Integration of a Mini Camera into a Tiny Quadrotor for Navigation with Visual Markers

Martin Didion, Roman Eisenhauer, Dennis Kraus, Garwin Lechner

***Abstract*— Nowadays, drones become progressively more important and intensive research is undertaken in this field of study. For this the nano quadrotor Crazyflie 2.0 from Bitcraze [1] is a perfect research tool, since the code is open source based on a framework ,the so called, *Robot Operating System* (*ROS*).**

In this paper we propose a design for a camera based control of a quadcopter for an autonomous flight with fiducial markers called *ArUco*. Instead of a ceiling mounted camera, as in the previous work [2], a tiny camera is directly mounted on the Crazyflie. This represents a portable solution to locate a drone in space, which is necessary for controlling. For this purpose the code from Bitcraze, including three PID-Controller for x-, y- and z-direction, is extended.

Finally, two experiments are accomplished to evaluate the performance of the implementation. The tests address hovering above a marker and waypoint navigation between several markers.

I. Introduction

QUADCOPTERS become more and more popular in research especially in the area of autonomous flight. They are a promising solution for many applications like supervising an area or aerial controlling of ground robots. To assign a drone to such tasks its pose, consisting of its position and orientation, in space must be determined. Therefore, the use of a suitable sensor is needed. Since every professional quadcopter has a gyroscope and an accelerometer available, they seem to be an appropriate solution. However, they are not suitable for a reliable determination of the pose due to lacking accuracy and drifting. So the use of a visual system like a camera in combination with fiducial markers for calculating the pose proves to be a promising solution.

An interesting drone for research is the Bitcraze nano quadcopter Crazyflie 2.0 [1] that has the dimensions of $9\,\mathrm{cm}^2$ and an extremely low weight of $27\,\mathrm{g}$. Its big advantage is that the software of the quadcopter is based on an open source framework, the so called *Robot Operating System* (*ROS*). *ROS* offers the possibility of a simple extension of the existing code and provides a communication between the drone and a computer. Thereby the drone can easily be upgraded by further modules to achieve a desired functionality.

The purpose of this work is to attach a camera to the Crazyflie and extend the code from Bitcraze to realise an autonomous flight. Calculating the pose using the image of a camera is solved by special markers, called *ArUco* markers. Furthermore, the extreme small size and weight of the Crazyflie is a challenge for the implementation, because the additional weight to attach a camera is highly restricted.

This work has been supported by M.Sc. Raúl Acuña Godoy.

The following section gives an overview about related work in the field of autonomous flight with the Crazyflie. In section III we describe the requirements and practical implementation of the camera casing and its attachment to the drone. In section IV we first describe fundamentals such as the *ArUco* markers and image processing. Then we explain different aspects of the autonomous flight such as controlling and waypoint navigation. In section V we point out the major results of this work and the last section completes the paper giving a conclusion and a perspective for future work.

II. Related Work

Different approaches have been done to achieve autonomous flight with a quadrotor. Especially the use of fiducial markers for calculating the pose of a drone is popular as in [3], [4] and [5]. Olivares-Mendez et. al [5] combined *Virtual Robotics Experimental Platform* (*V-REP*) with *ROS* to create a controller for a quadrotor. Before using a real drone, all steps are simulated for setting up a controller using their implemented testbed. The final controller consists of three parallel fuzzy controllers that use the pose information gained with *ArUco* markers. Good results are achieved to responding on a step and interference, while hovering is moderately accurate and stable. Their results are oscillating in a sector of $\pm 20\,\mathrm{cm}$.

There exist scientific papers that extend the Crazyflie for realising an autonomous flight. Beguer [2] equips the quadrotor with fiducial markers and uses an external camera for determining its position. The controller is based on the Crazyflie-package for *ROS* with the extension of a feed-forward PID-Controller for the height. This controller accomplishes an accurate and stable flight with some problems in maintaining stationary accuracy.

A further approach which is similar to our task is presented in the thesis of Dunkley [6]. The author attaches a camera directly to the Crazyflie and uses *ROS* for the communication between a computer and the drone. Two systems are tested for calculating the pose: The first one is called *Visual Inertial SLAM* and is working with the camera of the drone. It tracks key-points like edges in a picture for determining the motion. The second one is an external system called *Kinect based pose estimator* that calculates the pose by depth information. The controller consists of four parallel PID-Controller for x, y, z and yaw and is also based on the Crazyflie package. For a better analysis of the results concerning the pose, a professional motion capture system is used as ground truth. It uses twelve ceiling mounted cameras and infra-red markers. Good results are achieved with the Kinect system with an error of about $10\,\mathrm{cm}$ for hovering and a fast response to a step of

about two seconds. The results for the *Visual Inertial SLAM* shows some degradation but is still sufficient for a system without markers. So the error during hovering raises to $\pm$ 20 cm.

The system proposed in this paper is based on the approaches in [2] and [6] since they have similar conditions and requirements regarding the task.

III. Hardware

To mount a camera on the quadrotor the development of a casing that attaches the camera to the drone is essential. The requirements for the casing are determined by the characteristics of a nano drone.

Regarding the small weight of 27 g including the battery the payload is for different reasons extremely restricted. The nominal value of the maximum payload amounts to 15 g.

First, the maximum flight time without camera is only about 7 min and increasing the overall weight of the quadcopter will further reduce this time. Second, the thrust needed to capture a downward movement into a level position or even an upward movement increases with the weight, which especially gets more critical the lower the battery status is. In the end, this effects the total possible flight time as well, because as soon as the remaining available thrust of the rotors is not sufficient to maintain a proper flight, the quadcopter will result in decreasing the height.

Furthermore, it needs to be taken into account that the camera uses the same battery as the Crazyflie, since adding another battery is not feasible. This again affects the maximum flight time.

Since the base area of the quadcopter constitutes only 9 cm^2 and because of the compact design, there are just few possible points of attachment for the camera casing. In Fig. 1 the bottom of the Crazyflie without camera and casing is depicted.

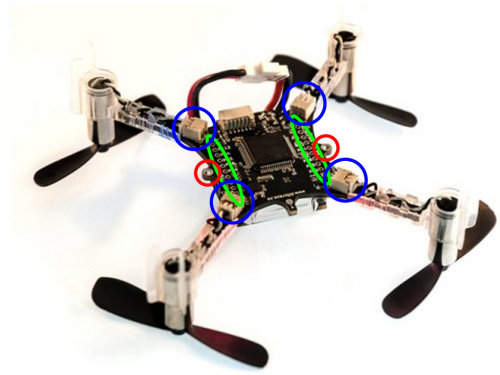


Figure 1. Bottom view of the Crazyflie 2.0 without camera and casing. Three possible attachments for the camera are shown: rotor arm connectors (blue), metallic holes (red) and coupler stripes (green) [7].

One approach is to use the four white connectors between circuit board and rotor arms which brings up the problem of the concrete attachment. A second approach utilises the two metallic holes protruding to the left and right of the circuit board allowing an easy attachment of the casing. Unfortunately, there are just two metallic holes that lead to a possible momentum around the y-axis. A third approach uses the longitudinal arranged coupler strips on both sides of the circuit board. There are each ten allowing variable length between two longitudinal points of attachment. The distance in lateral direction is fix and adds up to 16.7 mm. In regard to the dimensions of the base area of the camera the outermost pinholes are preferable to enclose the camera.

The camera needs to meet three requirements: The weight and the dimensions are crucial due to the restricted maximum payload and the small size of the quadcopter. It is not feasible to attach a camera whose dimension exceeds the one of the drone by far.

On the one hand the camera should be as small and lightweight as possible but on the other hand it requires a high resolution, since the higher the resolution of the camera the better the results of the process of the marker identification.

Moreover, the camera needs to be available for a reasonable price matching the other parts of the quadcopter and fitting in the concept of an affordable research platform in the field of nano drones.

A camera satisfying these requirements is the model FX797T from FXT Technology. It has a maximum resolution of 720x480 with an overall weight of 4.5 g and a prize of around 40 Euro. The viewing angle of 120°, in conjunction with 30 fps, guarantee a reliable marker detection. The camera with the dimensions of 20 mm x 17 mm can be seen in Fig. 2.

It is powered by the same battery (charge: 240 mAh, weight: 7 g) as the Crazyflie.

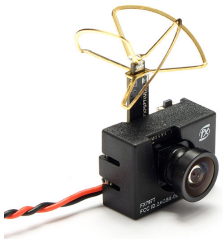


Figure 2. The camera model FX797T from FXT Technology is used to extend the Crazyflie [8].

A. Test Casing

The process to elaborate a professional solution from an idea takes time which implies that in the meantime no flight of the quadcopter with mounted camera would be possible. Hence, neither the impact of distortions during a flight such as the influence of the motors on the power supply for the camera nor the marker detection could be analysed. Therefore, we constructed a rapid solution to allow an immediate take off of the Crazyflie equipped with the camera. This, the so called, *test casing* is also used as a start point to elaborate more sophisticated solutions.

The *test casing* consists of a Printed Circuit Board (PCB) with a round opening for the lens of the camera. Two adhesive tapes hold the camera in position. Four pins of a multi-pin connector are used to attach the casing to the quadcopter utilising its pinholes on the circuit board. The pins are put through the PCB as well as through the pinholes on the drone and end caps on both sides prevent them from slipping out. The Crazyflie with the camera using the *test casing* can be seen in Fig. 3. This solution is crude and not lightweight because of the massive PCB, but it fulfils its purpose in a satisfying way.

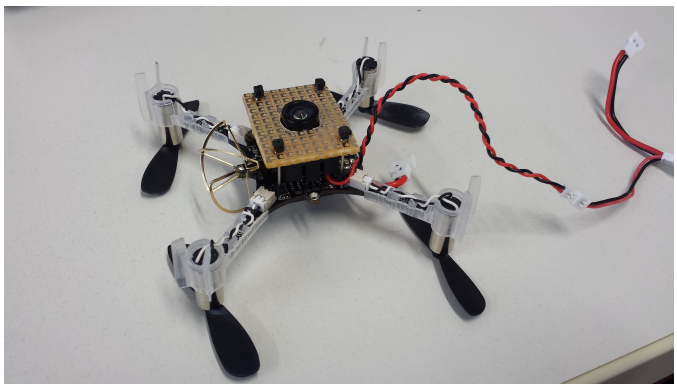


Figure 3. Bottom view of the Crazyflie with camera and *test casing*.

B. 3D Printing

To ensure a conventional use of the quadrotor including the camera, a professional casing is needed. This casing should be easy to produce and as well as easy to be replicated. Following the current temporary solution a CAD model is being designed (see Fig. 5). The final production is realised by 3D printing that can be categorised in two different manufacturing methods in general.

The first one is called *Fused Deposition Modelling* (*FDM*). Melted Plastic is applied by a tip on a heated printing bed. Due to the fusion of multiple layers, a three dimensional object is being built.

The alternative is the *Selective Laser Melting (SLM)* method. Here, a thin layer of powder is punctually being melted and liquefied by a precise laser.

In this case, the *FDM* method is recommended due to the fact that it is quicker and cheaper, but nonetheless fulfils all requirements for the ideal casing. Due to the fact that there is no 3D printer provided by the subject area, the 3D printing is executed in a manufacturing laboratory. Following printers are provided:

- MakerBot Replicator 2
- Ultimaker 2
- Prusa i3

Those printers differ in price as well as in technology regarding the extrusion of the filling material. Both, Makerbot and Prusa i3, use a Direct Extruder, which is located at the hot end and also pulls the filling towards it.

Instead the Ultimaker 2 uses a Bowder Extruder on its shell, which pushes the material all the way to the hot end. Due to the fact that the results from the Ultimaker 2 were not sufficient, the casing is being printed with the Prusa i3.

Polylactic acid (PLA) will be the filling material and is going to be used in a thickness of $3\,\mathrm{mm}$. The high stability in addition with the small density is ideal for our casings' requirements. With a height of $0.2\,\mathrm{mm}$ per layer a fast print and production is given. The overall amount of costs for the material per casing is less than 5 cents.

C. Fabrication

The original idea of the *test casing* is going to be the base of the final one. The casing consists of a plate with an elliptic cut-out for the lens. With the shape of an ellipse the best trade-off between weight reduction and structural strength is gained. The lens of the camera is placed through the cut-out, facing towards the ground. As the connection between that base plate and the quadcopter, four $0.7\,\mathrm{mm}$ wide, square hollow feet are being used. Small metal pins are simply being stuck in these feet and put through the pin-holes on the circuit board of the drone. Summarised the casing needs to meet the following requirements:

- needs to be as lightweight as possible
- central positioning of the camera lens onto the drone
- quick and easy exchange of the casing in case of damage
- no reduction of power or harm on the rotor drive
- proper and secure fixation of the camera

Four wall-like plates secure and hold the camera in place with only a small amount of pressure being used. Thus, the camera is fixed in x- and y-direction. Movement within coordinate z on the other hand is prevented by the plate as well as by the quadcopter itself. To prevent electrical contact, a non-conductive styrofoam sheet is placed between the drone and the camera.

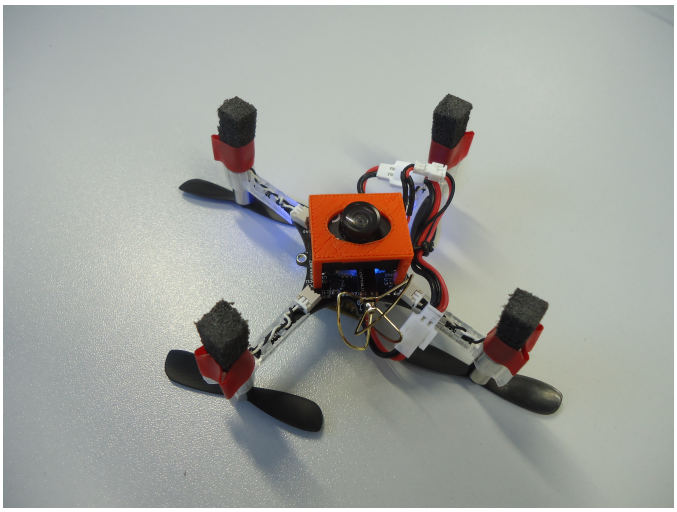


Figure 4. Bottom view of the Crazyflie with the final casing.

The distance between the metal pins exceeds the distance between the pin-holes. Thus, by pushing the pins through the

holes a tension is generated, which prevents movement in coordinate z.

Nonetheless, it is necessary to attach end caps to the end of the pins coming out of the pin-holes on the top side of the drone to prevent the casing from slipping out. The proposed attachment provides an effortless exchange of the casing.

In addition to the casing, so called, bumpers are added, to prevent damage on the lens after a rough landing or even a crash.

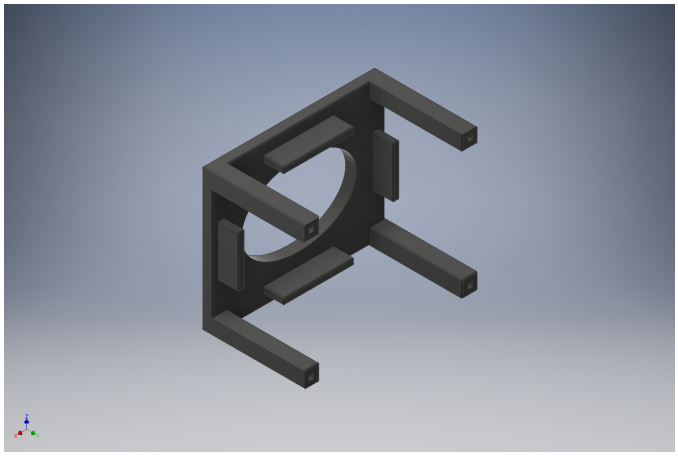


Figure 5. CAD model of the final casing.

In Conclusion, all requirements are sufficiently fulfilled. Especially the casings' weight of only $1.3\,\mathrm{g}$ confirms the choice of the production method.

IV. Software

Most important for achieving an autonomous flight is a variety of accurate sensors. The Crazyflie 2.0 comes with a 9-axis motion tracking device MPU-9250 which includes a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer. Besides, the high precision pressure sensor LPS25H is installed. Unfortunately, those are not very reliable, because on the one hand noise and air condition changes and on the other hand drifts while quantizing the analogue sensor signals are occurring. The former especially affects the barometer and slightly the magnetometer, because of objects and motion in the room. The latter raises problems with the use of the gyroscope and accelerometer for measuring the travelled distance. So an attempt with an additional on-board camera (see Sec. III) as sensor replacement is made to get rid of these negative impacts. In the following, some fundamentals and our set-up are described before our approach is explained.

A. Fundamentals

1) ArUco markers: For a better determination of the drone position in space, we use some special markers called *ArUco* markers. The advantage of these markers are that they yield much information because their appearance and shape is known. An *ArUco* marker consists of a black square with several smaller white squares in it that build a specific shape. In Fig. 6 a complete marker board is shown that consists of four markers. These boards are more reliable compared to one marker.

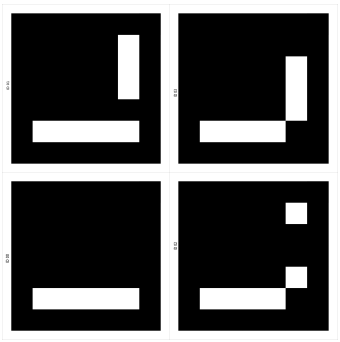


Figure 6. A marker board that consist of four *ArUco* markers.

It is possible to use a different number of markers to build up a board. To get the best ones, they are newly created for every desired number and are saved in a dictionary. The criteria for choosing a marker is the inner distance of a marker itself and the inter distance between a new marker and the already saved ones. In this case the inner distance is determined by the number of transitions between black and white squares during rotation and the inter distance describes the differences between the shape of the compared markers [9].

The detection of the marker in a scene is realised by adaptive thresholding, contour extraction and polygonal approximation. After receiving the relevant contours, an analysis of the inner region is performed. This is done by removing the perspective projection and using a rectangular grid to determine the binary code of the marker. For a better understanding, the process of a marker detection and the calculation of the binary code is shown in Fig. 7. The marker identification and error correction of the binary code is carried out by comparing its code to the dictionary. The final step consists of a corner refinement and the calculation of the pose of the marker in respect to the camera [9].

2) Camera Calibration: The process of estimating the intrinsic and extrinsic parameters of a pinhole camera is called camera calibration [10]. The intrinsic ones are describing parameters of a lens and image sensor like the focal length $\boldsymbol{f} = (f_x, f_y)$ and the principle point $\boldsymbol{c} = (c_x, c_y)$. Finally, they are recorded in a 3×3 camera matrix

$$\boldsymbol{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

With their help it is possible to measure the size of objects and their distance from the camera in a captured scene. The extrinsic ones are describing the transformation of the camera and the scene and are further explained in the following section.

While the pinhole model has no lens, real cameras do. Because of its curvature, captured world points appear distorted on the real image. This is called radial distortion as originally

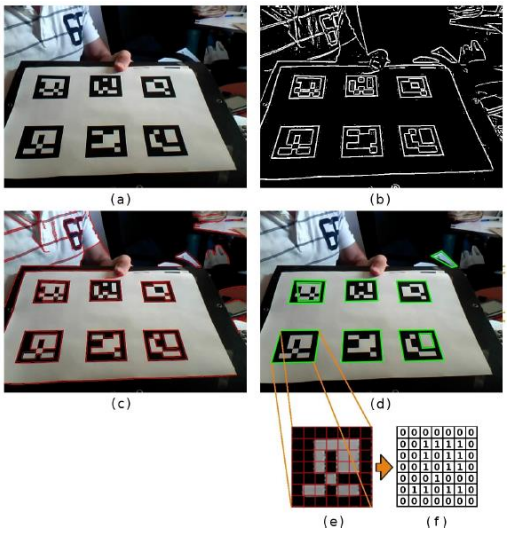


Figure 7. Complete process from thresholding to marker identification: (a) Original Image. (b) Adaptive thresholding. (c) Contour detection (d) Polygonal approximation. (e), (f) Binary code by rectangular grid [9].

straight lines are bended in the image as it can be seen in Fig. 8. Another type of distortion is the tangential one which happens due to not perfectly aligned lenses during production. This is seen by objects in the image appearing closer or further away than they actually are. A method to correct these errors is presented in [11] by capturing a well suited pattern for example a chessboard to know the world points and their projected distorted image points. Subsequently an polynomial equation containing the, so called, distortion parameters for each type of distortion has to be found. With this equation every distorted point can be remapped to its correct position.

Because there is plenty of literature regarding this topic the interested reader is encouraged to look up further information and methods in [11] and [12].

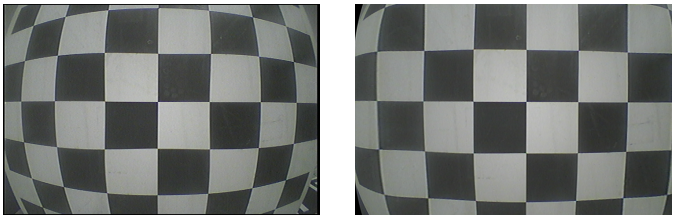


Figure 8. Radial distorted image of a chessboard (left) and the corresponding undistorted one (right)

3) Rigid Body Transformation: The orientation of a camera can be described by a, so called, camera coordinate system where the x-axis is pointing to the right, the y-axis along bottom and the z-axis in view direction. Now an arbitrary world coordinate system is chosen which serves as a reference frame. By comparing both coordinate systems it is possible to compute the translation along the three axes and the rotation around each axis as it can be seen in Fig. 9. Hence, a translation vector

$$\boldsymbol{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{2}$$

can be created. The resulting rotation matrix

$$\boldsymbol{R} = \boldsymbol{R}_x \cdot \boldsymbol{R}_y \cdot \boldsymbol{R}_z \tag{3}$$

follows from a multiplication of each individual rotation $\boldsymbol{R}_x$, $\boldsymbol{R}_y$ and $\boldsymbol{R}_z$ around an axis with the general form of a rotation matrix. For example

$$\boldsymbol{R}_z = \begin{pmatrix} cos\delta & -sin\delta & 0 \\ sin\delta & cos\delta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

is the corresponding matrix for a rotation around the z-axis with the rotation angle δ.

With this information it is possible to express a point $\boldsymbol{x}$ from the camera frame in the world frame and vice versa using

$$\boldsymbol{x}_w = \boldsymbol{R}(t) \cdot \boldsymbol{x}_c + \boldsymbol{t}(t). \tag{5}$$

Commonly a camera captures a moving scene which is why both quantities need to be time-dependent.

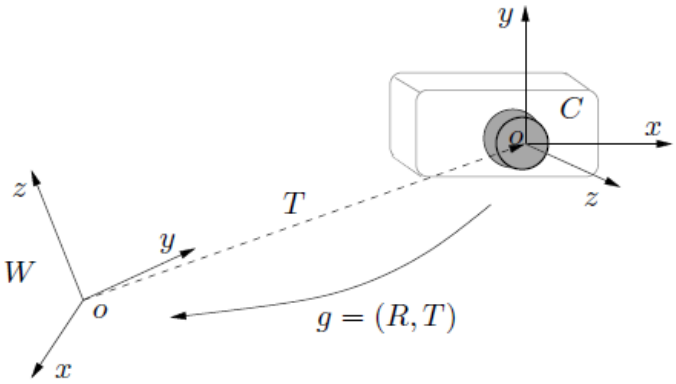


Figure 9. Rigid body motion (from [12])

4) Motion of a quadrotor: In contrast to a helicopter, a quadrotor is distinguished by a simple mechanical structure. There is no need of a mechanical actuation of the rotor slope because of the fixed four rotors. To enable a stable flight one pair of the motors rotate clockwise while the other pair rotate counter-clockwise.

This also allows the quadcopter to apply motion in the six degrees of freedom: Translation in x, y and z direction and the three rotations called yaw, pitch and roll. The change of the rotational speed of each motor to realise one of these movements are illustrated in Fig. 10.

Taking these flight characteristics into account, a motion model for a quadcopter can be described as follows [6]:

$$Motor_1 = thrust + u_{\dot{r}} - u_{\dot{p}} + u_{\dot{y}} \tag{6}$$
$$Motor_2 = thrust + u_{\dot{r}} + u_{\dot{p}} - u_{\dot{y}} \tag{7}$$
$$Motor_3 = thrust - u_{\dot{r}} + u_{\dot{p}} + u_{\dot{y}} \tag{8}$$
$$Motor_4 = thrust - u_{\dot{r}} - u_{\dot{p}} - u_{\dot{y}}. \tag{9}$$

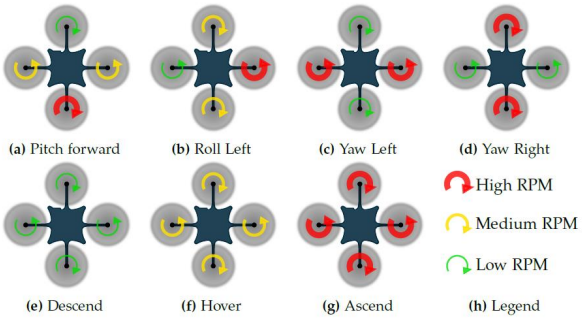


Figure 10. The figure shows the rotation direction of the rotors and the resulting movement by changing the ratio of the rotational speed between the them (from [6])

The formula presents the distribution of the thrust to fulfil a desired motion. The parameters $u_{\dot{r},\dot{p},\dot{y}}$ are the actuating variables for the corresponding rotation and $thrust$ describes the needed thrust to hover at a specific height.

B. Autonomous Flight

After clarifying the basics, the crucial part of the software will be explained in the following. For realising an autonomous flight the set-up of a controller and the management of the flight-states as well as the waypoint-navigation is necessary. Before describing these topics, the structure of the whole program in ROS is described.

1) ROS Dependencies: Our program is integrated in the *ROS* environment wherefore a brief introduction is necessary. The *Robot Operating System* (*ROS*) is an open source framework for software developing and can be seen as a meta operation system. Its main purpose is the reuse of code for research and development projects. Thus, a strategy for interfaces need to exist. The individual programs are called nodes and are delivered in packages. Each node can publish and subscribe to a topic which is like a channel on that messages are sent. In addition, every node may have services implemented which are public functions that can be called from every other node.

In this work some additional packages are used. The connection between every node and their topics is illustrated in Fig. 11 and the process is explained in the following. First, a node for establishing a camera stream is needed, so we choose *usb_cam* because it is easy to use and still maintained. However, there are no functional advantages over equivalent packages like *uvc_camera* recognized. Next, the resulting camera image stream needs to be preprocessed by *tud_img_prep* to remove interlacing, which occurs during the shaking of the camera. Performing a histogram equalisation is often useful for strengthening details, but in this case it compounds light reflection on the marker which leads to a worse recognition. Now the preprocessed image needs to be rectified for further computations where the native ROS package *image_proc* comes in handy. This node performs this task as explained in Sec. IV-A.2. Afterwards, the markers in the image need to be detected and their pose has to be computed. This is done by *ar_sys*, which publishes their transformations. Our newly written package *ar_nav* subscribes to this and modifies the transformation axes to match the crazyflies' coordinate system. Additionally, it registers the new frame relation so it will be available for other nodes. Next, the official crazyflie package *crazyflie-ros* is used for communicating with the crazyflie and controlling it with our target value coming from *ar_nav*. To meet our requirements some modifications are made, which are explained in their corresponding sections.

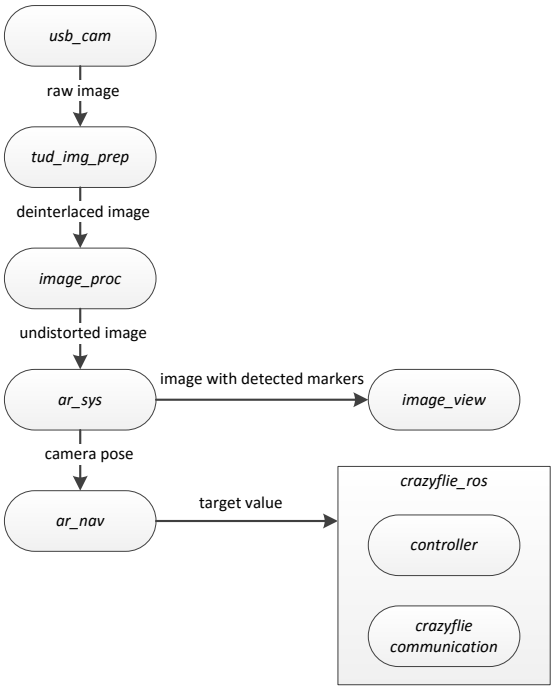


Figure 11. Connections of every node (ellipse) and their topics (arrows).

2) PID-Controller: The problem of setting up a controller for an autonomous flight of the Crazyflie 2.0 is the absence of a drone model. There are no information about the relationship between input signal and output behaviour so the number of applicable controllers reduces to some fundamental ones that do not need a mathematical model. The most common one for such situations is the PID-Controller. This controller provides a linear relationship between its input and output signal and is composed of a proportional, integral and derivative part. Each part has its own functionality:

- **The proportional term** provides the base for the controller.
- **The integral term** guarantees stationary accuracy.
- **The derivative term** reduces overshoot.

The equation for the complete controller can be described

as follows, whereas the integral and derivative part is approximated by a discrete numeric calculation:

$$u(t) = \underbrace{K_p e(t)}_{P} + \underbrace{K_i \sum_{\tau=0}^{t} e(\tau)}_{I} + \underbrace{K_d \frac{e(t) - e(t-\delta_t)}{\delta_t}}_{D} \quad (10)$$

The error $e(t)$ describes the input, $u(t)$ the output and δ_t the time difference between two calculations of the controller. The gains $K_{p,i,d}$ of the proportional (P), integral (I) and derivative part (D) can be determined by experiments so the controller is not dependent of a model. However, its disadvantage is the requirement of a linear system.

For our application we use two PID-Controllers, one in the inner loop for the attitude and rate control of the drone and the other one in the outer loop for controlling the position. The whole controller is shown in Fig. 12.

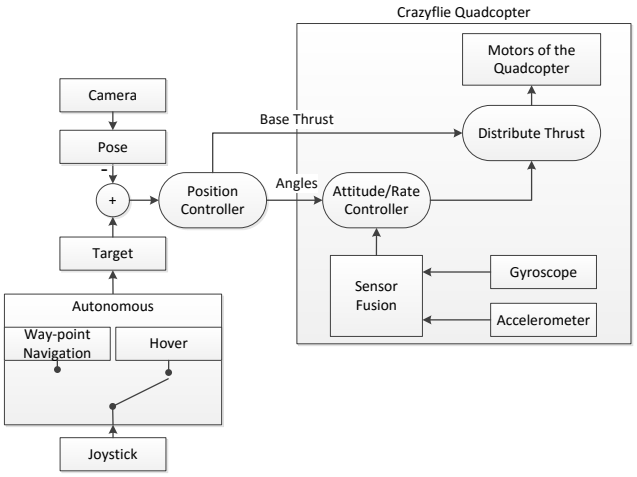


Figure 12. The complete controller for an autonomous flight: The inner loop is shown together with the distribution of the thrust and the fusion of gyroscope and accelerometer. The outer loop is extended by a switch between hover and waypoint navigation that provides the target for controlling.

The attitude control is executed on board of the drone and is implemented by the team of Bitcraze [13]. Its task is to reach the desired roll, pitch and yaw angles. The position controller is based on the Crazyflie-package *crazyflie-ros*. It consists of four linear PID-Controllers for each of the four parameters x, y, z and yaw. Since this configuration proves to be insufficient for our conditions, some additional changes need to be done. A linear height controller is inadequate because a flying drone is a non-linear system. So a base thrust is added to control around a working point that is called *feed-forwarding*. Furthermore, the integral part for x- and y-direction causes an unstable behaviour during large differences between target and actual position. The drone strongly accelerates and overshoots so much that it looses the marker. Therefore, the integral part for x and y is only used in direct surrounding of the target (within $20\,\mathrm{cm}$). Finally, this configuration of the controller is a proper solution for our requirements. As shown in Fig. 12 the last step of the overall control loop is the distribution of the thrust to the motors.

3) Flight States: To additionally realise an autonomous start and landing of the drone a management of some flight states is necessary. These flight states are based on the Crazyflie-package *crazyflie-ros*. The state *Safety-Landing* has been added and *Landing* as well as *Automatic-Mode* has been changed completely to fit our conditions. In the following the five flight states are explained:

- **Take-Off:** The drone starts with a base thrust that is raised step by step until a marker is detected for the first time.
- **Automatic-Mode:** As soon as a marker is detected the drone switches in *Automatic-Mode*. It uses the information of the position gained by the markers to control its pose. The mode consist of two different versions. One is for hovering on a specific hight directly above a marker while the other one is for performing a waypoint navigation explained in section IV-B.4.
- **Landing:** This mode can be started by a manual command during *Automatic-Mode*. It sets the target height zero and jumps in *Automatic-Mode* to reach this position. The problem is that a marker is lost at a low height, because the camera is too close to a marker to completely capture it. At this moment the thrust is reduced step by step for a specific time without regarding a marker. The time is sufficient to reach the ground and the drone switches in the below explained mode *Idle*.
- **Idle:** This mode ensures that the drone is doing nothing. The Crazyflie remains in a sleeping mode and waits for the next command.
- **Safety-Landing:** This state is for special situations and should not be activated in a successful flight. It is started automatically when no marker is found for a duration of one second. The drone starts to land by quickly reducing the thrust for a specific time that is sufficient to reach the ground. The mode has the function to protect the drone from crashing an obstacle in the room after loosing track of a marker. The landing is done without the help of a marker and is finished by switching to *Idle*.

A gaming controller can be used to execute the manual commands for starting a flight or switching to landing. So a complete automatic flight can be performed with pressing only one button.

4) Waypoint Navigation: In addition to hovering over a certain marker a waypoint navigation is possible where the drone flies a given course of markers. This is shown schematically in Fig. 13. The route can be configured on start up of *ar_nav* as a parameter which contains the *ArUco* marker board names in the desired order. After taking off the drone listens for the first waypoint pose transformation and approaches it as it would happen in single board use. The current waypoint can be changed either automatically or manually. If the drone

stays for a given time within a certain range around its target position, the current marker counts as successfully approached and the next waypoint is targeted. Alternatively, the next or previous waypoint can be selected by pressing a certain button on the gamepad. This procedure goes on in a cycle meaning after the last waypoint the first one is approached again.

When switching the waypoint the deviation between target value and setpoint is large, therefore the manipulated variable is high as well. This leads to a massive overshoot when the target value is reached because in the air the quadcopter has almost no resistance. To prevent this a stepping target value $\boldsymbol{w}_s = (x_s, y_s, z_s)$ is computed while approaching a new waypoint. It is set a certain step size s away from the quadcopter along its translation vector $\boldsymbol{t} = (t_x, t_y, t_z)$ to the waypoint. Using the intercept theorem the new step coordinates

$$x_s = \frac{t_x \cdot s}{\sqrt{t_x^2 + t_y^2}} \tag{11}$$

$$y_s = \frac{t_y \cdot s}{\sqrt{t_x^2 + t_y^2}} \tag{12}$$

$$z_s = t_z \tag{13}$$

are computed with specifying that the hight remains constant. As soon as the quadcopter is within a certain range of its real target value stepping is deactivated.

Needless to say, the current and next respectively previous markers have to be visible by the camera for calculating each pose. If this is not given while triggering a waypoint change, the new waypoint is only requested and the drone stays over the current marker. When the requested waypoint comes into sight an instant change is executed with setting the requested marker as the current marker. However, if the current marker is lost, it is possible to switch into another flight state like *Safety-Landing*.

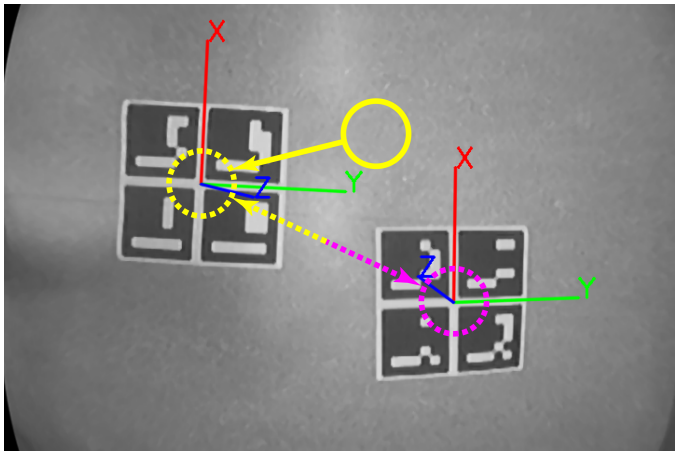


Figure 13. Waypoint navigation with two markers: Yellow circles represent the position of the drone, magenta one the requested position and yellow/magenta lines the flight direction.

V. Results

This chapter describes the results achieved with the proposed control system of the Crazyflie 2.0. It is divided into three parts. In the first one the differences between the output of the *ar_sys*-package and the real position of the drone is determined. The other two present the results during hovering and waypoint navigation.

Table I
DEVIATION BETWEEN THE MEASURED z-DISTANCE AND THE DISTANCE CALCULATED BY *ar_sys*

Measured distance [cm]	Calculated distance [cm]
30	47
40	61
60	94
80	128
100	162

A. Deviation of target position

There is a noticeable difference between the real distance from the quadcopter to the *ArUco* marker and the measured distance. Especially the z-axis shows a high inaccuracy due to an imperfect camera calibration, which is executed with the ROS package *camera_calibration* using 100 images of a calibration pattern. But the fact that the camera uses a wide-angle lens demands a more sophisticated calibration method.

The error between computed and measured heights are not caused by the calculation in *ar_sys*, but these calculations are based on the camera calibration characteristics and thus lead to distorted heights (see Table I). It is remarkable that the measured heights are 1.6 times greater than the real heights.

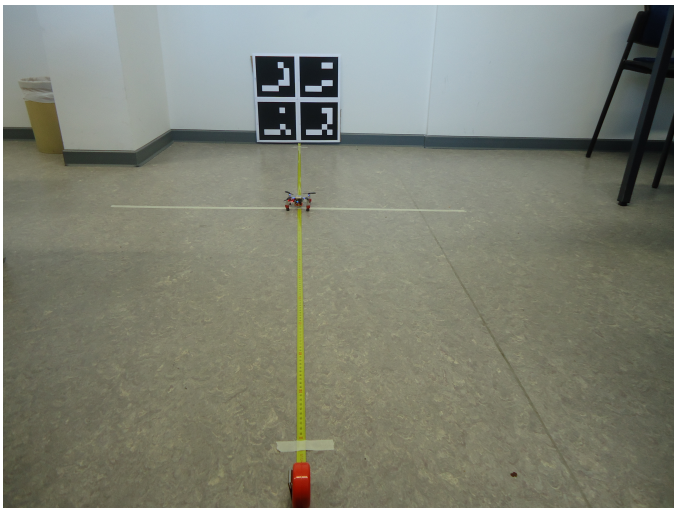
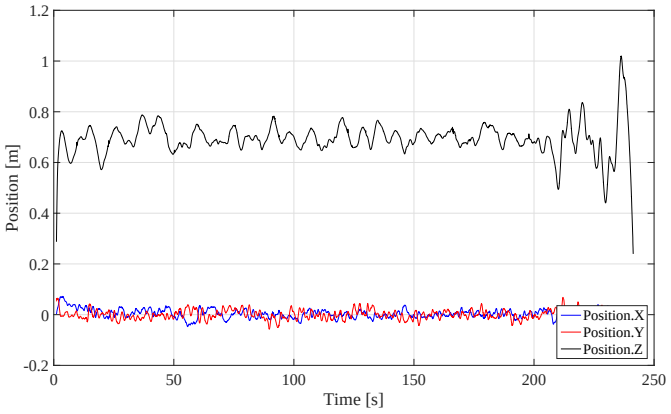


Figure 14. Experimental set-up for measuring the deviation of the target value

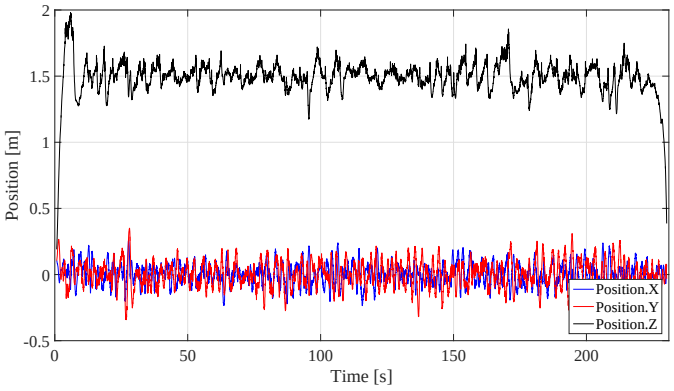
B. Hovering

For examining the robustness of our controller the deviation from the target value on each axis is analysed. Hereby, the target value of the x- and y-axis respectively were constantly set to zero position on a board containing four markers with a side length of 18.6 cm and 1.5 cm padding each during the whole process of evaluating.

First, the deviation on each axis at certain heights is determined. It needs to be mentioned that the command variable position refers to measured values. In Fig. 15 three plots illustrate this scenario. Fig. 15(a) shows a flight at 0.70 m



(a) Target value of 0.70 m.



(b) Target value of 1.50 m.

Figure 15. Deviation and battery status during hovering at different heights

which has a Root Mean Square Error (RMSE) of 3.0 cm in z-direction. Both x- and y-axis are stationary accurate as well with a RMSE of 1.6 cm and are only slightly affected by differences in height. These errors are calculated during a steady state flight. In contrast, a flight at 1.50 m is shown in Fig. 15(b). This height turned out to be the maximum at which a stable and reliable flight is possible, although the RMSE is increased to 7.7 cm in z-direction and 8.9 cm in x-,y-direction, respectively. The reason is that the higher the drone flies, the smaller the amount of pixels that are used for displaying the marker, since the scene enlarges. Thus, the position of the drone can be calculated less precisely.

For emphasising the smooth flight at a height of 0.70 m the battery power is shown in Fig. 16. This shows the graph of a discharging battery as it is known from literature. The flight started with full power at 3.6 V under load condition and the battery is drained until 2.65 V, which is the point when the quadcopter shuts down. Under similar flight conditions the possible flight duration almost stayed the same, wherefore a time of about four minutes and 15 seconds is supposed.

Another test is performed to identify the maximum allowed initial displacement in one axis from the centre of a 2x2

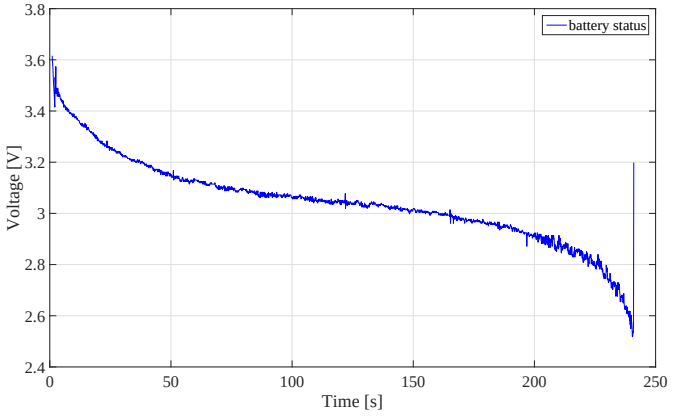


Figure 16. Battery status during the flight with target value of 0.70 m.

marker board of the quadcopter to maintain a reliable take off so that the drone is still able to centre itself above the detected marker. The maximum possible displacement could be identified to 60 cm, whereas reliability is not guaranteed any more. However, multiple tests verified a reliable take off with 50 cm displacement. The adjustment behaviour with this displacement can be seen in Fig. 17, wherein the intended height is 70 cm. Since the start of record is initialised with the first marker recognition, a typical delay of about 1 s exists at the beginning. Hence, the displacement is already shifted somewhat. The quadcopter adjusts the displacement within 5 s after take off, which is a satisfying result. It is remarkable that major corrections in x- and y-direction during take off decrease the quality of the control-behaviour of the height, since it is a coupled system.

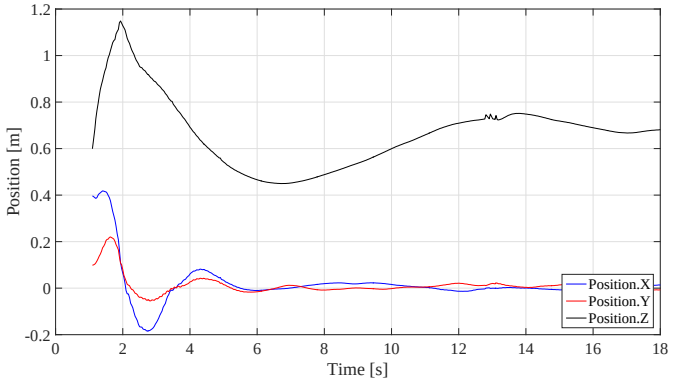
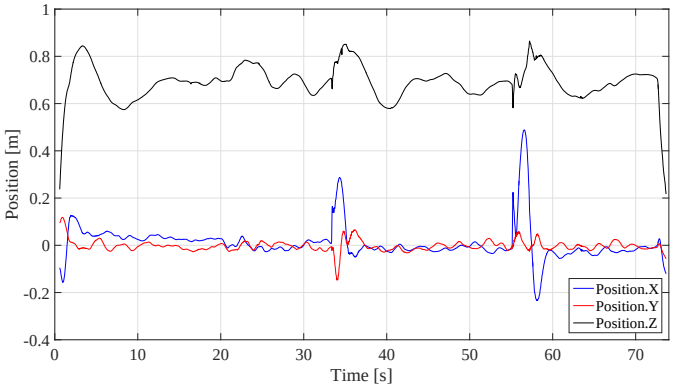


Figure 17. Take off with an initial displacement of 0.5 m in x-direction.

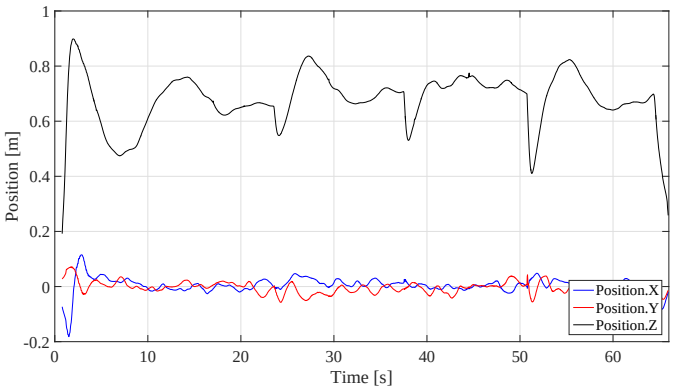
To conclude the single marker board test, the disturbance behaviour of the Crazyflie is analysed. The disturbances are simulated by manually pushing the drone in either x- and y-direction respectively or in z-direction. In Fig. 18(a) it can be seen that the two disturbances in x-direction of 29 cm at 35 s and 49 cm at 55 s are well met by the quadcopter with a response time of 1.8 s and 4.1 s, respectively. In Fig. 18(b) three disturbances by pushing the Crazyflie downwards are done at 24 s, 38 s and 51 s each with increasing force. The drone reacts to these disturbances in a satisfying way, returning to its intended height within 6 s in case of the greatest

distortion of 29 cm. However, it displays an overshoot when adjusting to the disturbances, which also occurs directly after the take off phase. The x- and y-position are minor affected by disturbances in z-direction.

The previous mentioned tests were executed under slight changing light conditions that have no noticeable effect on the flight characteristic.



(a) Distortion in x- and y-direction at 35 s and 55 s.

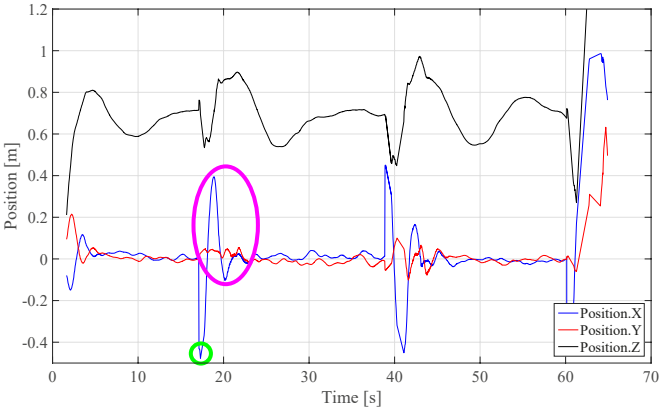


(b) Distortion in z-direction at 24 s, 38 s and 51 s.

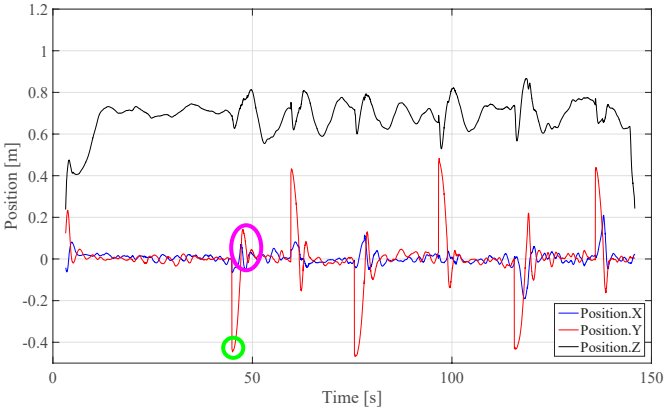
Figure 18. Distortions in different axes during hovering.

C. Waypoint Navigation

The waypoint navigation is evaluated by how fast the quadcopter is stabilising above a new marker and how much overshoot is occurring. In Fig. 19 two plots are showing flight deviations at a height of 0.70 m with both stepping enabled and disabled and a distance between markers of 0.44 m. Each first huge peak on the x- and y-axis means a waypoint change is triggered. In the first image a change without stepping is executed and it can clearly be seen that the overshoot with averagely 0.425 m is massive compared to the second image where a step size of 5 cm and a range of 15 cm reduces this to 0.13 m. Due to a small step size and therefore a small manipulated value, the recovery time with stepping of about 4.83 s is significantly longer than the one without stepping with an average of 3.8 s. However, changing waypoints without a stepping target value or with a large step size results in losing sight of the marker due to the overshoot. This leads to an unstable flight behaviour which happened in the end of the first plot. With a stepping target value this problem was never encountered and the mentioned restrictions are found as the ideal quantities. It is noteworthy, that the maximal distance



(a) No stepping



(b) Step size of 5 cm and range of 15 cm

Figure 19. Waypoint navigation without and with a stepping target value at a height of 0.70 m. An example of a new marker reference is illustrated with a green ellipse and an overshoot sequence with a magenta one.

between serial waypoints is primarily limited by the field of view of the camera. In this work, it is larger horizontally than vertically, therefore the maximal vertical distance has to be selected. At a height of 0.70 m the maximal distance is 0.52 m. If the height is scaled up to 1.00 m, the maximal distance is raised to 0.73 m.

VI. Conclusion and Future Work

In this work, a camera extension of the Crazyflie 2.0 for autonomous flight is presented. The camera and the casing are robust, low-cost, small in size and especially lightweight. The effortless dismounting of the casing guarantees preferable flexibility. The camera sits centrally underneath the quadcopter and provides, regarding its size a high-quality image. However, a flight above a real height of 1 m is not possible due to the restricted resolution of the camera.

Furthermore, we implemented a system allowing to hover above a marker as well as waypoint navigation between different markers. The PID-Controller is optimised for a reliable and stable flight and yields satisfying closed-loop control with stationary accuracy. While hovering a small root mean square error of 3.0 cm in z direction and 1.6 cm in x and

y direction, respectively, occurs at a recommended height of $70\,\mathrm{cm}$. The best results for waypoint navigation regarding the compromise between a recovery time of $4.83\,\mathrm{s}$ and an overshoot of $13.0\,\mathrm{cm}$ are achieved with stepping. Overall the proposed system achieves promising results considering the presented papers in Sec. II.

To equip the quadcopter with a camera has the major drawback of a significantly shorter flight duration. It decreases from seven minutes to about four minutes and 15 seconds. The used battery weights $7\,\mathrm{g}$ and has $240\,\mathrm{mAh}$.

For future work an extended flight duration is desirable. Different alternative batteries with up to $750\,\mathrm{mAh}$ exist, but since they also increase in length there is but one reasonable alternative with $380\,\mathrm{mAh}$ and a weight of $11\,\mathrm{g}$ resulting in a flight duration of 9 minutes without camera [14].

Regarding the motors of the quadcopter, there is no alternative with the same dimensions available and motors with different sizes require a modification of the rotor arms.

Besides, the problem of the deviation between real and calculated heights has to be targeted. Therefore, a proper calibration for the used wide-angle camera needs to be performed.

To extend the possible flight height it is recommendable to research an alternative camera with similar characteristics but better resolution.

QUELLEN

[1] www.bitcraze.io. Crazyflie 2.0. https://www.bitcraze.io/crazyflie-2/, January 2017.
[2] Xavier Gisbert Baguer. Tracking and control system for a mini-quadcopter. Bachelorthesis, TU Darmstadt, 2016.
[3] Gang Hua and Hervé Jégou. Autonomous flight system using marker recognition on drone. Computer Vision - ECCV 2016 Workshops - Amsterdam, 2016.
[4] Pedro Velez, Novel Certad, and Elvis Riuz. Trajectory generation and tracking using the ar.drone 2.0 quadcopter uav. 12th Latin American Robotics Symposium, 2015.
[5] Miguel A. Olivares-Mendez, Somasundar Kannan, and Holger Voos. Setting up a testbed for uav vision based control using v-rep & ros: A case study on aerial visual inspection. *International Conference on Unmanned Aircraft Systems*, pages 447–458, 2014.
[6] Oliver Montague Welton Dunkley. Visual inertial control of a nano-quadrotor. Master's thesis, TU München, 2014.
[7] Diigiit Robotics. Crazyflie 2.0 nano quadcopter. http://www.ca.diigiit.com/image/cache/catalog/bitcraze/A00599-3-1000x750.jpg, January 2017.
[8] www.banggood.com. Rc quadcopter parts. http://img.banggood.com/images/oaupload/banggood/images/51/23/34fc1173-9dbe-40c7-a784-c03933545f35.jpg, January 2017.
[9] S. Garrido-Jurado, R. Muñoz Salinas, F.J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. In *Pattern Recognition*, pages 2280–2292. 2014.
[10] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
[11] Duane C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, pages 444–462, 855–866, 1971.
[12] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision - From Images to Geometric Models*. Springer Science+Business Media, 2012.
[13] www.bitcraze.io. Improving crazyflie 2.0 flight performance. https://www.bitcraze.io/2016/05/improving-crazyflie-2-0-flight-performance/, January 2017.
[14] www.banggood.com. Rc quadcopter parts. http://m.banggood.com/5X-Eachine-3_7V-380mah-25C-Lipo-Battery-for-Hubsan-H107-H107L-H107C-H107D-p-993948.html, January 2017.

Martin Didion studies business administration and electrical engineering at the TU Darmstadt. His area of specialisation is automation, where he received his bachelor's degree in 2016. Currently he is pursuing his master's degree in the same area.

Roman Eisenhauer studies electrical engineering and information technology at the TU Darmstadt. His area of specialisation is automation engineering, where he received his bachelors' degree in 2015. Currently he is pursuing his master's degree in the same area.

Dennis Kraus studies electrical engineering and information technology at the TU Darmstadt. His area of specialisation is automation engineering, where he received his bachelor's degree in 2016. Currently he is pursuing his master's degree in the same area.

Garwin Lechner studies electrical engineering and information technology at the TU Darmstadt. His area of specialisation is automation engineering, where he received his bachelors' degree in 2016. Currently he is pursuing his master's degree in the same area.