou, J. Tang, and H.-F. Cheng (2003). "Complete Solution Classification for the Perspective-Three-Point Problem". *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **25**, pp. 930–943. DOI: 10.1109/TPAMI.2003.1217599.

Garrido-Jurado, S., R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez (2014). "Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition* **47**:6, pp. 2280–2292. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. URL: http://www.sciencedirect.com/science/article/pii/S0031320314000235.

Goeller, L. (2018). *Drone Precision Landing using Computer Vision*. Semester project. Computer Engineering and Networks Laboratory, ETH, Zürich, Switzerland.

Gopalakrishnan, E. (2017). *Quadcoptor flight mechanics model and control algorithms*. MSc Thesis. Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic.

Greiff, M. (2017). *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*. eng. MSc Thesis. Department of Automatic Control, Lund University, Lund, Sweden.

He, X., G. Kou, M. Calaf, and K. K. Leang (2019). "In-Ground-Effect Modeling and Nonlinear-Disturbance Observer for Multirotor Unmanned Aerial Vehicle Control". *IFAC (International Federation of Automatic Control)* **141**:7, p. 071013. ISSN: 0022-0434. DOI: 10.1115/1.4043221. URL: https://doi.org/10.1115/1.4043221.

IKEA (2020). *Nordmärke*. URL: https://www.ikea.com/se/sv/p/nordmaerke-tradloes-laddare-vit-kork-00442574/ (visited on 2020-05-20).

Kannala, J. and S. S. Brandt (2006). "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, pp. 1335–1340.

Karlsson, C. (2019). *Vision based control and landing of Micro aerial vehicles*. BA Thesis. Department of Engineering and Physics, Karlstads Universitet, Karlstad, Sweden. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-73225 (visited on 2020-07-04).

Klaffenbach, D. (2010). *Simple state machine implementation (Python recipe)*. URL: http://code.activestate.com/recipes/577308-simple-state-machine-implementation/ (visited on 2020-05-05).

Kumar, V. (2018). *Rigid Body Transformations*. URL: https://www.seas.upenn.edu/~meam620/slides/kinematics0.pdf (visited on 2020-04-23).

LaValle, S. M. (2006). *Planning Algorithms*. URL: http://msl.cs.uiuc.edu/~lavalle/planning/ (visited on 2020-03-05).

LaValle, S. M. (2019). *Virtual Reality*. URL: http://vr.cs.uiuc.edu/web.html (visited on 2020-04-29).

Lu, X. (2018). "A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation". *Journal of Physics: Conference Series* **1087**, p. 052009. DOI: 10.1088/1742-6596/1087/5/052009.

Ma, L., Y. Chen, and K. Moore (2003). "Rational Radial Distortion Models with Analytical Undistortion Formulae". *ArXiv* **cv.CV/0307047**, pp. 135–147.

Macharia, J. (2017). *Wireless Inductive Charging for Low Power Devices*. eng. BA Thesis. Helsinki Metropolia University of Applied Sciences, Helsinki, Finland.

Mason, M. (2006). *Representing Rotation, Mechanics of Manipulation*. URL: http://www.cs.cmu.edu/~mason (visited on 2020-04-23).

McGuire, K. (2020). *State estimation: To be or not to be!* URL: https://www.bitcraze.io/2020/01/state-estimation-to-be-or-not-to-be/ (visited on 2020-04-14).

Mearian, L. (2018). *Qi - Mobile Computing*. URL: https://www.computerworld.com/article/3235176/wireless-charging-explained-what-is-it-and-how-does-it-work.html (visited on 2020-05-23).

Muñoz-Salinas, R., M. Marín-Jiménez, and R. Medina-Carnicer (2018). "SPM-SLAM: Simultaneous Localization and Mapping with Squared Planar Markers". *Pattern Recognition* **86**, pp. 156–171. DOI: 10.1016/j.patcog.2018.09.003.

Muñoz-Salinas, R. and R. Medina-Carnicer (2020). "UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers". *Pattern Recognition* **101**, p. 107193. ISSN: 0031-3203.

OpenCV (2020a). *About*. URL: https://opencv.org/about/ (visited on 2020-02-04).

OpenCV (2020b). *Detection of ArUco Markers*. URL: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html (visited on 2020-02-04).

Richardsson, K. (2016). *Position control moved into the firmware*. URL: https://www.bitcraze.io/2016/05/position-control-moved-into-the-firmware/ (visited on 2020-04-14).

Richardsson, K. (2018). *Preparations for iros 2018*. URL: https://www.bitcraze.io/2018/09/preparations-for-iros-2018/ (visited on 2020-05-08).

Rodrigues, O. (1840). "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire." fre. *Journal de Mathématiques Pures et Appliquées* **5**, pp. 380–440. URL: http://eudml.org/doc/234443.

RunCam (2016). *RunCam Nano 3*. URL: https://shop.runcam.com/runcam-nano-3/ (visited on 2020-04-28).

RunCam (2019). *RunCam TX25 vTx*. URL: https://www.getfpv.com/runcam-tx25-vtx.html (visited on 2020-04-28).

RunCam (2020). *RunCam TX25*. URL: https://www.runcam.com/download/RunCam-TX25-User-Manual.pdf (visited on 2020-04-28).

Scaramuzza, D., A. Martinelli, and R. Siegwart (2006). "A Toolbox for Easily Calibrating Omnidirectional Cameras". In: IEEE, Beijing, China, pp. 5695–5701. DOI: 10.1109/IROS.2006.282372.

Semple, S. (2020). *Black 3.0 - The world's blackest black acrylic paint*. URL: https://culturehustle.com/products/black-3-0-the-worlds-blackest-black-acrylic-paint-150ml (visited on 2020-05-12).

STMicroelectronics (2018). *VL53L1X*. Rev 3. STMicroelectronics. URL: https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html (visited on 2020-05-06).

TDK (2017). *Background: Various wireless power transfer technologies*. URL: https://product.tdk.com/info/en/techlibrary/developing/wireless/index.html (visited on 2020-05-06).

The MathWorks Inc. (2020). *Camera Calibrator*. URL: https://www.mathworks.com/help/vision/ref/cameracalibrator-app.html (visited on 2020-05-12).

Voler Systems (2020). *Wireless Charging*. URL: https://www.volersystems.com/v2019/wireless-charging/ (visited on 2020-05-08).

Wei, P., S. Chan, S. Lee, and Z. Kong (2019). "Mitigating ground effect on mini quadcopters with model reference adaptive control". *International Journal of Intelligent Robotics and Applications* **3**, pp. 283–297. DOI: 10.1007/s41315-019-00098-z.

Wireless Power Consortium (2018a). *Magnetic Resonance and Magnetic Induction*. URL: https://www.wirelesspowerconsortium.com/knowledge-base/magnetic-induction-technology/resonance/magnetic-resonance-and-magnetic-induction-making-the-right-choice-for-your-application.html (visited on 2020-05-07).

Wireless Power Consortium (2018b). *Qi - Mobile Computing*. URL: https://www.wirelesspowerconsortium.com/qi/ (visited on 2020-05-23).

Wu, P.-C., J.-H. Lai, and S.-Y. Chien (2012). "Stable Pose Estimation with a Motion Model in Real-Time Application". In: IEEE, pp. 314–319. ISBN: 978-1-4673-1659-0. DOI: 10.1109/ICME.2012.176.

Zhang, Z. (2000). "A Flexible New Technique for Camera Calibration". *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**, pp. 1330–1334. DOI: 10.1109/34.888718.

A

Appendix: Recommendations to Verisure

Based on the conclusions drawn from this project we have some recommendations and thoughts about how Verisure should move on with quadcopters as a part of their alarm solutions.

A.1 Further Improvements

Below follows some of the limitations of the current design and suggestions on how to improve it.

Alternative UAV Platform

Along with the progression and development of the project we have noted that the Bitcraze Crazyflie 2.1 has limitations. These limitations are mainly due to its small size. While being a great and safe development platform, it lacks some desirable functionality. First of all, it has a limited flight time due to its small battery. This becomes a problem when considering that the final product must be able to navigate an entire house without losing power. Second, it has limited processing power and would not be able to process images. Third, it is unable to simultaneously support accurate velocity/position control (enabled by the "Flow deck V2") and wireless charging (enabled by the "Qi 1.2 charger deck") in its current configuration. Fourth, it is likely more sensors are needed in the final product.

With the above in mind, we suggest that an alternative UAV platform should be investigated.

Requirements In order to use the current implementation as-is, the requirements on an alternative platform are:

- Downward facing camera with a minimum resolution of 640x480 pixels and wide field of view (FOV) $\approx 160°$.
- Stable flight controller including velocity controller in x,y,z and yaw axis. This requires:
  - measurement/accurate estimation of roll and pitch.
  - measurement/accurate estimation of velocities in x,y,z, and yaw.
- Image processing unit able to process video feed at ≥ 30 FPS.
- Python code compatibility, although all code can be rewritten using corresponding C/C++ libraries.

Considerations If further development to the current implementation were to happen, we recommend considering the following:

- Using ROS and the Crazyflie ROS driver instead of the Python library. ROS is modular and additional functions are added more easily.

Recommendations To further improve the platform and build something better than the Crazyflie 2.1, we highly recommend considering the following:

- Bigger battery. This requires:
  - more powerful motors, larger propellers, and bigger frame.
- More and better camera(s).
- On-board image processing capabilities.
- Charging solution compatible with base station, e.g., wireless charging.
- Protect rotor blades to avoid damage to them, people or surroundings.

New Base Station Design

Since the current base station design diminishes the landing accuracy a new design should be investigated. As explained in the report, the designated groove designed for the drone to slide into position is probably the reason why the landing was worse on the station. On a flat surface the landing was a lot better so a good approach would probably be making a flat design. It should be easy to integrate a wireless charger in a flat design.

Requirements In order to use this projects implementation as-is, the requirements on a base station are:

- One large and one small ArUco marker with side lengths ≈ 11 cm and ≈ 2 cm, respectively.

Considerations If further development to the current design were to happen, we recommend considering the following:

- Avoid reflections by using the paint "Black 3.0" or a bright display.

Recommendations To further improve the base station, we highly recommend considering the following:

- Charger compatible with UAV platform, e.g., wireless charging.
- Investigating how/why the platform design affects landing performance.
- Alternative to fiducial markers/ArUco library. Base station tracking/pose estimation may be possible with other methods which do not require larger 2D markers. This may be desirable from an aesthetic perspective.

A.2 Future Work Suggestions

To proceed with the drone project here are some suggestions on upcoming thesis projects and probable student profiles.

- Design of a new base station - Technical and/or industrial design.
- Design of a Verisure drone. From ground up or by finding a suitable drone manufacturer to collaborate with - Electrical and/or mechanical engineering.
- Communication to alarm central - Computer science and/or information technology
- Further work on the thesis of Sofie Olsson (localization and mapping) - Engineering physics/mathematics and/or computer science
- Navigation and path planning - Engineering physics/mathematics and/or computer science
- Collision avoidance - Engineering physics/mathematics and/or computer science
- Alarm verification/burglar identification (computer vision) - Engineering physics/mathematics and/or computer science