Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Localization of Unmanned Aerial Vehicles Using an Optical Flow in Camera Images

Tomáš Novák

Supervisor: Ing. Martin Saska Dr. rer. nat.
Field of study: Cybernetics and Robotics
Subfield: Robotics
May 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Tomáš N o v á k

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Localization of Unmanned Aerial Vehicles Using an Optical Flow in Camera Images

Guidelines:

The goal of the thesis is to design and implement an alternative solution to the well-known px4flow smart camera that would provide a better reliability and enable flying faster, in higher altitude, and in environments with lower illumination. Work plan:

- To implement a Fast Fourier Transform (FFT) based method for estimation of the relative velocity of an Unmanned Aerial Vehicle (UAV) [1, 3] and a proper data post-processing method (possibly Random sample consensus - RANSAC). Consider the limited onboard computational resources (onboard PC) and the required high framerate [2].
- To verify the algorithm using a Gazebo simulator under ROS and real datasets obtained by CTU platform of quadrotor helicopters.
- To compare the FFT solution with a block matching algorithm designed in parallel within Multi-robot systems group and with the px4flow smart camera [4].
- To verify possibility of measuring of a general 3D movement of the UAV using the optical flow from a single camera (rotation in Yaw, 3D translation). Theoretically analyze precision of each component of the UAV motion.

Bibliography/Sources:

[1] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In IJCAI, pages 674-679. William Kaufmann, 1981.
[2] M. R. Balazadeh Bahar and G. Karimian. High performance implementation of the horn and schunck optical flow algorithm on fpga. In 20th Iranian Conference on Electrical Engineering (ICEE2012), 2012.
[3] B. Xian, Y. Liu, X. Zhang, M. Cao, and F. Wang. Hovering control of a nano quadrotor unmanned aerial vehicle using optical flow. In Proceedings of the 33rd Chinese Control Conference, 2014.
[4] D. Honegger, L. Meier, P. Tanskanen and M. Pollefeys, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications," IEEE International Conference on Robotics and Automation, 2013.

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 14, 2017

Acknowledgements

Firstly, I would like to thank the supervisor of this work Martin Saska for his support during the project. Furthermore, big thanks go to other members of Multi-robot Systems group namely Viktor Walter, Vojtěch Spurný and Tomáš Báča for their persistent readiness to help. I would also like thank to my family for their great support and encouragement.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24. May 2017

Abstract

Navigation of Unmanned Aerial Vehicles in GPS-denied environments can be done with multiple techniques. On-board optical flow calculation using single camera gives the user fast-deployable and reliable solution. The goal of this work was to create a replacement for popular *PX4FLOW Smart Camera*, which is burdened by many drawbacks, and to integrate the solution onto a UAV platform. We used Phase correlation for optical flow estimation and a RANSAC-inspired post-processing method. The solution was tested on real-world datasets and compared with *PX4FLOW* sensor. We were able to provide significantly higher accuracy and reliability of horizontal speed measurement in our tests. Moreover, a method for yaw rate and vertical velocity measurement using optical flow in different parts of the image was designed and tested. Tests on real-world datasets showed that the accuracy of the yaw rate estimation method was good enough for practical applications. This makes the method open for usage in magnetometer-denied environments such as reinforced concrete buildings.

Keywords: UAV, localization, optical flow, computer vision

Supervisor: Ing. Martin Saska Dr. rer. nat.

Abstrakt

Existuje mnoho technik navigace bezpilotních letadel v prostředích bez dostupnosti GPS. Kalkulace optického toku na palubě pomocí jedné kamery poskytuje uživateli rychle nasaditelné a spolehlivé řešení. Cílem této práce bylo vytvořit náhradu pro populární senzor *PX4FLOW Smart Camera*, který je zatížen mnoha nevýhodami, a integrovat vzniklé řešení na UAV platformu. Použili jsme fázovou korelaci pro odhad optického toku, pro následné zpracování byla použita metoda inspirována algoritmem RANSAC. Řešení bylo otestováno na datech z reálného světa a porovnáno se senzorem *PX4FLOW*. Byli jsme schopni poskytnout výrazně větší přesnost a spolehlivost měření horizontální rychlosti v rámci našich testů. Dále byla také vytvořena a otestována metoda pro určení vertikální rychlosti a rychlosti rotace, která používá odhadnutý optický tok z více částí obrazu. Testy na datech z reálného světa ukázaly, že přesnost měření rotace je dostatečná pro praktické použití. To umožňuje, aby metoda byla nasazena i v prostředí, kde není možné používat kompas např. v železobetonových budovách.

Klíčová slova: Bezpilotní letoun, lokalizace, optický tok, počítačové vidění

Překlad názvu: Lokalizace bezpilotní helikoptéry analýzou optického toku v obraze

Contents

Figures

Tables

Chapter 1

Introduction

The popularity of small multi-rotor aircraft has been on a constant rise in recent years. They are favourite for their construction simplicity and a small number of moving parts. The UAVs are very often equipped with a semi- or full- autonomous control system that is able to navigate the craft in space and maintain its position. Such functionalities, however, require appropriate position sensors that can provide sufficient update rate and accuracy.

As a localisation sensor, very often a GPS receiver in combination with inertial measurement unit is employed. Such a solution has many advantages like high precision position estimation that does not degrade during the flight and so forth. However, this solution is vitally dependent on reception of a GPS signal.

There are systems that provide high precision localization with high update rate in GPS-denied locations rate such as *Vicon*, *OptiTrack* or *Qualisys*. Nevertheless, these systems require multiple cameras to be installed on the place beforehand, which could be very inconvenient in some applications. Furthermore, the cost of such systems is very high.

Another group of localisation systems does not rely on any external infrastructure. In uses solely sensors placed on the aircraft. Very often, they use cameras to track the movement of steady surroundings to gain information about the craft's velocity and position. One of the most known and often used sensors in this group is *PX4FLOW Smart Camera* that tracks the velocity of the ground.

However, the *PX4FLOW Smart Camera* has very constrained parameters. The working altitude limit is very low, the requirements on the illumination of the ground are relatively high, and maximal reachable velocity is quite limited as well.

The goal of this thesis is to develop a substitution for *PX4FLOW Smart Camera* that uses hardware already available on the UAVs of the Multi-robot

systems group. Additionally, it should not share the drawbacks of the smart camera.

Our motivation is to build a tool that enables application of formation flying approaches [SVKP14][SKV+14][SBS16][SKP14][SKV+13][SVKP12][SBH16], being designed in a long-term research within the MRS group, in GPS-denied environments. In particular, we aim to deploy these principles for scanning of interiors of large historical building by a formation of cooperating UAVs [SKSB17] [`http://mrs.felk.cvut.cz/projects/cesnet`] and for surveillance [SVC+16][SBT+16][SCP+14] and stabilization of UAV swarms [Sas15][SVP14] in forests. All of these applications require to control the UAVs in environment with insufficient light conditions, in which PX4FLOW sensor cannot be used. The presented solution is planned to be used in feedback onboard Model Predictive Control (MPC) [BLS16] together with mutual localization of UAVs in the group [FKC+13][KNF+14], which enables fully autonomous flight in these demanding workspaces. Therefore, the proposed method is designed to fulfill requirements of these applications.

We thus created a method that uses the same principles as *PX4FLOW* sensor. It uses the on-board Bluefox camera to track ground movement and combines this information with readings from a gyroscope and an optical altitude sensor to create horizontal velocity estimates. The whole system is highly versatile thanks to its implementation as a module in the Robot Operating System. It can be easily modified to use different sensors.

Also, for optical flow calculation, we took a different approach than the *PX4FLOW* sensor, which uses a very simple method called Block Matching. Our solution gives a possibility to choose between Block Matching[1] or Phase Correlation. The second technique harvests the properties of Fourier transform. By taking this path, we are able to measure higher speeds with lower frame rate requirements.

Additionally, we created a method that can provide not just horizontal velocity, but also vertical velocity and yaw rate measurement using the same optical flow method.

[1] The Block Matching method was implemented in parallel to this work within Multi-robot systems group.

Chapter 2

State of the art

The camera localisation systems for UAVs are a well-researched area. Already mentioned systems using external cameras such as *Vicon*, *OptiTrack* or *Qualisys* are widely available. We also already mentioned *PX4FLOW Smart Camera* developed by Honegger et al. [HMTP13]. This sensor tracks ground using its on-board high-speed camera and ultrasonic range sensor to provide speed and position estimation for UAVs. We describe the sensor in more detail in section 3.3.

A similar work using dedicated hardware has been proposed by Heinrich [Hei17]. The sensor mimics the concept of *PX4FLOW* but it uses Raspberry Pi computer in combination with a compatible *Sony IMX219* camera. Probably the main feature is the usage of hardware H.264 video format decoder already integrated on the computer board for optical flow estimation. This shrinks the CPU load significantly. The verification shows about the same performance as *PX4FLOW*. Unfortunately, no verification under conditions, where *PX4FLOW* sensor stops working (such as high altitude) was done. Additionally, the construction used would be hard to integrate onto the existing UAV, because a Raspberry Pi would have to be installed on board.

Aasish et al. in [CEU+15] present an implementation of Horn and Schunk optical flow method [HS81] for two-dimensional position estimation. However, they show only a general concept and do not include any results from testing.

Self-motion estimation using visual systems is done by More et al. in [MKK+15]. They use feature detection implemented in OpenCV library along with iterative pyramidal Lucas-Kanade optic flow calculation method to make the estimation. The height measurement is done with an ultrasonic sensor. They also implement correction methods to deal with terrain height variation and tilting of the camera.

Santamaria-Navarro et al. in [SNSAC15] present a solution for 3D odometry sensor with *PX4FLOW* camera and a low-cost IMU unit. They test more variants of the Kalman filter and propose different covariance matrices for

different flight stages. They were able to show that a plain EKF filter performs equivalently to nonlinear versions with the given setup. They also proposed methods to overcome situations, where *PX4FLOW* gives erroneous readings. This solution, however, does not solve the problem of poor performance of the *PX4FLOW* in worse lighting conditions.

As Heinrich points out in [Hei17], computer mouse sensors are very popular for visual odometry, especially amongst hobbyists. He references Briod et al. in [BZF13] who uses multiple mouse sensors. Their method, unlike other cited, does not scale the readings of optical flow, but just creates a direction constraint that corrects for the drift of on-board IMU unit. This technique dramatically decreases the position estimation error compared to pure IMU solution. Another example according to Heinrich would be Kim et Brambley [KB07]. They use two mouse sensors to overcome the unknown depth information in combination with EKF. Gageik et al. in [GSM13] also test a mouse optical sensor in an implementation of an autonomous UAV.

So-called FPGAs (Field-programmable gate arrays) can be used to achieve our goal with many advantages such as low power consumption and small latency. This kind of computing hardware is employed for visual motion detection in [AFF04] or [BK12]. In the latter Bahar and Karimian describe implementation and testing of Horn and Schunck algorithm [HS81] on FPGA. In the former Aubepart et al. utilise more biologically inspired sensors and FPGA chip for visual guidance of an UAV. The sensor mimics housefly eye's properties. It consists of a linear array of angular sensitive light sensors, so-called Elementary Motion Detectors.

Both of these approaches would, however, require undesirable additional hardware. Moreover, additional development would be needed, because the authors presented just a general concept and did not focus on potential problems with real world application on UAVs.

Xian et al. in [XLZ+14] are implementing an approach that uses off-board computing power in order to compute optical flow to stabilise a nano UAV. They are successful in stabilising the aircraft in the radius of 20 cm from given point. This application is, however, unsuitable for our case, since it requires an external computer and other hardware such as a video transmitting device. Secondly, it does not estimate the position of the craft.

Iyer et al. in [IHC05] present a method for motion estimation from image sequences that doesn't use a common pinhole camera model but is inspired by compound eye found in e.g. house flies. It is purposed to be used on UAVs, but according to the paper, no experiments were done on this platform at the time. This new method is further developed in [IMB08] and [ITH08].

Another biologically inspired approach is presented by Conradt in [Con15]. It is intended to be used on miniature UAVs but all calculations are done on board. To achieve this, he uses a special event-based vision sensor where each pixel asynchronously emits a signal on change of the incoming light intensity.

These both works, though being very innovative, would require a special kind of sensor, which is very undesirable because of its price and availability on the market.

In our work, we also present a method for yaw rate and vertical velocity estimation from optical flow using a single camera. Stowers et al. in [SBSHM09] use a camera with fisheye lens for yaw speed measurement of an UAV. They do a log-polar transformation of the captured images in combination with Phase correlation. This allows them to estimate the rotational movement between the frames. However, this method demands more processing power than the method proposed in our work, which is caused by the need of log-polar transformation.

Joos et al. in [JZS10] present a method that estimates horizontal velocity and yaw rate using two optical mouse sensors. The yaw speed is estimated by a simple trigonometry. However, the method is presented on an automotive vehicle. Implementation on an UAV would bring a new set of problems.

Chapter 3

System overview

This chapter describes the hardware applied in this work as well as its constraints and technical parameters, which do not just define accuracy limits for our algorithm but also shape the possible requirements. We also briefly discuss the PX4FLOW Smart Camera we aim to replace with this work.

3.1 Hexacopter

The employed hexacopter uses a slightly modified *F550 frame* made by *DJI*[1]. Its diagonal wheelbase (distance between opposite motors) is 550 mm and its take-off weight is about 3 kg.

The UAV is equipped with *Intel NUC i7* computer, which provides processing power for control algorithms as well as for our localisation algorithm. Control is done in combination with *PixHawk* autopilot[2].

Figure 3.1: Photo of the hexacopter in flight.

[1] See `http://dji.com/flame-wheel-arf/spec`
[2] See `https://pixhawk.org/modules/pixhawk`

Velocity constraints are given by the control algorithms and construction. The maximal horizontal velocity is limited to 8.33 $\mathrm{m\,s^{-1}}$, the vertical velocity in constrained to 2 $\mathrm{m\,s^{-1}}$ while ascending and to 1 $\mathrm{m\,s^{-1}}$ while descending. The angular velocity in yaw has a maximum of 1 $\mathrm{rad\,s^{-1}}$. We should note that these are the maximal ratings, in the real-world application of our algorithm the horizontal velocity will be limited to approximately 4 $\mathrm{m\,s^{-1}}$ and the yaw rate to 0.5 $\mathrm{rad\,s^{-1}}$.

3.2 Sensory equipment

Three key sensors are employed in our solution - camera, an altitude sensor and gyroscope. In this section, we aim to describe the technical parameters of these sensors.

3.2.1 Bluefox camera

In the work *mvBlueFOX-MLC200w* camera by *Matrix Vision* is used as the main sensor[3]. Its chip has a resolution of 752×480 of grayscale pixels. It is equipped with global shutter and is capable of a maximal frame rate of 93 Hz. The camera is also capable of pixel binning mode, in which the resolution is halved but the frame rate can go up to 170 Hz.

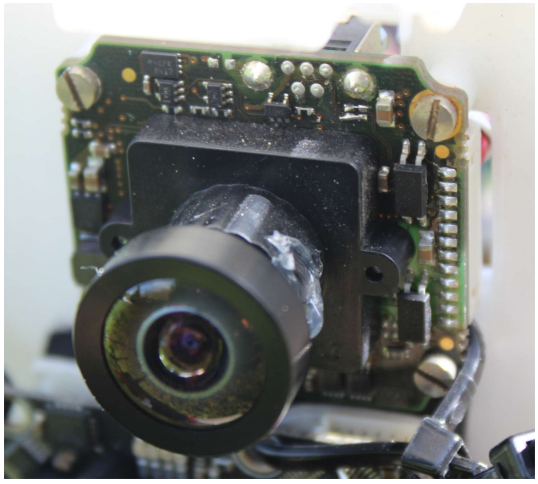


Figure 3.2: Photo of the Bluefox camera the 2.1 mm lens. The camera is installed on the UAV facing downwards.

After experimentation with different lenses, we decided to use lens *MV-O-SMOUNT 02.1 TN0212B* by *Matrix Vision* in the final application. It is a 1/3" lens with focal length of 2.1 mm that provides a big field of view. This

[3] See `https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html`.

choice allows us to keep the frame rate relatively low thus also lowering the computational power required.

3.2.2 Altitude sensor

As a source of altitude measurements, we use infra-red distance measurement sensor *TerraRanger One*[4] by *Terabee* that takes advantage of time-of-flight measurements. Its specified maximal range is 14 m, but it decreases to 6 m in sunlight. It has a very narrow field of view of only 3°. The claimed accuracy is up to ±4 cm.

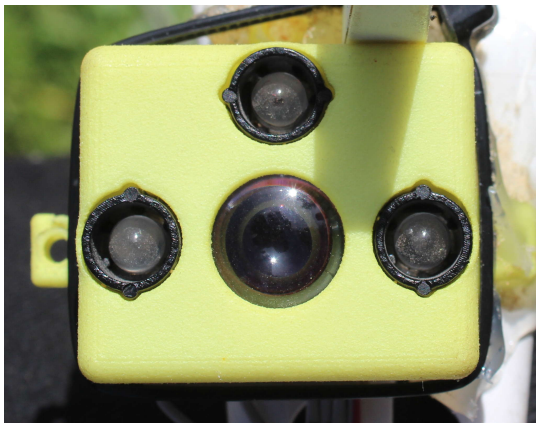


Figure 3.3: Photo of the Terraranger altitude sensor.

3.2.3 Angular rate sensor

Angular rate is measured within the *PixHawk* autopilot unit. This unit combines two separate three-axis gyroscopes. One of them is *ST Micro L3GD20H* and the second one *Invensense MPU 6000*. However, no accuracy estimations are known to the author to be available.

3.2.4 Position estimation system

In the real-world experiments, we compare the performance of velocity and position estimation of our method to UAV's internal estimates. They are derived from RTK satellite navigation module *PRECIS-BX305*[5] that works in coordination with *PixHawk* inertial measurement unit and *TerraRanger One* sensor. The RTK module itself is supposed to have an accuracy of horizontal

[4] See `http://teraranger.com/products/teraranger-one/`

[5] See `https://tersus-gnss.com/collections/rtk-boards-receivers/products/precis-bx305`.

positioning 10 mm + 1 ppm, vertical positioning 15 mm + 1 ppm and velocity accuracy of 0.03 $\mathrm{m\,s^{-1}}$ [TG17].

3.3 PX4FLOW Smart Camera

The PX4FLOW sensor comprises a camera with *MT9V034* CMOS sensor, 16bit high-precision gyroscope and sonar for altitude measurements. It calculates the optical flow present in the image, combines it with angular rate and altitude measurements and outputs horizontal velocities in both axes as well as altitude [PX413].

The camera has a resolution of 752×480 pixels, but the image is binned four times in each axis, which allows the chip to work at a frame-rate of 400 Hz. It can be used with different lenses, in our setup we use 16 mm focal length.

The manufacturer presents theoretical maximal measurements with different focal lengths in different altitudes displayed in table 3.1. However, practical usage showed the altitude to be limited to around 5 m and maximal horizontal velocity to be circa 2 $\mathrm{m\,s^{-1}}$ at 1.5 m altitude.

Altitude	**1 m**	**3 m**	**10 m**
16 mm lens	2.4 m/s	7.2 m/s	24 m/s
8 mm lens	4.8 m/s	14.4 m/s	48 m/s
6 mm lens	6.4 m/s	19.2 m/s	64 m/s
4 mm lens	9.6 m/s	28.8 m/s	96 m/s

Table 3.1: PX4FLOW sensor theoretical maximal velocities under different conditions [PX413].

Chapter 4

Preliminaries

This chapter describes the theory behind the core techniques used in our approach to the problem. In the beginning, we present Motion Field equations describing the relationship between camera movement and the amount of movement in the image. These are used later in the next section for horizontal movement estimation as well as for three-dimensional translational velocity and yaw rotation estimation.

In the next section, we introduce two methods of optical flow calculation. One of them, block matching, was implemented in parallel within Multi-robot systems group. The second one, Phase correlation, was implemented as a part of this work.

Finally, we describe post-processing methods used for filtering of the data gathered by these algorithms.

4.1 Motion Field equations

Following the same principles as in [HMTP13], we model the effect of movement of the camera on the optic flow present in the frames. We consider the situation illustrated in figure 4.1. The projection equation of the pinhole camera model is as follows

$$\boldsymbol{p} = f\frac{\boldsymbol{P}}{P_z}, \tag{4.1}$$

where $\boldsymbol{P} = [P_x, P_y, P_z]$ is a point in the camera reference frame, $\boldsymbol{p} = [p_x, p_y, f]$ is its projection onto an image plane I and f is focal length in terms of pixels.

Let us denote the velocity of the camera as $\boldsymbol{T} = [T_x, T_y, T_z]$ and its angular velocity as $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$. Then the velocity $\boldsymbol{V} = [V_x, V_y, V_z]$ of the point $\boldsymbol{P}$ will be:

$$\boldsymbol{V} = -\boldsymbol{T} - \boldsymbol{\omega} \times \boldsymbol{P}. \tag{4.2}$$

By taking the derivative with respect to time from both sides of equation (4.1), we get the velocity $\boldsymbol{v} = [v_x, v_y, v_z]$ of projected point $\boldsymbol{p}$:

$$\boldsymbol{v} = f\frac{d}{dt}\left(\frac{\boldsymbol{P}}{P_z}\right) = f\frac{P_z\boldsymbol{V} - V_z\boldsymbol{P}}{V_z^2}, \tag{4.3}$$

since the time derivative of $\boldsymbol{P}$ is $\frac{d}{dt}(\boldsymbol{P}) = \boldsymbol{V}$. After separating $\boldsymbol{v}$ to each component and substituting (4.2), we get:

$$\begin{aligned} v_x &= \frac{f}{P_z^2}\left(-P_zT_x - P_z^2\omega_y + P_yP_z\omega_z + P_xT_z - P_xP_y\omega_x + P_x^2\omega_y\right), \\ v_y &= \frac{f}{P_z^2}\left(-P_zT_y - P_z^2\omega_x + P_xP_z\omega_z + P_yT_z + P_y^2\omega_x - P_xP_y\omega_y\right). \end{aligned} \tag{4.4}$$

Afterwards, we substitute P_x and P_y with p_x, p_y according to (4.1). The result is expressed as:

$$\begin{aligned} v_x &= \frac{T_zp_x - T_xf}{P_z} - f\omega_y + p_y\omega_z + \frac{\omega_xp_xp_y - p_x^2\omega_y}{f}, \\ v_y &= \frac{T_zp_y - T_yf}{P_z} + f\omega_x - p_x\omega_z + \frac{p_y^2\omega_x - \omega_yp_xp_y}{f}. \end{aligned} \tag{4.5}$$

We assume the effect of the last term divided by f to be negligible. Thus, we write:

$$\begin{aligned} v_x &\approx \frac{T_zp_x - T_xf}{P_z} - f\omega_y + p_y\omega_z, \\ v_y &\approx \frac{T_zp_y - T_yf}{P_z} + f\omega_x - p_x\omega_z. \end{aligned} \tag{4.6}$$

These equations describe how the optical flow at each point on the image plane (p_x, p_y) is affected by translational movement of the camera (T_x, T_y, T_z) and its angular velocities $(\omega_x, \omega_y, \omega_z)$ with the knowledge of the depth position P_z of the projected point and the focal length f.

4.2 Optical flow estimation

In our work, we tested two approaches to optical flow calculation. The first, Block matching, compares the intensities of each pixel to find a perfect alignment of two frames. The second, Phase correlation, uses the properties of Fourier Transform to calculate a correlation between two frames.

4.2.1 Block matching

Block matching is a common technique for estimation of translational motion between two frames. It has been studied since the mid-1970s and many

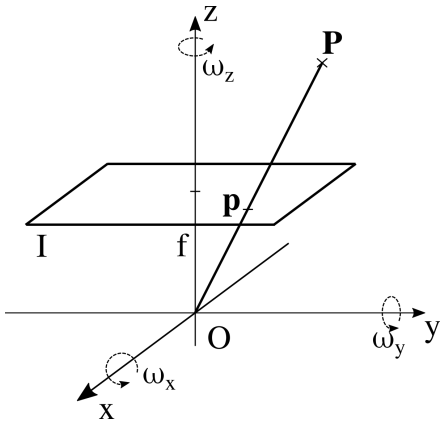


Figure 4.1: Illustration showing projection of a point in space to an image plane. The camera aperture is placed at the origin of coordinate system. The point P is projected to an image plane I of a camera with focal length f. Rotational velocities along each axis are denoted as ω_x, ω_y and ω_z.

modifications were proposed. They vary in the metric that is used to match two pixels and in a reduction of the steps that are needed to find an optimal match.

We must note that an implementation of this algorithm is not a part of this work, but we describe it here for better understanding of the subject. The implementation was inspired by the optical flow calculation method in [HMTP13].

First, let us describe the image difference by sum of absolute differences [Sze10]:

$$E_{SAD}(\boldsymbol{u}) = \sum_i |I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)|, \tag{4.7}$$

where I_0 and I_1 are two frames, vectors $\boldsymbol{x}_i = (x_i, y_i)$ point to all discrete pixel locations in a frame and vector $\boldsymbol{u}$ is a displacement vector, which describes the translational movement between the two frames. Our goal is to find such vector $\boldsymbol{u}$ from all possible displacement vectors, so that the $E_{SAD}(\boldsymbol{u})$ is minimal.

The employed algorithm divides the image into n square sections of certain size and distance. These sections can overlap. For each section the method tries to estimate $\boldsymbol{u}$. Mathematically, we could express this by defining sets $S_1, S_2, \ldots, S_n$ where each set contains vectors pointing to pixels in one section of the image.

$$S_i = \{\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}, \ldots, \boldsymbol{x}_{1,w/n}, \boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}, \ldots, \boldsymbol{x}_{w/n,w/n}\}, \forall i \in \langle 1, n \rangle, \tag{4.8}$$

where w is the dimension of the whole frame (frames are cropped to have square dimensions) and the vectors in one set S_i point to all pixels in i-th square section of the image. The number of sets n must be chosen such that the frame dimension is divisible by it.

We also have to define a set of displacement vectors U. It contains all considered displacement positions. The algorithm performs a search for minimal E_{SAD} over these positions. We define the search radius r in pixels so that U contains vectors:

$$U = \{(-r, -r), (-r+1, -r), \ldots, (r, -r), (-r, -r+1), \ldots, (r, r)\}. \quad (4.9)$$

Now, we can find for every set S_i (i.e. for every square section of the image) $\boldsymbol{u}_i$ such that

$$E_{SAD}(\boldsymbol{u}_i) = \min_{\boldsymbol{k} \in U} \sum_{x_{i,j} \in S_i} |I_1(\boldsymbol{x}_{i,j} + \boldsymbol{k}) - I_0(\boldsymbol{x}_{i,j})|. \quad (4.10)$$

After this step we have to combine the output from each section into one output. A histogram is created for estimated displacements separately for each dimension, i.e. if we denote $\boldsymbol{u}_i = (u_{i,x}, u_{i,y})$, then a histogram is constructed from all $u_{i,x}$ and $u_{i,y}$. From these histograms the number with highest occurrence u_x, u_y is picked for each axis. Thus, the output of the algorithm is $\boldsymbol{u} = (u_x, u_y)$.

4.2.2 Phase correlation

Another method to find a translational alignment is called Phase correlation. It benefits from the properties of Fourier transform to calculate correlation between two images. In our solution, we took advantage of the already available implementation in OpenCV library [Its17].

First, let us denote two image frames $I_1(\boldsymbol{x})$ and $I_2(\boldsymbol{x})$. Let I_2 be circularly shifted by vector $\boldsymbol{u}$, i.e. for all possible $\boldsymbol{x}$:

$$I_1(\boldsymbol{x}) = I_2((x_x + u_x) \bmod d_x, (x_y + u_y) \bmod d_y), \quad (4.11)$$

where $\boldsymbol{x} = (x_x, x_y)$ and $\boldsymbol{u} = (u_x, u_y)$ and d_x, d_y are the dimensions of the frames. In reality, the condition for circular shift is not satisfiable but experiments show that the method is able to work without the condition up to certain shift magnitudes.

In order to find $\boldsymbol{u}$, we first take the Fourier transform of both frames using FFT[1], thus we obtain $\mathcal{I}_1$ and $\mathcal{I}_2$, whereby $\mathcal{I}$ we denote a complex Fourier spectrum of an image. Then, using these spectra, we calculate Phase correlation function (more precisely its Fourier transform) [Sze10]:

$$\mathcal{F}\{E_{PC}(\boldsymbol{x})\} = \frac{\mathcal{I}_1(\boldsymbol{\omega})\mathcal{I}_2^*(\boldsymbol{\omega})}{\|\mathcal{I}_1(\boldsymbol{\omega})\|\|\mathcal{I}_2(\boldsymbol{\omega})\|}, \quad (4.12)$$

where $\mathcal{I}_2^*(\omega)$ denotes the complex conjugate of the spectrum.

[1] To remove edge effects, a Hanning window function is applied beforehand.

For this function

$$\mathcal{F}\{E_{PC}(\boldsymbol{x})\} = e^{-j\boldsymbol{\omega}\cdot\boldsymbol{u}} \tag{4.13}$$

holds. To show this, we take the Fourier transform of both sides of equation (4.11). By using properties of the transform we can write:

$$\mathcal{F}\{I_1(\boldsymbol{x})\} = \mathcal{F}\{I_2(\boldsymbol{x}+\boldsymbol{u})\}, \tag{4.14}$$

$$\mathcal{I}_1(\boldsymbol{\omega}) = \mathcal{I}_2(\boldsymbol{\omega})e^{-j\boldsymbol{\omega}\cdot\boldsymbol{u}}. \tag{4.15}$$

Then we can substitute to equation (4.12):

$$\mathcal{F}\{E_{PC}(\boldsymbol{x})\} = \frac{\mathcal{I}_2(\boldsymbol{\omega})\mathcal{I}_2^*(\boldsymbol{\omega})}{\|\mathcal{I}_2(\boldsymbol{\omega})e^{-j\boldsymbol{\omega}\cdot\boldsymbol{u}}\|\|\mathcal{I}_2(\boldsymbol{\omega})\|}e^{-j\boldsymbol{\omega}\cdot\boldsymbol{u}} = e^{-j\boldsymbol{\omega}\cdot\boldsymbol{u}}. \tag{4.16}$$

Now, we take the inverse Fourier transform of the function calculated in (4.12) and get:

$$E_{PC}(\boldsymbol{x}) = \delta(\boldsymbol{x}-\boldsymbol{u}), \tag{4.17}$$

where $\delta(\boldsymbol{x}-\boldsymbol{u})$ denotes an impulse shifted by $\boldsymbol{u}$.

Afterwards, the task is to find the peak in E_{PC}. First, the maximum is found in the phase correlation function [Its17]:

$$\boldsymbol{u}_m = (u_{mx}, u_{my}) = \arg\max_{\boldsymbol{x}}(E_{PC}(\boldsymbol{x})). \tag{4.18}$$

To achieve sub-pixel accuracy, we compute a centroid weighted by intensities of E_{PC} in the 5×5 neighbourhood R of $\boldsymbol{u}_m$:

$$\begin{aligned} u_x &= \frac{1}{\sum_{\boldsymbol{x}\in R}E_{PC}(\boldsymbol{x})}\sum_{\boldsymbol{x}\in R}x_x E_{PC}(\boldsymbol{x}), \\ u_y &= \frac{1}{\sum_{\boldsymbol{x}\in R}E_{PC}(\boldsymbol{x})}\sum_{\boldsymbol{x}\in R}x_y E_{PC}(\boldsymbol{x}). \end{aligned} \tag{4.19}$$

obtaining $\boldsymbol{u} = (u_x, u_y)$ as a result of the estimation.

4.3 Data post-processing

The output data from optical flow calculation methods sometimes contain outliers, especially when an optically uniform area is contained in a part of the image. The image is segmented into more rectangular parts, and optical flow is calculated for each one separately to avoid this issue (this is further described in 5.1). Having more estimates available at a time, we can employ an outlier detection technique inspired by Random sample consensus method.

Let U be the set of available estimates:

$$U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}, \tag{4.20}$$

where $n \geq 2$. We will compute an average vector $\boldsymbol{a}_{i,j}$ for each pair, that is for $\forall \boldsymbol{u}_i, \boldsymbol{u}_j \in U; i, j \in \langle 1, n \rangle; i < j$:

$$\boldsymbol{a}_{i,j} = \frac{1}{2} \left(\boldsymbol{u}_i + \boldsymbol{u}_j \right). \tag{4.21}$$

Now, we construct a set $E_{i,j}$ for each vector $\boldsymbol{a}_{i,j}$ with elements from U that are placed in a radius of r from $\boldsymbol{a}_{i,j}$:

$$E_{i,j} = \{ \boldsymbol{u}_i \in U \text{ such that } \| \boldsymbol{u}_i - \boldsymbol{a}_{i,j} \| \leq r \}. \tag{4.22}$$

Radius r is a tunable parameter that is adjusted in section 6.1.

Afterwards, we pick a set E from all $E_{i,j}$ that has the most elements:

$$E = \underset{E_{i,j}}{\arg\max} \left(\text{card} \left(E_{i,j} \right) \right). \tag{4.23}$$

If more than one set has the maximal number of elements, the first found is taken.

Finally, we create output vector $\boldsymbol{u}$ as an average of elements in E:

$$\boldsymbol{u} = \frac{1}{\text{card}(E)} \sum_{\boldsymbol{e} \in E} \boldsymbol{e}. \tag{4.24}$$

Using this method, we can compute an average only from such vectors that are within a certain distance from each other. This ensures that all outliers present in the input set do not have any impact on the final result. The key parameter for the method is the threshold radius r which has to be determined empirically.

Chapter 5

Implementation overview

This chapter describes the movement estimation system. First, we will describe the function of two-dimensional motion estimation from an optical flow that was inspired by [HMTP13]. Afterwards, we present a method that can estimate translational as well as yaw velocity.

5.1 Horizontal velocity estimation from optical flow

This chapter discusses the approach to horizontal velocity estimation of the UAV from optical flow. We describe the method as a whole and the approximations made in our approach.

Image data is first loaded from the camera and then they are converted to grayscale and cropped into a square image. Optical flow is then measured by comparing two consecutive images. To gain more readings at a time for better post-processing possibilities, we measure the translational alignment in more parts of the picture. The frame is thus separated into square sections without overlaps and Phase correlation in calculated for each one. We will assume that we get n optical flow estimates from each image section[1]:

$$\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n. \tag{5.1}$$

The exact number is subject to experimental tuning. In chapter 6.2 we estimate the optimum to be sixteen frame sections.

We assume that the ground below the UAV is flat so that all sections of the camera frame are at the same vertical level. Thanks to the assumption, we need to obtain only simple altitude measurement denoted by A. This is done

[1] This is, however, true only for measurements with Phase correlation. Block matching method, that was not implemented as a part of this work, yields only one estimate at a time that is already filtered by histogram filter.

by the on-board *TerraRanger One* sensor described in chapter 3.2.2. If this assumption is not fulfilled, the optical flow caused by horizontal translation has different values in different parts of the image, which could lead to inaccurate or noisy readings.

Rotation in all roll, pitch and yaw axes can have a significant impact on the accuracy of the measurement. They cause an undesired optical flow present in the image. Thus, we have to compensate these effects by measuring angular rates. We get the measurements from on-board gyroscope further described in chapter 3.2.3. Since x-,y- and z- axes of the camera are aligned to the drone's axes, we do not perform any further recalculation of the angular velocities. Further, we take the assumption that the camera is placed in the centre of rotation. In the real implementation, this is not entirely true, but the effect on the readings is negligible. Let us denote the angular velocities by ω_x, ω_y and ω_z.

Now, we take into account the simplifying assumptions above and also that the vertical velocity has insignificant effect on the readings. Then, for each optical flow measurement $\boldsymbol{u}_i = (u_{x,i}, u_{y,i})$ the estimated translation according to equations (4.6) can be calculated as follows [HMTP13]:

$$\begin{aligned} T_{x,i} &\approx \frac{A}{f}\left(-u_{x,i} - f\omega_y + p_{y,i}\omega_z\right), \\ T_{y,i} &\approx \frac{A}{f}\left(-u_{y,i} - f\omega_x + p_{x,i}\omega_z\right), \end{aligned} \tag{5.2}$$

where $p_{x,i}, p_{y,i}$ are the coordinates of the center of the given image section in the image reference frame (origin is in the center of the frame) and f is the focal length of the camera in terms of pixels.

Finally, post-processing method described in chapter 4.3 is used on vectors $\boldsymbol{T}_i = (T_{x,i}, T_{y,i})$. It aims to neglect the effect of any outliers and yields final velocity estimate.

5.2 3-D velocity and yaw rate estimation

Using the properties of the Motion Field described in 4.1, we can not only estimate the horizontal velocity, but it is also possible to get the vertical and yaw speed estimates.

In the method, image is loaded from the camera, cropped and grayscaled in a way similar to previous chapter. It is then symmetrically divided into nine parts. For each part, we separately calculate optical flow. Let us denote the optical flow in each section vectors as

$$\boldsymbol{v}_{11}, \boldsymbol{v}_{12}, \boldsymbol{v}_{13}, \boldsymbol{v}_{21}, \ldots, \boldsymbol{v}_{33} \tag{5.3}$$

as it is shown in figure 5.1.

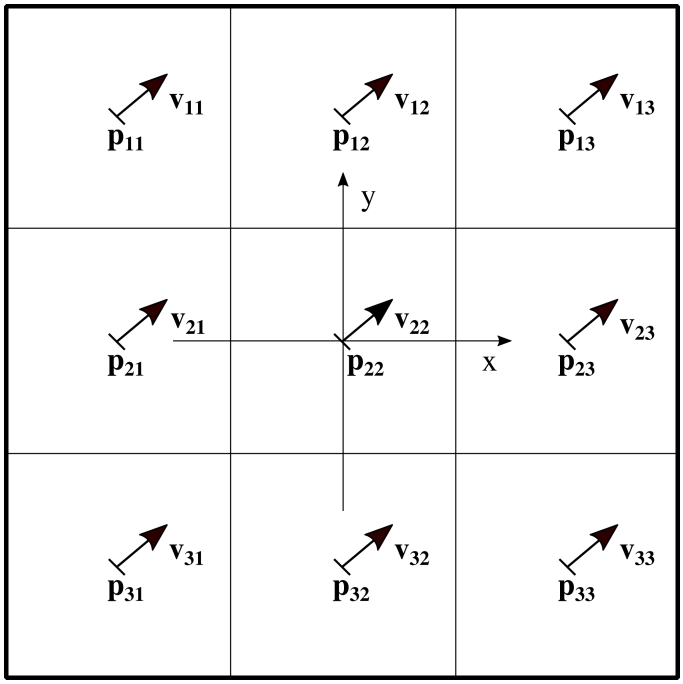


Figure 5.1: Illustration showing notation of optical flow vectors in parts of the frame. The square frame is separated into nine parts with centres at $\boldsymbol{p}_{ij}$. The optical flow vector in each part is denoted as $\boldsymbol{v}_{ij}$.

Taking an assumption of flat ground beneath the UAV and taking other approximations described in the section above, each of the vectors $\boldsymbol{v}_{ij}$ can be approximately expressed according to equations (4.6) as a sum of five different factors:

$$\begin{aligned} v_{ij,x} &\approx -\frac{f}{A}T_x + \frac{p_{ij,x}}{A}T_z + p_{ij,y}\omega_z - f\omega_y, \\ v_{ij,y} &\approx -\frac{f}{A}T_y + \frac{p_{ij,y}}{A}T_z - p_{ij,x}\omega_z + f\omega_x, \end{aligned} \tag{5.4}$$

where $(p_{ij,x}, p_{ij,y})$ is the center of each square section, A is the altitude measured by *TerraRanger One* sensor, (T_x, T_y, T_z) is translational and $(\omega_x, \omega_y, \omega_z)$ angular velocity of the UAV.

First, we use measurement of the angular rate in roll and pitch axes from on-board gyroscope to compensate for effect of ω_x and ω_y in the equations. We thus obtain vectors $\boldsymbol{r}_{ij}$ as follows:

$$\boldsymbol{r}_{ij} = \boldsymbol{v}_{ij} + f\left(\omega_y, -\omega_x\right). \tag{5.5}$$

As a next step, we estimate horizontal velocity T_x and T_y. We take advantage of the fact that the effect of T_z and ω_z on the motion field vectors is centrally symmetric - the factors are being multiplied by p_x or p_y. That means, we eliminate the effect by summing the opposite vectors. Hence, we

can calculate four horizontal velocity estimates $\boldsymbol{T}_i$ as follows:

$$\boldsymbol{T}_1 = \frac{-A}{2f}\left(\boldsymbol{r}_{11} + \boldsymbol{r}_{33}\right), \boldsymbol{T}_2 = \frac{-A}{2f}\left(\boldsymbol{r}_{12} + \boldsymbol{r}_{32}\right), \ldots, \boldsymbol{T}_4 = \frac{-A}{2f}\left(\boldsymbol{r}_{21} + \boldsymbol{r}_{23}\right). \tag{5.6}$$

They are fed to our outlier filtering method described in section 4.3 along with estimation from central vector $T_5 = \frac{-A}{f}\boldsymbol{r}_{22}$ which is not affected by vertical nor yaw movement. We thus obtain an estimation of horizontal velocity $\boldsymbol{T}$.

By adding the effect of horizontal velocity to vectors $\boldsymbol{r}_{ij}$, we create vectors

$$\boldsymbol{k}_{ij} = \boldsymbol{r}_{ij} + \frac{f}{A}\boldsymbol{T}. \tag{5.7}$$

These vectors should be simply a product of yaw and vertical movement[2], since we can write:

$$\begin{aligned} k_{ij,x} &\approx \frac{p_{ij,x}}{A}T_z + p_{ij,y}\omega_z, \\ k_{ij,y} &\approx \frac{p_{ij,y}}{A}T_z - p_{ij,x}\omega_z. \end{aligned} \tag{5.8}$$

The estimation of vertical and yaw velocity can be done for each of the eight vectors by solving the set of linear equations as follows

$$\begin{bmatrix} T_{z,ij} \\ \omega_{z,ij} \end{bmatrix} = \begin{bmatrix} \frac{p_{ij,x}}{A} & p_{ij,y} \\ \frac{p_{ij,y}}{A} & -p_{ij,x} \end{bmatrix}^{-1} \begin{bmatrix} k_{ij,x} \\ k_{ij,y} \end{bmatrix}. \tag{5.9}$$

All of the estimations are then averaged, and a final vertical velocity and yaw velocity estimates are then obtained.

[2]We do not consider central vector $\boldsymbol{r}_{22}$, since it is unaffected by rotational and vertical movement in our model.

Chapter 6

Optimizing the parameters

6.1 Optimizing threshold radius of post-processing method

To set the threshold radius for the post-processing algorithm described in 4.3, we performed two experiments on a supplied real-world dataset. We focused mainly on the chief influence of the parameter - the number of measurements averaged in the result. All experiments showed that the method yields the best results with threshold radius set to 1 $\mathrm{m\,s^{-1}}$.

Two trajectories were designed for this experiment. In the first "straight" trajectory, the UAV flew along the global y-axis at a constant speed of 2 $\mathrm{m\,s^{-1}}$ while also rotating along vertical axis at a rate of 0.5 $\mathrm{rad\,s^{-1}}$. The altitude was 4 m.

The second "complex" trajectory was designed to have a shape of the number "8", while the maximum speed was around 4 $\mathrm{m\,s^{-1}}$. The altitude was 4 m. No rotation changes were planned in the trajectory.[1].

Regarding other parameters, the input frame size was 240×240 px and it was divided into nine sections[2]. The maximal number of averaged velocity estimates is thus nine. The real world video stream had 130 frames per second.

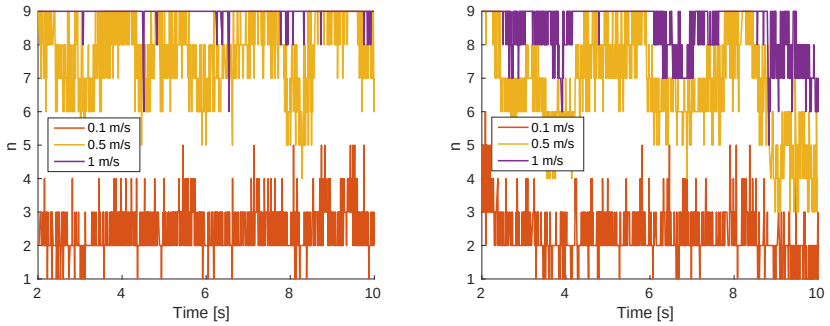
The main influence of the parameter in question is the number of averaged velocity estimates in the final result. When the threshold is set to a very low value, right estimates are split into more groups making the final value a product of chance. With high values of threshold, even wrong results are sorted into the final average. We tested parameter values ranging from

[1]For further reference, we took videos from the experiments. They can be viewed on YouTube. Link to playlist is: `https://www.youtube.com/playlist?list=PLSwHw6pigPZqNijnZfIL8_-otOzRgdQwV`.

[2]This experiment was performed before optimising the number of sections.

0.1 $\mathrm{m\,s^{-1}}$ to 1.5 $\mathrm{m\,s^{-1}}$. Figure 6.1 shows the number of averaged velocity estimates for selected values at relevant parts of the trajectories.

The number of averaged velocity estimates is staying low or fluctuating for low parameter setting. We can observe it with values of 0.1 $\mathrm{m\,s^{-1}}$ or 0.5 $\mathrm{m\,s^{-1}}$ on the figure. We found that this effect starts to vanish from 1 $\mathrm{m\,s^{-1}}$ setting. Therefore we estimated this was presumably the optimal parameter setting.



(a) : "Straight" trajectory **(b) :** "Complex" trajectory

Figure 6.1: Figures showing the number of averaged velocity estimates (maximum is nine) in time for two trajectories for different threshold radius settings.

6.2 Optimizing the number of frame sections for horizontal velocity estimation

As it is further described in chapter 5.1, when estimating the horizontal velocity, the input frame is divided into more rectangular sections. For each one, the optical flow is computed separately, and the results are then combined.

The number of such rectangular sections is a subject for optimisation. A small number of bigger sections leave space for a higher optical flow - the overlap is large even with higher speeds. However, this also results in a smaller number of estimates, thus lowering the potential post-processing method performance. On the other hand, a greater number of smaller sections decreases the maximal measurable optic flow, but it creates more estimates for post-processing method.

To set the optimal number of sections, we performed tests with different settings of this parameter. We used dataset created on a simulator. The input video stream had on average 25 FPS, resolution of 480×480 px and camera had a field of view of 91.5° which corresponds to the lens employed in the final solution.

	Position error [m]			
	Maximal		Average	
Section size/no. of sections	**x**	**y**	**x**	**y**
60 px/64	5.59	2.48	1.43	1.48
80 px/36	2.53	1	0.87	0.5
96 px/25	2.37	0.58	0.85	0.23
120 px/16	**1.96**	0.55	**0.72**	0.20
160 px/9	2.17	**0.49**	0.77	**0.18**
240 px/4	2.12	0.57	0.73	0.22

Table 6.1: Absolute error in position estimation for different number of sections.

The UAV's trajectory was copying the shape of number "8" in altitudes of 1.5 m and 3 m in speeds ranging from 0.5 $\mathrm{m\,s^{-1}}$ to 4 $\mathrm{m\,s^{-1}}$[3].

To select the best number of sections (i.e. the best section size), we integrated the velocity to gain position estimates. We then compared the results to ground truth and calculated the absolute error. The average and maximal errors for a different number of sections are shown in table 6.1.

The results do not show a clear answer. They indicate we should avoid the area lower than 80 px. The final choice of 120 px gives in our opinion an optimal compromise between sufficient number of optical flow measurements for post-processing and maximal measurable velocity.

[3]To gain a better overview, the processed video with 16 sections (120 px) was recorded and uploaded to `https://youtu.be/bFa2c0LzPZ4`

Chapter 7

Simulator verification

We used *Gazebo* simulator to verify all mechanisms of the velocity estimator before trying them in real-world. The behaviour of the UAV in the simulation is very similar to the actual one since it simulates all on-board flight controllers. It was shown that using Gazebo under ROS significantly speeds up the transfer from simulation to real world.

The parameters of the camera used in the simulation were mimicking the real hardware described in chapter 3.2.1. The field of view was set to correspond to the 2.1 mm lens (i.e. in terms of pixels 366.8 px), the output frame rate was 24 Hz at resolution 752×480 pixel. Unfortunately, we were not able to simulate the lens distortion.

7.1 Horizontal velocity estimation

In the first section, we test the performance of horizontal velocity estimation further described in 5.1. First, we verify the function of angular rate correction. Afterwards, we aim to give information about the limits and the accuracy of the solution. In the end, we compare the Phase correlation to parallel-developed Block matching method for optical flow calculation.

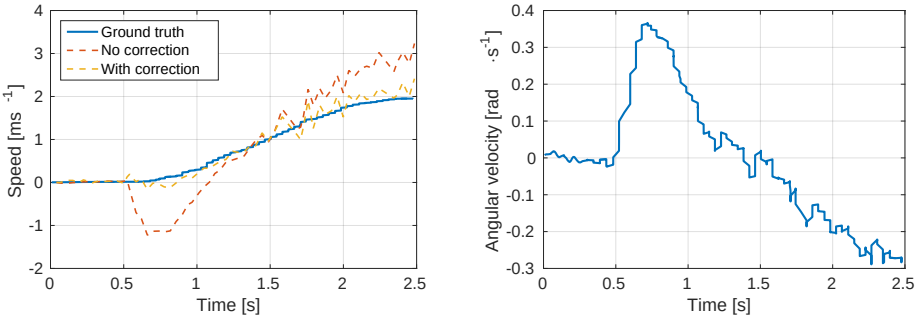
7.1.1 Angular rate correction

Angular rate compensation prevents the estimates to be influenced by rotations in all axes. We divide the mechanism into two parts. The first one is tilt correction, that compensates for rotation in pitch and roll axes. The second part, yaw rotation correction aims to compensate movement in the yaw axis.

Tilt correction

The helicopter moves in pitch and roll axes especially when accelerating. Therefore a trajectory where the UAV accelerates along the x-axis from zero to 2 $\mathrm{m\,s^{-1}}$ in smallest possible time was designed. The compensated and uncompensated velocity estimates where then compared. The result is shown on figure 7.1 along with angular rate in the y-axis.

As we can observe, the corrected estimates are significantly closer to the ground truth than the uncorrected ones.



(a) : Velocity in global x-axis. **(b) :** Angular rate in y-axis.

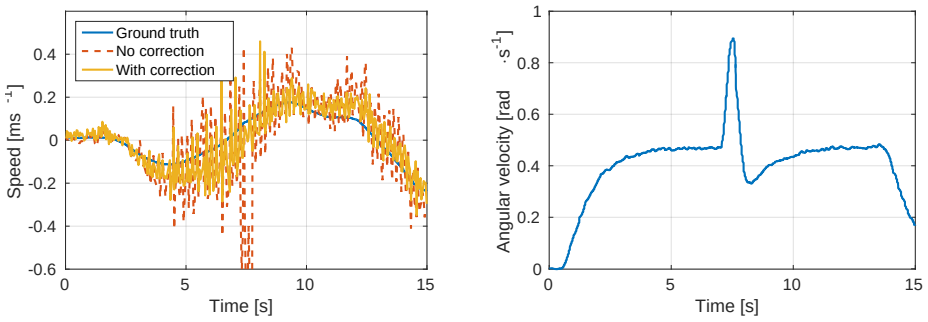
Figure 7.1: Figure showing tilt correction verification on a simulator dataset. On the left-hand side, we show actual velocity in x-axis along with corrected and uncorrected readings from our algorithm. On the right side is shown angular rate in the y-axis.

Yaw rotation correction

When performing rotation along yaw axis, the optical flow calculation method is prone to indicate glitch or biased horizontal movement. Yaw rotation compensation is designed to cope with this problem.

To test the method, we designed a simple trajectory where the UAV performs two full rotations along its z-axis at a rate of 0.5 $\mathrm{rad\,s^{-1}}$ while trying to maintain its position. We then compared compensated and uncompensated estimates of velocity in the x-axis. The result is shown on figure 7.2 along with angular rate about the yaw axis.

It is visible that the estimates without correction contain more noise and even a glitch at circa 7.5 s when angular velocity suddenly changed. Such inaccuracies can lead to degraded performance of position estimation.



(a) : Velocity in global x-axis. **(b) :** Angular rate in yaw axis.

Figure 7.2: Figure showing yaw velocity correction verification on a simulator dataset. On the left-hand side, we show actual velocity in x-axis along with corrected and uncorrected readings from our algorithm. The angular rate about the yaw axis is shown on the right side.

7.1.2 Maximal measurable velocity

Maximal measurable velocity will vary with altitude, frame rate and focal length. To determine the maximum for each combination of variables, we need to specify the maximal measurable optical flow, i.e. the maximal shift in pixels between two frames.

Let s_m be the maximal measurable shift and R the frame rate. Then, if we presume the effect of angular rate to be negligible, we can according to equation (5.2) state the relation between s_m and maximal measurable velocity v_m as

$$v_m = \frac{s_m R A}{f}, \tag{7.1}$$

where and A is the altitude and f is focal length in terms of pixels.

To determine s_m we designed a trajectory where the UAV slowly accelerates from steady state to 4 $\mathrm{m\,s^{-1}}$. We limited the frame rate to $R = 11.5$ Hz, the altitude was $A = 1.5$ m. The resulting velocity estimates are shown in plot 7.3. We observe that the maximal velocity estimates for the setup are around 2 $\mathrm{m\,s^{-1}}$. We can thus conclude the maximal frame shift is:

$$s_m = 42.43 \ \mathrm{px}\,. \tag{7.2}$$

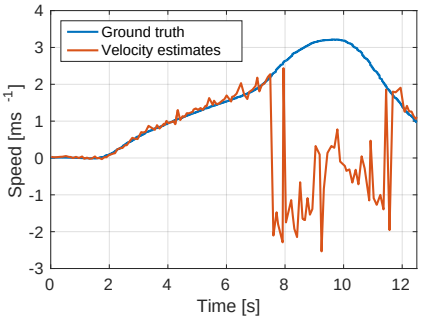
Estimated maximal velocities at different altitudes for different frame rates are shown in table 7.1.

7.1.3 Accuracy testing

To make an accuracy test, we used the same trajectory as in experiment for frame sections optimization described in chapter 6.2. The trajectory has a

$[\mathrm{m\,s^{-1}}]$	**Altitude**			
Frame rate	**1.5 m**	**2 m**	**4 m**	**8 m**
20 Hz	3.48	4.64	9.27	18.55
35 Hz	6.09	8.12	16.23	32.46
50 Hz	8.69	11.59	23.19	46.37
80 Hz	13.91	18.55	37.10	74.19
100 Hz	17.39	23.19	46.37	92.74

Table 7.1: Maximal velocities based on simulation results for different frame rates and altitudes. The table assumes a setup with Bluefox camera with 2.1 mm lens.



(a) : Velocity in global x-axis.

Figure 7.3: Maximal velocity test for Phase correlation on a simulator dataset. Frame rate was limited to $R = 11.5$Hz to reach maximal value easily.

shape of a figure "8" that is followed twice in altitude of 1.5 m and thrice in altitude of 3 m. The shape of the trajectory can be seen on figure 7.4. The velocity ranges from 1 $\mathrm{m\,s^{-1}}$ to 4 $\mathrm{m\,s^{-1}}$ in global x-axis and from 0.5 $\mathrm{m\,s^{-1}}$ to 2 $\mathrm{m\,s^{-1}}$ in y-axis. The total length of the trajectory is about 130 m.

Processed video was captured for the trajectory. An example snapshot can be found on figure 7.5 along with hyperlink to the video.

We integrated the velocity estimates to create position estimates and compared them with ground truth. We then calculated the maximal and average velocity and position errors as shown in Table 7.2. Position estimation comparison is shown on figure 7.6.

As the table shows, velocity estimates have on average an error of no more than 0.1 $\mathrm{m\,s^{-1}}$ but sometimes can contain outliers. As we see on the image, position estimation error in the x-axis is rising with time as expected up to 2 m at the end. Both average and maximal errors in y-axis are smaller, which is caused by the fact that a shorter distance was flown in this axis.

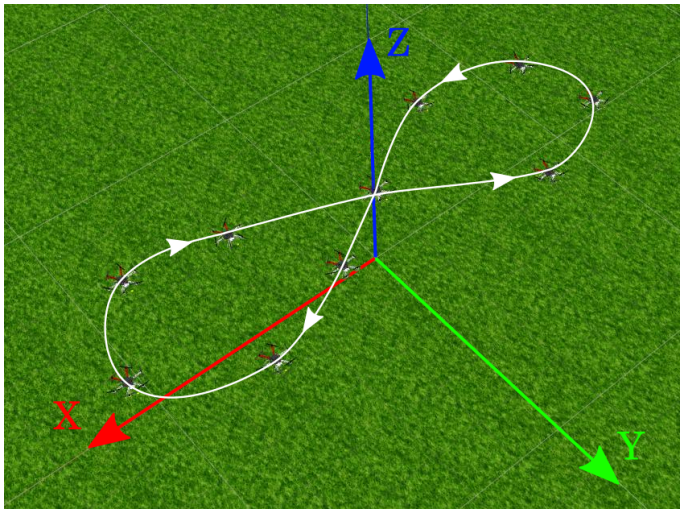


Figure 7.4: The shape of the trajectory used for accuracy testing on a simulator.

Velocity error [$\mathrm{m\,s^{-1}}$]				**Position error** [m]			
Maximal		**Average**		**Maximal**		**Average**	
x	**y**	**x**	**y**	**x**	**y**	**x**	**y**
1.9	2.2	0.096	0.063	1.96	0.55	0.72	0.20

Table 7.2: Maximal and average velocity and position estimates errors for horizontal velocity measurements based on results from simulator dataset.

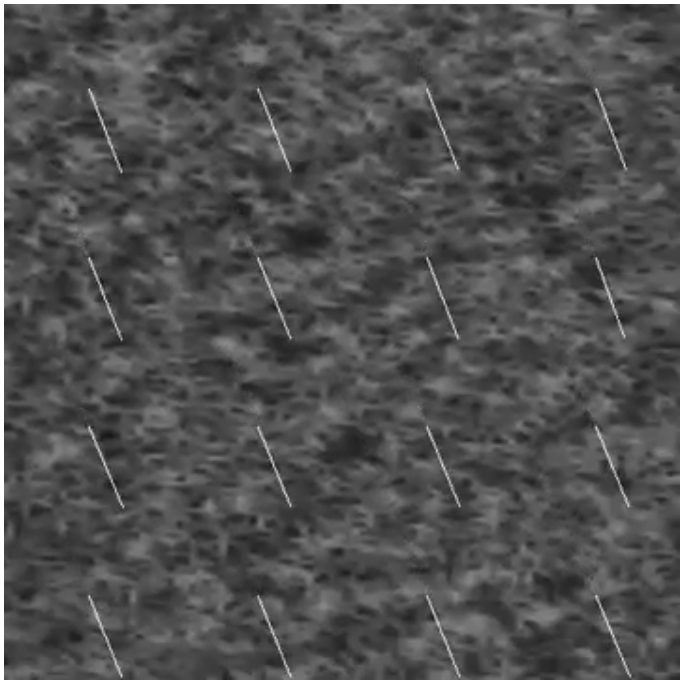
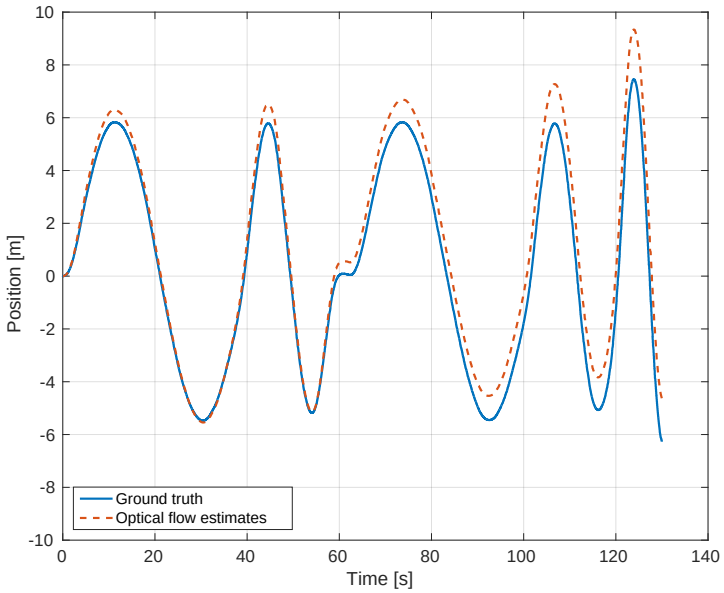
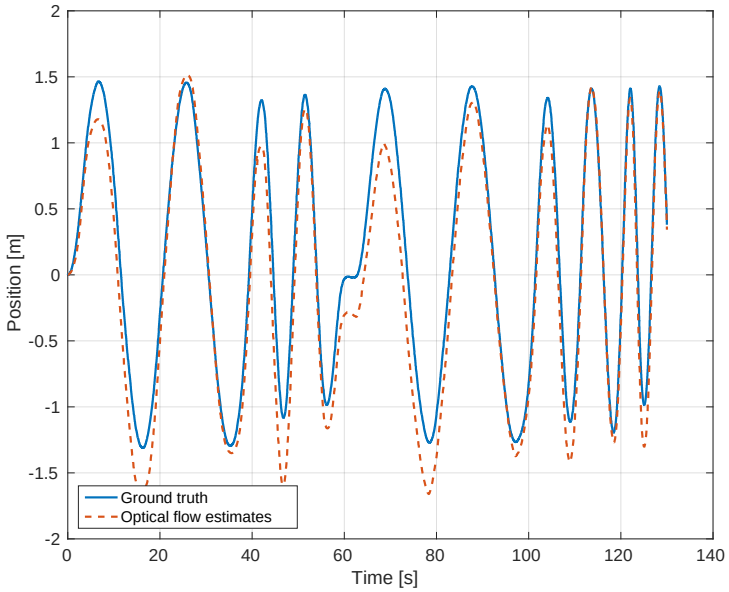


Figure 7.5: Example picture from processed video from horizontal velocity estimation testing in simulator. White vectors show estimated optical flow for each section during flight. The video can be found on `https://youtu.be/bFa2c0LzPZ4`.



(a) : Position in global x-axis.



(b) : Position in global y-axis.

Figure 7.6: Figure showing position estimation for Phase correlation with a simulator dataset. The UAV was following a trajectory shaped as a figure "8". The velocities ranged from 0.5 $\mathrm{m\,s^{-1}}$ to 4 $\mathrm{m\,s^{-1}}$. The first 60 s the altitude was 1.5 m and it changed to 3 m afterwards. The total length was about 130 m.

7.1.4 Comparison with Block Matching algorithm

A comparison with Block Matching method for optical flow calculation developed in parallel to this work was done. The method is described in chapter 4.2.1. The section size was set to 24 px, the distance between sections to 24 px and the scan radius was 21 px. At first, we performed experiments to find constraints of the method. Accuracy tests taking into account the limits were done afterwards.

Maximal measurable velocity

We used the same methodology as in section 7.1.2 to find the maximal measurable optical flow s_m. The UAV was again slowly accelerating in the global x-axis to 4 $\mathrm{m\,s^{-1}}$. We limited the frame rate to $R = 14.1$ Hz, the altitude was $A = 1.5$ m. The resulting velocity estimates are shown in figure 7.7.

The maximum was reached at a speed of around 1.2 $\mathrm{m\,s^{-1}}$. We can thus conclude according to equation (7.1) that maximal measurable optical flow is:

$$s_m = 20.7 \text{ px}\,. \tag{7.3}$$

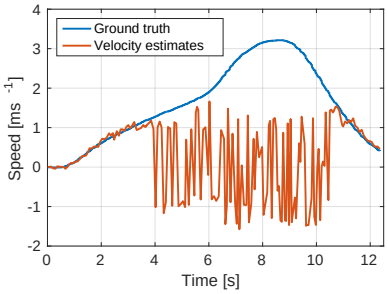
We can create a table 7.3 of maximal velocities for Block matching method based on the value. The table shows that the maximal velocities are slightly lower than the limits of Phase correlation. It is especially significant for the frame rates of 20 Hz and 40 Hz, where they are partly below the velocities that are expected to be employed in the usecase for the optical flow localization. But as long as the frame rate stays around 60 Hz and above, we would not expect any problems.

$[\mathrm{m\,s^{-1}}]$	**Altitude**			
Frame rate	**1.5 m**	**2 m**	**4 m**	**8 m**
20 Hz	1.70	2.26	4.52	9.05
35 Hz	2.97	3.96	7.92	15.84
50 Hz	4.24	5.66	11.31	22.62
80 Hz	6.79	9.05	18.10	36.20
100 Hz	8.48	11.31	22.62	45.25

Table 7.3: Maximal velocities with Block Matching method at different altitudes and for different frame rates based on the results from simulator. The table assumes a setup with Bluefox camera with 2.1 mm lens.

Accuracy comparison

To preform an accuracy comparison we created a trajectory that respected the findings of maximal velocities. The UAV copied the shape of figure "8"



(a) : Velocity in x-axis.

Figure 7.7: Maximal velocity test for Block matching on a simulator dataset. Frame rate was limited to $R = 14.1$ Hz to reach maximal value easily.

two times in speeds ranging from 1 $\mathrm{m\,s^{-1}}$ to 2 $\mathrm{m\,s^{-1}}$ at an altitude of 3 m. We simulated the usual Bluefox camera setup with 2.1 mm lens, the frame rate was 24 Hz. The total length of the trajectory was around 48 m.

The velocity estimates were integrated to create position estimates. The outputs from the two methods were then compared to ground truth. The absolute is error shown in table 7.4. The comparison of position estimates is displayed on figure 7.8.

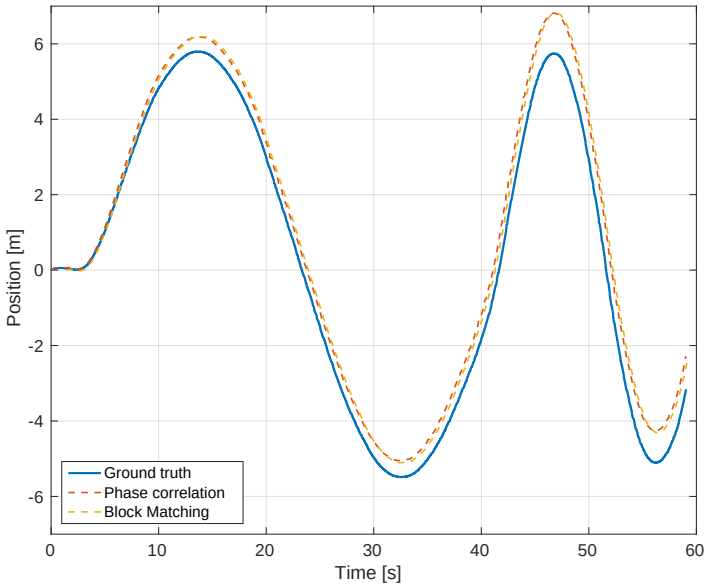
We observe from the table that maximal error is very similar for the two methods in both position and velocity estimates. Regarding the velocity estimates, Phase correlation is slightly more accurate on average. The position error shows different results in each axis. In the x-axis, in which more portion of total length was absolved, Phase correlation is better.

The figures show that estimates of position in x-axis are very similar. In y-axis, the difference between the two method is more significant, but does not go over 0.5 m.

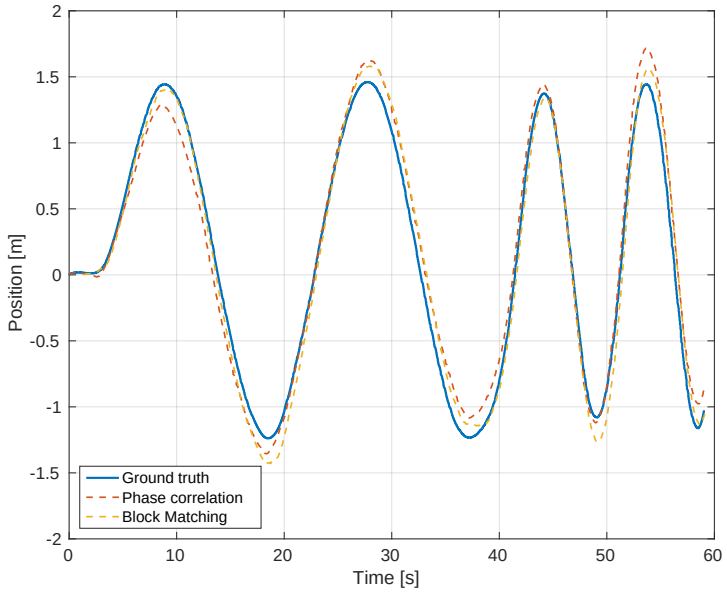
Overall, we can state the two methods perform very similarly regarding the accuracy. The decision between the two should therefore be based more on the available frame rate (closely connected with the required processing power) and required maximal velocity.

	Velocity error [$\mathrm{m\,s^{-1}}$]				**Position error** [m]			
	Maximal		**Average**		**Maximal**		**Average**	
	x	**y**	**x**	**y**	**x**	**y**	**x**	**y**
Phase corr.	0.85	0.49	0.077	0.058	1.10	0.31	0.538	0.129
Block Mat.	0.88	0.40	0.092	0.071	1.14	0.20	0.581	0.092

Table 7.4: Comparison of accuracies of Block matching and Phase correlation based on simulator measurements. The testing trajectory was 48 m long.



(a) : Position in global x-axis.



(b) : Position in global y-axis.

Figure 7.8: Accuracy comparison between Block Matching and Phase correlation on a simulator dataset. The UAV was following a trajectory shaped as a figure "8". The velocities ranged from 1 $\mathrm{m\,s^{-1}}$ to 2 $\mathrm{m\,s^{-1}}$ at an altitude of 3 m. The total length was about 48 m.

7.2 3D translational velocity and yaw rate estimation

In this section, we test the accuracy of the method for the translational 3D-velocity and yaw rate estimation described in chapter 5.2. Firstly, we focus on the translational velocity estimation. Then we test the yaw rate estimation.

7.2.1 Translational velocity estimation testing

In this chapter, we test horizontal velocity estimation as well as vertical velocity measurement. Horizontal velocity is already being tested in the section above, but since this method uses slightly different approach to the measurement, we test it here too.

We prepared four trajectories for the experiment. The first, "flat", was designed to specifically test accuracy of horizontal velocity estimation. It is the same trajectory as e.g. in experiment in section 7.1.3. It copies the shape of figure "8" (see figure 7.4) twice at altitude of 1.5 m and thrice at 3 m in horizontal speeds varying from 0.5 $\mathrm{m\,s^{-1}}$ to 4 $\mathrm{m\,s^{-1}}$. The length of the trajectory is around 130 m.

The other trajectories tested accuracy of vertical speed estimation and how it is affected by horizontal movement. They are illustrated on figure 7.10. In the second "vertical" trajectory, the UAV was ascending at a rate of 1 $\mathrm{m\,s^{-1}}$. In the third and fourth, vertical and horizontal speeds were equal. In third trajectory the speeds were 1 $\mathrm{m\,s^{-1}}$ and in the fourth 2 $\mathrm{m\,s^{-1}}$. Processed video was captured for each of these trajectories. An example snapshot can be found on figure 7.9 along with a hyperlink to the video.

We integrated the velocity estimates and created position estimates. We then compared the outputs to ground truth, calculated the absolute error shown in tables 7.5 and 7.6. Vertical velocity estimates from last three trajectories are also shown in figure 7.11.

First, let us compare the horizontal velocity estimation accuracy. In comparison with tests done in section 7.1.3 we see that the average error is comparable when measuring velocity. However, the integrated position is burdened with higher error when using the 3D method.

Vertical velocity average error does not seem to be very affected by horizontal movement up to the vertical speeds of 2 $\mathrm{m\,s^{-1}}$. It is around 0.06 $\mathrm{m\,s^{-1}}$ for all trajectories with lower vertical speed. Figures show the estimates contain noise that seems to be directly proportional to the velocity. Average and maximal position errors get higher with longer experiments, because of the integration.

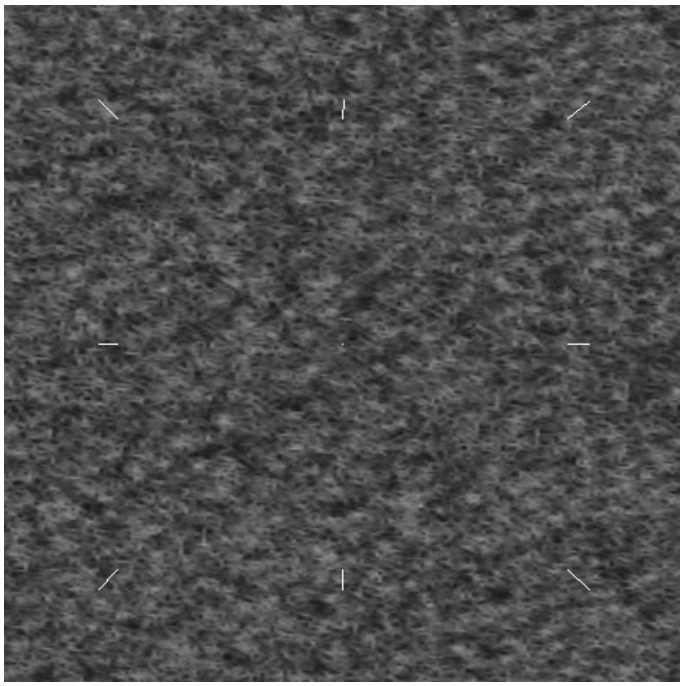


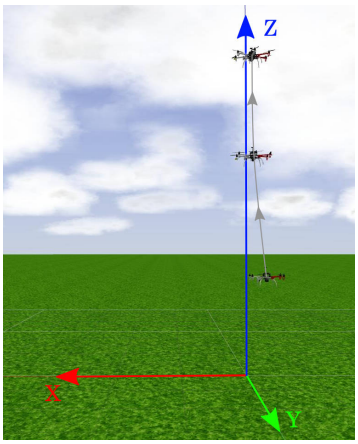
Figure 7.9: Example picture from processed video from vertical velocity estimation testing in simulator. White vectors show estimated optical flow for each section. As we see on them, the image was taken while the UAV was ascending. The videos can be found on `https://www.youtube.com/playlist?list=PLSwHw6pigPZqsUZsWBD0J5-BBvexpk_A9`.

	Velocity error [$\mathrm{m\,s^{-1}}$]					
	Maximal			**Average**		
Trajectory	**x**	**y**	**z**	**x**	**y**	**z**
Flat	1.91	0.77	1.45	0.095	0.060	0.051
Vertical	-	-	0.70	-	-	0.067
1 $\mathrm{m\,s^{-1}}$	0.33	0.11	0.28	0.061	0.028	0.054
2 $\mathrm{m\,s^{-1}}$	0.54	0.14	0.50	0.11	0.041	0.11

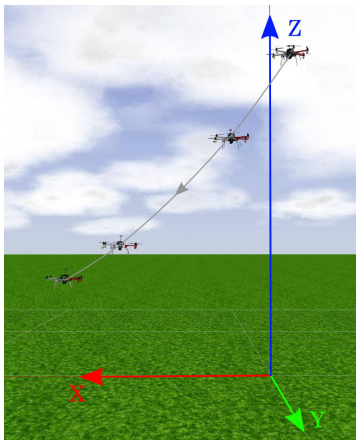
Table 7.5: Velocity estimates errors for different trajectories for translational 3D-velocity testing on the simulator dataset.

	Position error [m]					
	Maximal			**Average**		
Trajectory	**x**	**y**	**z**	**x**	**y**	**z**
Flat	2.5	0.43	1.3	0.87	0.13	0.25
Vertical	-	-	0.074	-	-	0.023
1 $\mathrm{m\,s^{-1}}$	0.31	0.13	0.097	0.15	0.062	0.029
2 $\mathrm{m\,s^{-1}}$	0.096	0.046	0.074	0.038	0.012	0.023

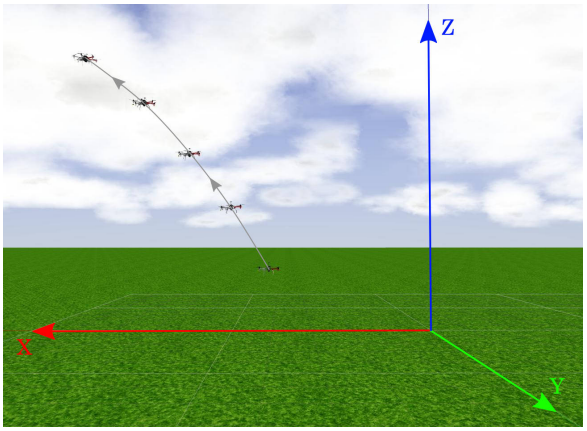
Table 7.6: Position estimates errors for different trajectories for translational 3D-velocity testing on the simulator dataset.



(a) : No horizontal movement, vertical speed 1 $\mathrm{m\,s^{-1}}$.

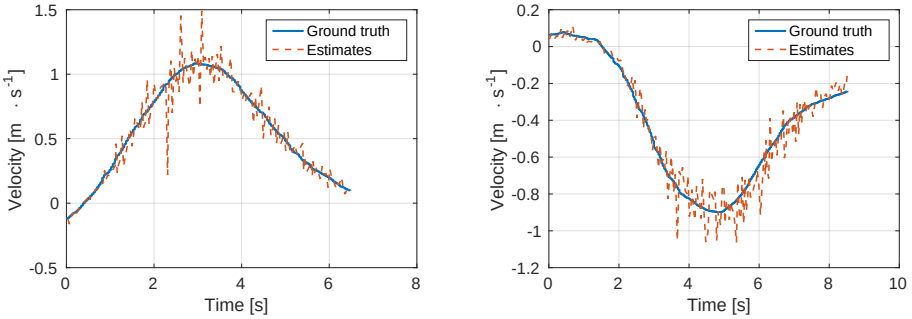


(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$, vertical 1 $\mathrm{m\,s^{-1}}$.



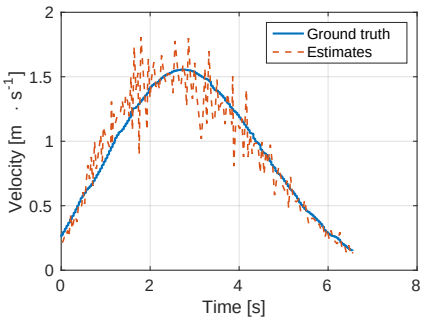
(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$, vertical 2 $\mathrm{m\,s^{-1}}$.

Figure 7.10: Trajectories used for vertical rate estimation testing.



(a) : No horizontal movement

(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$



(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$

Figure 7.11: Vertical rate estimation at different horizontal speeds along global x-axis with a simulator dataset.

7.2.2 Yaw velocity estimation testing

We tested accuracy of yaw rate estimation on three different trajectories. In all of them the UAV was rotating at a rate of 0.5 $\mathrm{rad\,s^{-1}}$, i.e. the maximal safe speed. The difference was in the horizontal movement. In the first, none was planned. In the rest, movement along global x-axis at a speed of 1 $\mathrm{m\,s^{-1}}$ and 2 $\mathrm{rad\,s^{-1}}$ was designed. This was done to show the effect of horizontal movement on accuracy of the measurement. The last two trajectories are illustrated on figure 7.13. The setup stays the same as in other experiments, simulated Bluefox camera with 2.1 mm lens, frame rate of 24 Hz.

Processed video was captured for each trajectory. An example snapshot can be found on figure 7.12 along with a hyperlink to the video.

As in other experiments, we integrated the velocity estimates, calculated error from ground truth and put maximal and average values into table 7.7. The yaw velocities are also compared to ground truth on figure 7.14. As we can see, the horizontal movement does not seem to have any effect on the accuracy. The maximal velocity error with 1 $\mathrm{m\,s^{-1}}$ shows that outliers sometimes get into the results. But overall, the average yaw rate error is not more than 0.035 $\mathrm{rad\,s^{-1}}$.

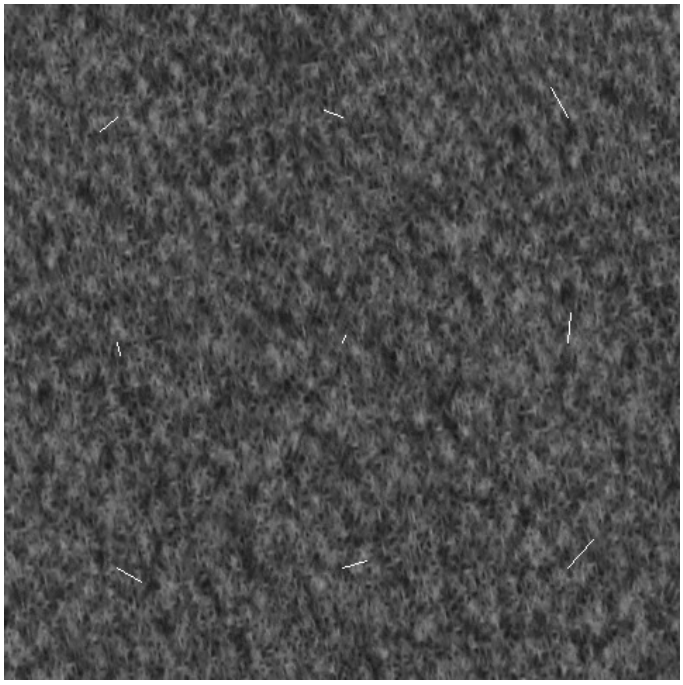
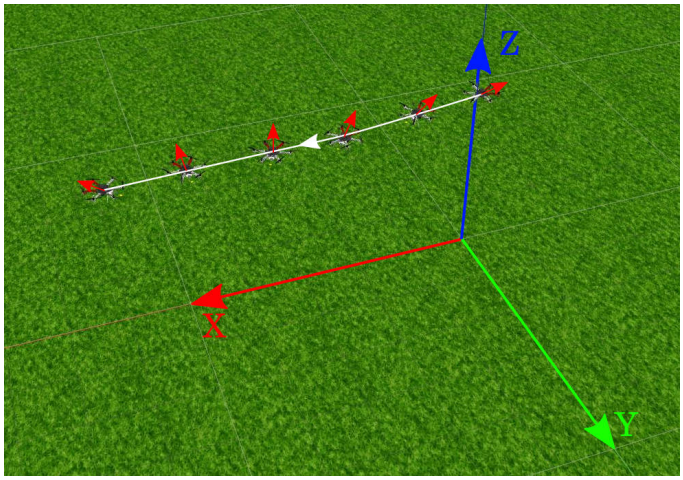
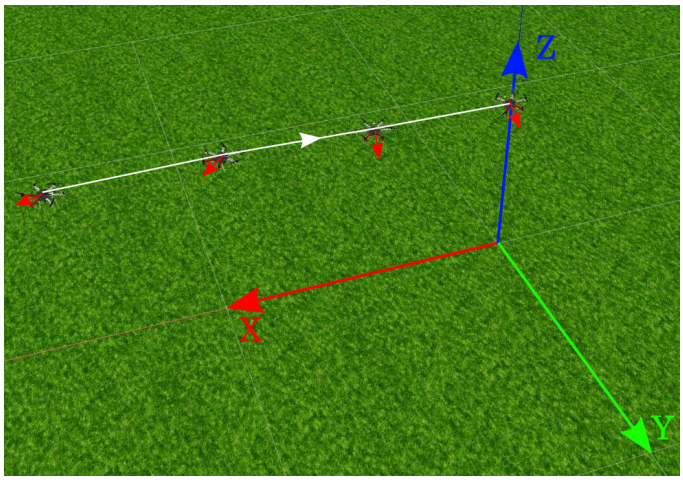


Figure 7.12: Example picture from processed video from yaw velocity estimation testing in simulator. White vectors show estimated optical flow for each section. As we see on them, the image was taken while the UAV was rotating. The videos can be found on `https://www.youtube.com/playlist?list=PLSwHw6pigPZoPw90UYATG34SJXWWQMQw8`.



(a) : Yaw rate 0.5 $\mathrm{rad\,s^{-1}}$, horizontal speed 1 $\mathrm{m\,s^{-1}}$.

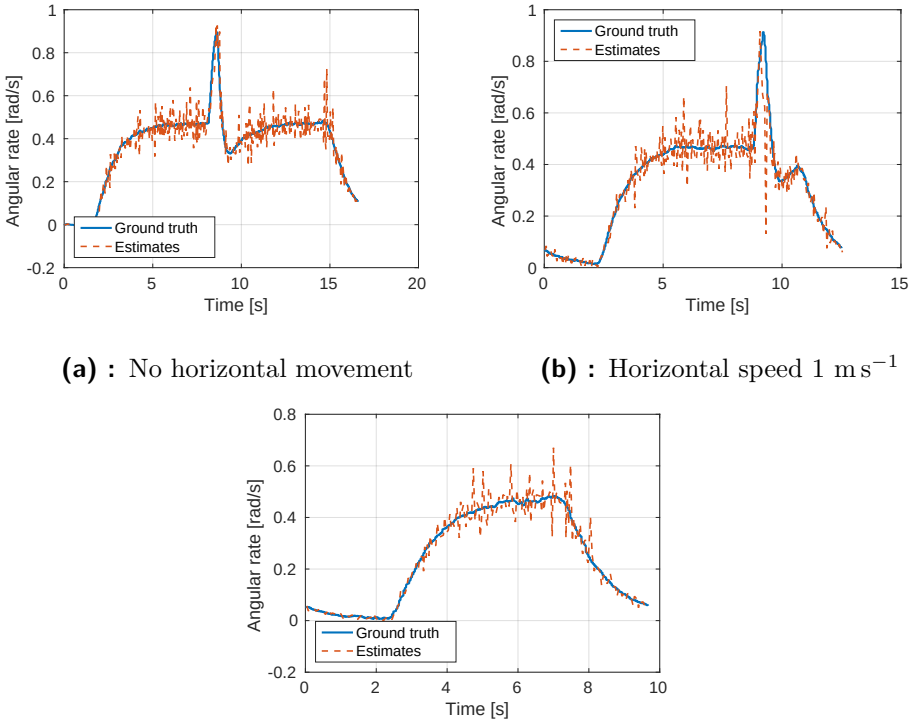


(b) : Yaw rate 0.5 $\mathrm{rad\,s^{-1}}$, horizontal speed 2 $\mathrm{m\,s^{-1}}$.

Figure 7.13: Trajectories used for yaw rate estimation testing. The red arrow shows the UAV orientation.

	Velocity error [$\mathrm{rad\,s^{-1}}$]		**Position error** [rad]	
Horiz. speed	**Maximal**	**Average**	**Maximal**	**Average**
0	0.26	0.035	0.10	0.039
1 $\mathrm{m\,s^{-1}}$	0.66	0.035	0.13	0.044
2 $\mathrm{m\,s^{-1}}$	0.20	0.024	0.09	0.028

Table 7.7: Yaw rate estimations errors at different horizontal speeds.



(a) : No horizontal movement

(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$

(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$

Figure 7.14: Yaw rate estimation at different horizontal speeds along global x-axis with a simulator dataset.

Chapter 8

Real-world verification

In this chapter, we present a verification of our method on a real-world dataset as well as its comparison to Block matching method and *PX4FLOW Smart Camera* sensor. First, we verify the horizontal velocity estimation. We then focus on the proposed 3D-translational speed and yaw rotation estimation method.

The datasets employed in real-world verification have the same parameters as those in simulator verification. The UAV was commanded with the same trajectories and a raw video stream was captured from the Bluefox camera equipped with 2.1 mm lens (i.e. in terms of pixels 366.8 px). The only difference was in the frame rate, which was set to 35 Hz. Internal position estimation system described in 3.2.4 was used to provide the ground truth data.

All of the testing was done in an outside environment. The ground texture is grass since the experiments were done on a greenfield side.

8.1 Horizontal velocity estimation

This section describes verification of our horizontal velocity estimation method described in section 5.1 and simultaneously provides a comparison with Block matching method and *PX4FLOW Smart Camera*. To carry out such a comparison, a special test setup was done on the UAV. The *PX4FLOW* sensor was mounted next to the Bluefox camera as figure 8.1 shows.

8.1.1 Maximal measurable velocity

Following the same methods as in sections 7.1.2 and 7.1.4, we first determine the maximal measurable optical flow for Phase correlation and Block matching

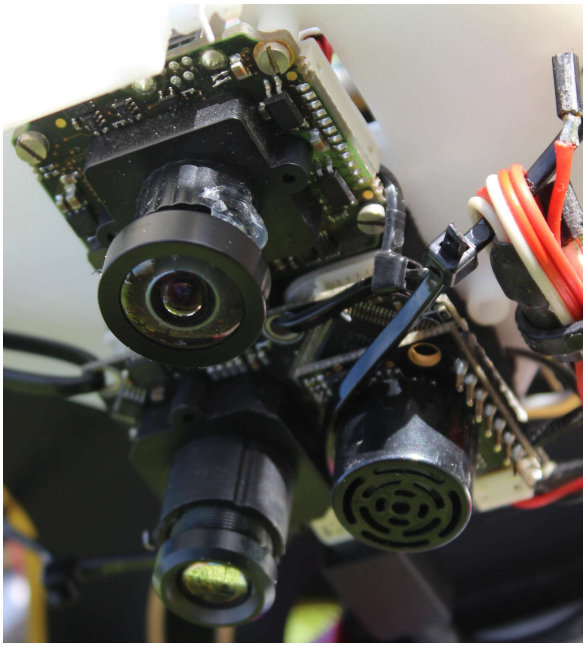


Figure 8.1: Figure showing the test setup for comparison with *PX4FLOW Smart Camera* sensor. The Bluefox camera is on the top part of the image, PX4FLOW sensor on the bottom. Both sensors are mounted on the bottom of the UAV facing the ground.

methods to create a table of velocity constraints. We then present measured maxima for *PX4FLOW* sensor for comparison.

Similarly to the sections referred above, the UAV was slowly accelerating in the test trajectory from the steady state up to 4 $\mathrm{m\,s^{-1}}$ in an altitude of 1.5 m. The frame rate was limited to 17.5 Hz. Block matching method was using the same parameters as in section 7.1.4. The measured speeds are shown for both methods on figure 8.2. We observe that maximal stably measured velocity is 1.95 $\mathrm{m\,s^{-1}}$ for Phase correlation and 1.5 $\mathrm{m\,s^{-1}}$ for block matching. This corresponds to maximal measurable frame shift for Phase correlation according to equation (7.1)

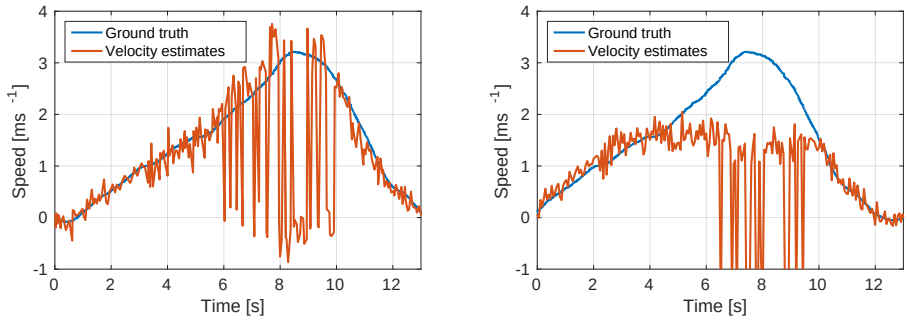
$$s_{m,PC} = 27.2 \ \mathrm{px} \tag{8.1}$$

and for Block matching:

$$s_{m,BM} = 20.3 \ \mathrm{px}\,. \tag{8.2}$$

The value for Block matching is basically the same as the results from simulation. However, the performance of Phase correlation has dropped significantly. This may be caused by lens distortion and more noise present in the image.

Based on these values we can calculate the maximal measurable velocities following the equation (7.1). The results are shown for each optical flow calculation method in tables 8.1 and 8.2.



(a) : Estimated speed with Phase correlation.

(b) : Estimated speed with Block matching.

Figure 8.2: Figure showing maximal speed testing for Phase correlation and Block matching methods on a real-world dataset. The UAV was accelerating in the global x-axis at an altitude of 1.5 m. The frame rate was limited for the test to 17.5 Hz.

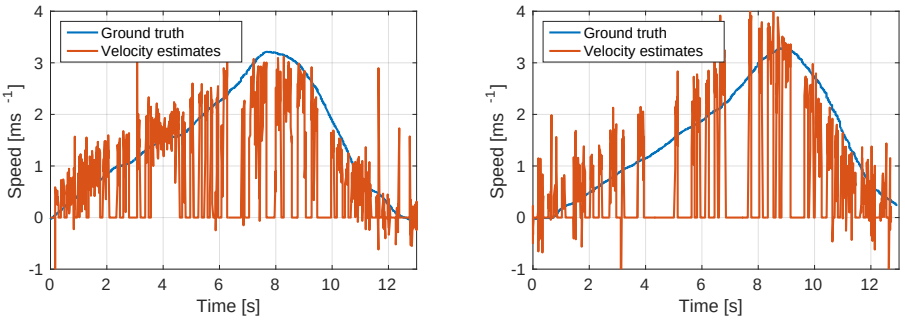
$[\mathrm{m\,s^{-1}}]$	**Altitude**			
Frame rate	**1.5 m**	**2 m**	**4 m**	**8 m**
20 Hz	2.23	2.97	5.94	11.87
35 Hz	3.90	5.19	10.39	20.77
50 Hz	5.56	7.42	14.84	29.68
80 Hz	8.90	11.87	23.74	47.48
100 Hz	11.13	14.84	29.68	59.36

Table 8.1: Maximal velocities for Phase correlation based on real-world experiments for different frame rates and altitudes. The table assumes a setup with Bluefox camera with 2.1 mm lens.

$[\mathrm{m\,s^{-1}}]$	**Altitude**			
Frame rate	**1.5 m**	**2 m**	**4 m**	**8 m**
20 Hz	1.66	2.21	4.43	8.86
35 Hz	2.91	3.88	7.75	15.50
50 Hz	4.15	5.54	11.07	22.15
80 Hz	6.64	8.86	17.72	35.44
100 Hz	8.31	11.07	22.15	44.30

Table 8.2: Maximal velocities for Block matching based on real-world experiments for different frame rates and altitudes. The table assumes a setup with Bluefox camera with 2.1 mm lens.

To test the maximal velocity of *PX4FLOW* sensor, we performed the same test flight with UAV accelerating from steady state to 4 $\mathrm{m\,s^{-1}}$ at altitudes of 1.5 m, 2 m, 4 m and 8 m. The sensor provided results only for the first two altitudes, at the other two the readings remained at 0 possibly because of an exceeded range of the sonar installed on the sensor. The estimates in the first two altitudes are shown compared to ground truth in figure 8.3.



(a) : *PX4FLOW* sensor output at 1.5 m altitude.

(b) : *PX4FLOW* sensor output at 2 m altitude.

Figure 8.3: Figure showing maximal speed testing for *PX4FLOW* sensor. The UAV was accelerating in the global x-axis at an altitude of 1.5 m and 2 m.

We see in the figures that we were not able to reach the maximal velocity. However, the estimates often fall to zero, especially in the higher altitude. This is probably caused by instability of the sonar measurements. The results are concluded in table 8.3.

$[\mathrm{m\,s^{-1}}]$	**Altitude**			
Frame rate	**1.5 m**	**2 m**	**4 m**	**8 m**
250 Hz	>4	>4 (unstable)	0	0

Table 8.3: Maximal velocities for *PX4FLOW* sensor based on real-world experiments in different altitudes. The sensor was not able to measure velocity in 4 m and above. Results in lower altitudes were unstable as shown in figures.

8.1.2 Accuracy testing

To perform accuracy tests we created a trajectory where the UAV copied the shape of figure "8" (see figure 7.4) two times at an altitude of 1.5 m and three times at 3 m in speeds ranging from 1 $\mathrm{m\,s^{-1}}$ to 4 $\mathrm{m\,s^{-1}}$. However, the results from *PX4FLOW* sensor were highly inaccurate at the higher altitude. We thus present only the results from the flight in the lower altitude, where the maximal velocity was limited to 2 $\mathrm{m\,s^{-1}}$. The total length was about 56 m. Other parameters were set the same as in other experiments (see the beginning of the chapter). For further reference, we created a video on showing the processed dataset, see figure 8.4.

The velocity estimates were integrated to obtain position estimates. The *PX4FLOW* sensor provided many zero measurements. These were ignored since that provided better results than including them. All of the readings were compared to ground truth and average and maximal absolute error was computed. The results are shown in table 8.4. The velocity estimates are shown in figure 8.5, position in figure 8.6.

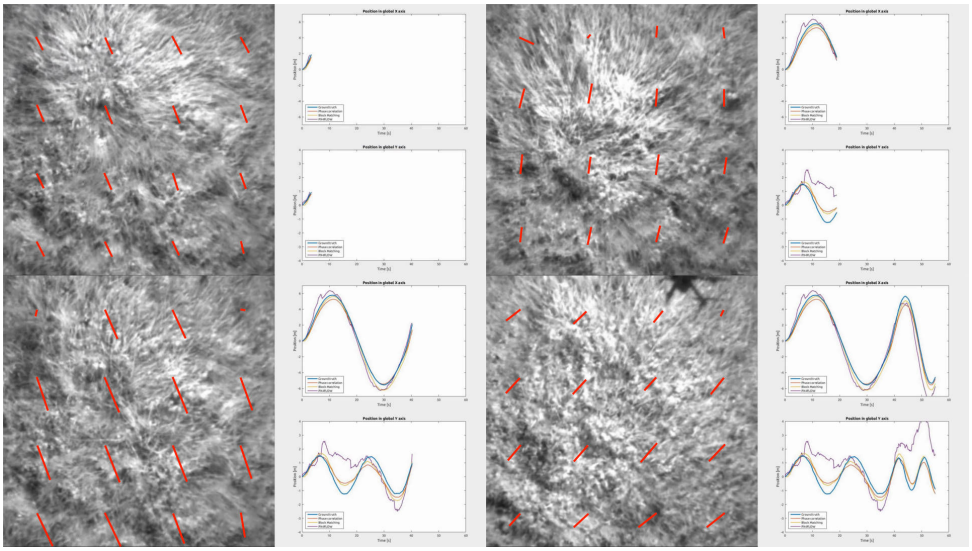


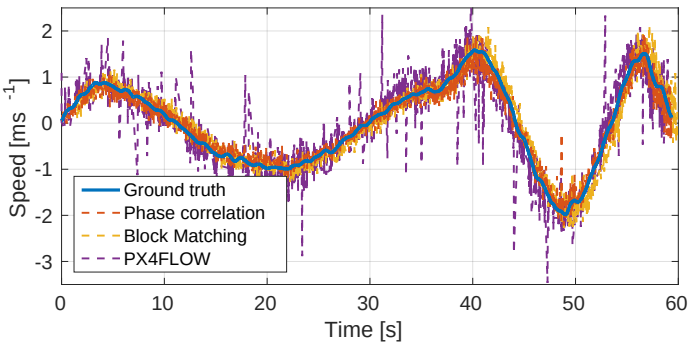
Figure 8.4: Snapshots of a video compiling the accuracy testing. Highlighted vectors in the image show measured optical flow. The video can be viewed on `https://youtu.be/u3pxrFuNzs8`.

The table and also figures show that the average errors in both velocity and position estimation are very similar between Phase correlation and Block matching. Furthermore, the values do not differ significantly between x- and y-axis. An occasional outlying result is given by both methods, but this does not happen very frequently. The average velocity error of both approaches is around 0.1 $\mathrm{m\,s^{-1}}$. The endpoint error is again almost the same for both methods. After the 56 m distance, the position estimation error was around 1 m. Compared to the simulator, the errors are slightly higher, but this has to be expected.

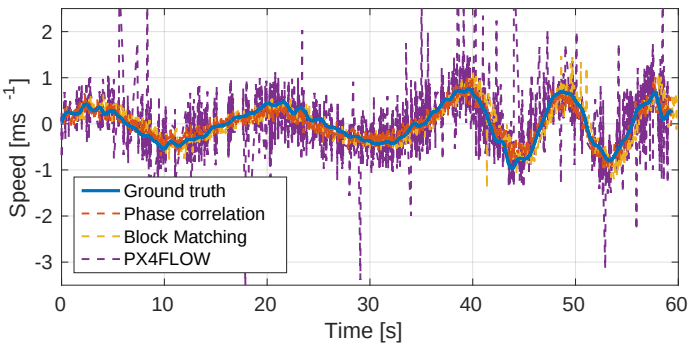
Looking at *PX4FLOW* results in figure 8.5, we see that they contain very much noise. The average position and velocity errors considering only x-axis are about twice as big as with the other two methods. However, the performance of the sensor in the y-axis is very poor. The sensor seems to fail in providing a velocity measurement in y-axis in significant parts of the trajectory thus deviating the position estimates significantly. Possible explanation for the effect might be that the velocity in x-axis was high so that the sensor could not measure the velocity in the other axis accurately. However, the real cause of the problem is unknown to the author and to our team.

	Velocity error [m s^{-1}]				Position error [m]			
	Maximal		Average		Maximal		Average	
	x	y	x	y	x	y	x	y
Phase corr.	1.7	0.57	0.13	0.11	1.78	0.78	0.36	0.31
Block mat.	0.90	1.7	0.12	0.11	0.62	0.85	0.25	0.36
PX4FLOW	2.51	4.4	0.25	0.38	2.03	3.72	0.67	1.40

Table 8.4: Maximal and average velocity and position estimates errors for more methods based on the real-world dataset.

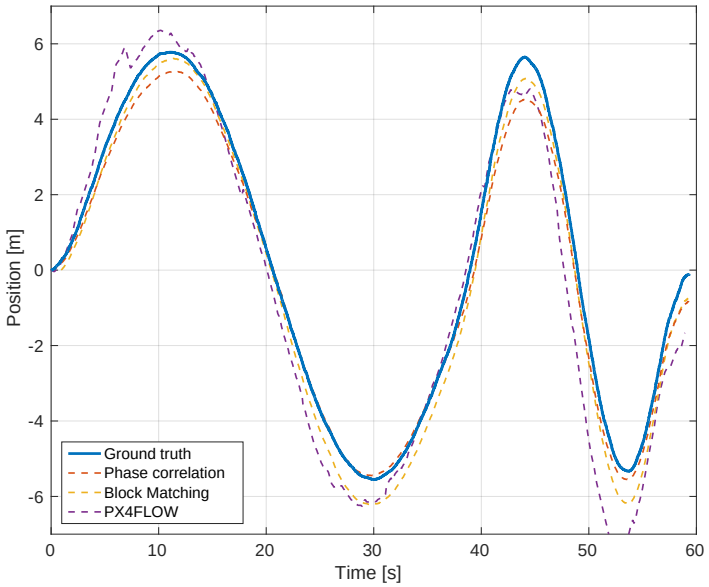


(a) : Velocity in global x-axis.

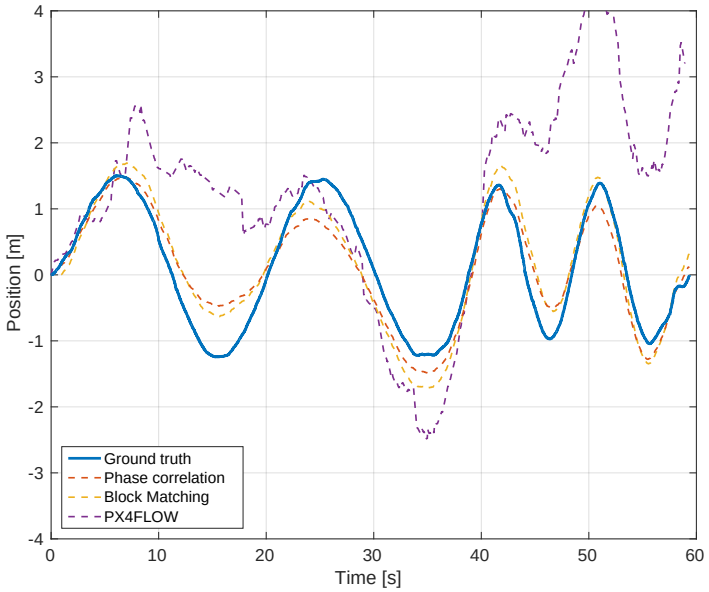


(b) : Velocity in global y-axis.

Figure 8.5: Velocity estimates obtained by Block Matching, Phase correlation and *PX4FLOW* sensor from a real-world dataset. The UAV was following a trajectory shaped as a figure "8" at an altitude of 1.5 m. The total length was 56 m.



(a) : Position in global x-axis.



(b) : Position in global y-axis.

Figure 8.6: Position estimates obtained by Block Matching, Phase correlation and *PX4FLOW* sensor from a real-world dataset. The UAV was following a trajectory shaped as a figure "8" at an altitude of 1.5 m. The total length was 56 m.

8.2 3D translational velocity and yaw rate estimation

In this section, we test the accuracy of the 3D translational velocity and yaw rate estimation method proposed in section 5.2 on a real world dataset. The tests were done in a way similar to tests on the simulator in section 7.2. First, we focus on translational velocity, and then we verify yaw velocity.

8.2.1 Translational velocity estimation testing

In this chapter, we examine the accuracy of horizontal and vertical velocity estimation. The horizontal velocity accuracy is already being tested above, but since this method estimates the horizontal velocity with a slightly different method and parameters, we test it here as well.

We created four trajectories for the test. The first, "flat" trajectory was the same as in section 8.1.2. It was used to test the accuracy of horizontal velocity estimation. The UAV copied the shape of figure "8" (see figure 7.4) two times at an altitude of 1.5 m at speeds ranging from 0.5 $\mathrm{m\,s^{-1}}$ to 2 $\mathrm{m\,s^{-1}}$. The length of the trajectory was around 56 m.

The other three trajectories were the same as in section 7.2.1. They tested the accuracy of vertical speed estimation and how it is affected by horizontal movement. They are illustrated on figure 7.10. In the second "vertical" trajectory, the UAV was ascending at a rate of 1 $\mathrm{m\,s^{-1}}$. In the third and fourth, vertical and horizontal speeds were equal. In third trajectory the speeds were 1 $\mathrm{m\,s^{-1}}$ and in the fourth 2 $\mathrm{m\,s^{-1}}$. The results from these trajectories are compiled in a video, see figure 8.7.

Again, velocity estimates were integrated to obtain position estimates, and after a comparison with ground truth, an absolute error was computed for both. The tables 8.5 and 8.6 show maximal and average errors for both position and velocity. Figure 8.8 shows the vertical velocity estimates for the last three trajectories.

As we observe on the figures, the vertical velocity with no horizontal movement was estimated without much noise. However, the estimates seem to saturate at around 0.5 $\mathrm{m\,s^{-1}}$. This might be caused by a short *TerraRanger* sensor failure. We see more noise on the tests with horizontal movement, which is to be anticipated. Compared to the simulator results, the estimates are again noisier.

Considering the tables, the average errors in vertical velocity and position estimation are very similar for all of the trajectories. They are one order of magnitude higher than the results from the simulator. We could state that the average velocity error is about 0.2 $\mathrm{m\,s^{-1}}$.

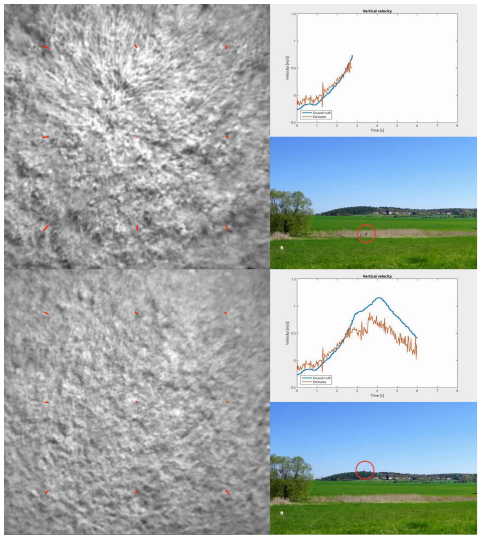
Observing the errors of horizontal velocity estimates, we see that they are significantly higher than the errors of methods tested in the previous section. We can thus recommend the tested method only for vertical velocity estimation.

	Velocity error [$\mathrm{m\,s^{-1}}$]					
	Maximal			**Average**		
Trajectory	**x**	**y**	**z**	**x**	**y**	**z**
Flat	1.7	1.31	1.21	0.41	0.30	0.19
Vertical	-	-	0.57	-	-	0.20
1 m/s	1.3	0.95	0.85	0.35	0.21	0.22
2 m/s	1.7	1.3	1.2	0.41	0.30	0.18

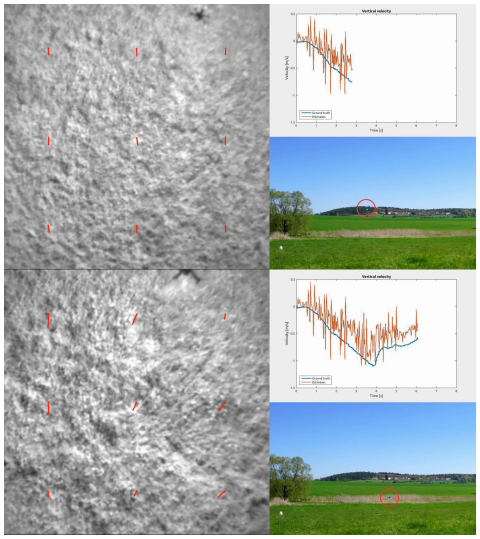
Table 8.5: Maximal and average 3-D velocity estimation error with real-world dataset for different trajectories.

	Position error [m]					
	Maximal			**Average**		
Trajectory	**x**	**y**	**z**	**x**	**y**	**z**
Flat	1.0	0.76	1.88	0.36	0.30	0.51
Vertical	-	-	1.4	-	-	0.59
1 m/s	0.17	0.47	0.96	0.058	0.17	0.43
2 m/s	1.7	0.52	1.1	0.96	0.22	0.59

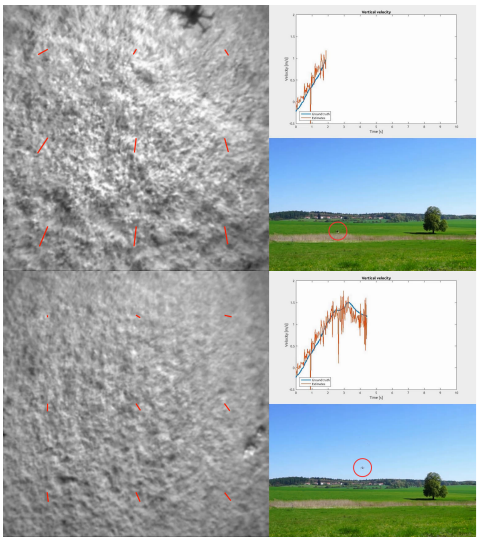
Table 8.6: Maximal and average 3-D position estimation error with real-world dataset for different trajectories.



(a) : No horizontal movement, vertical speed 1 $\mathrm{m\,s^{-1}}$.

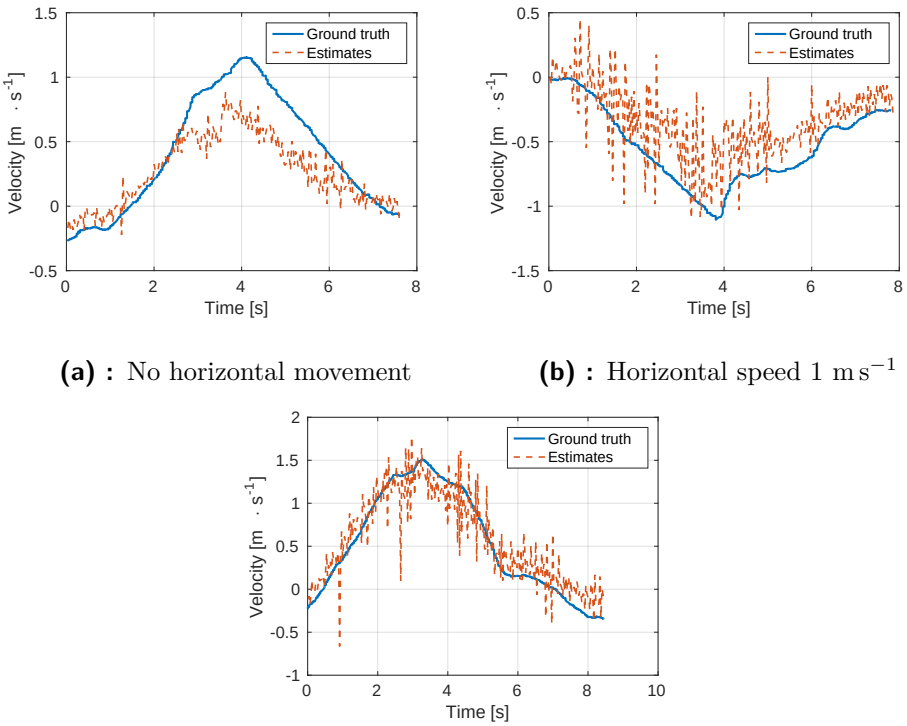


(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$, vertical 1 $\mathrm{m\,s^{-1}}$.



(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$, vertical 2 $\mathrm{m\,s^{-1}}$.

Figure 8.7: Snapshots from a video compiling the vertical velocity estimation testing. Highlighted vectors in the image show measured optical flow. The video can be viewed on `https://youtu.be/K1bvQMz9fOo`.



(a) : No horizontal movement

(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$

(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$

Figure 8.8: Vertical rate estimation on real-world data at different horizontal speeds along global x-axis.

8.2.2 Yaw velocity estimation testing

Yaw rate estimation accuracy was tested with three different trajectories already employed in simulator testing 7.2.2. All the tests were done in a 3 m altitude, and the UAV was commanded to rotate at a rate of 0.5 $\mathrm{rad\,s^{-1}}$ during the flight. We created a video compiling the experiment, see figure 8.9. The three trajectories differ in the horizontal velocity along the global x-axis - we used speeds of 0, 1 and 2 $\mathrm{m\,s^{-1}}$.

The resulting velocity estimates were integrated and compared to ground truth in the same way as in the other experiments. The resulting errors are shown in table 8.7. The estimates compared to ground truth are shown in figure 8.10.

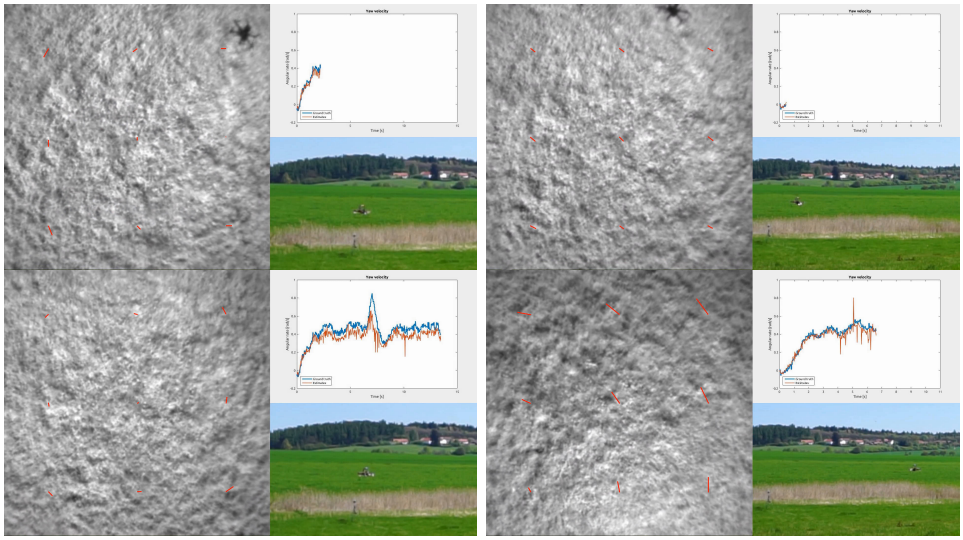
Figures show that the yaw rate estimation is very accurate and independent of the horizontal movement. However, with the current parameters, the estimates saturate at a rate circa 0.6 $\mathrm{rad\,s^{-1}}$. This claim is also supported by the results from the table that show very low average velocity error.

The maximal velocity errors in the table show that the results sometimes contain outliers which are visible on the figures as well. However, deviated results under 0.6 $\mathrm{rad\,s^{-1}}$ occur rarely.

In conclusion, we can thus state that the yaw velocity estimation works with a very low average error. Deviated measurements sometimes occur but a filtering method can be presumably developed to cope with this issue.

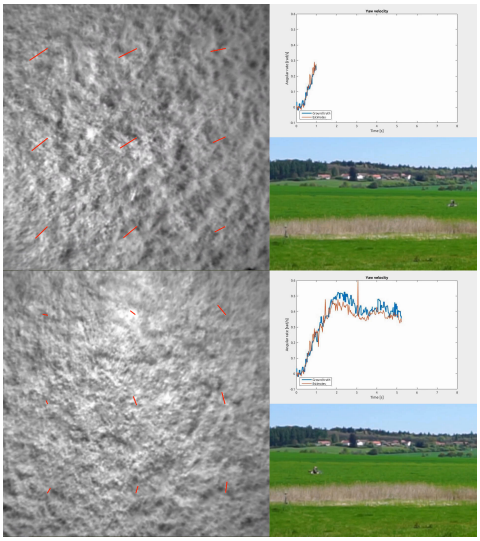
	Velocity error [$\mathrm{rad\,s^{-1}}$]		**Position error** [rad]	
Horiz. speed	**Maximal**	**Average**	**Maximal**	**Average**
0	0.42	0.070	0.95	0.43
1 $\mathrm{m\,s^{-1}}$	0.69	0.050	0.38	0.13
2 $\mathrm{m\,s^{-1}}$	0.22	0.039	0.22	0.090

Table 8.7: Yaw velocity estimates errors for trajectories that differ in horizontal speed along the global x-axis. The yaw rate was 0.5 $\mathrm{rad\,s^{-1}}$ same for all.



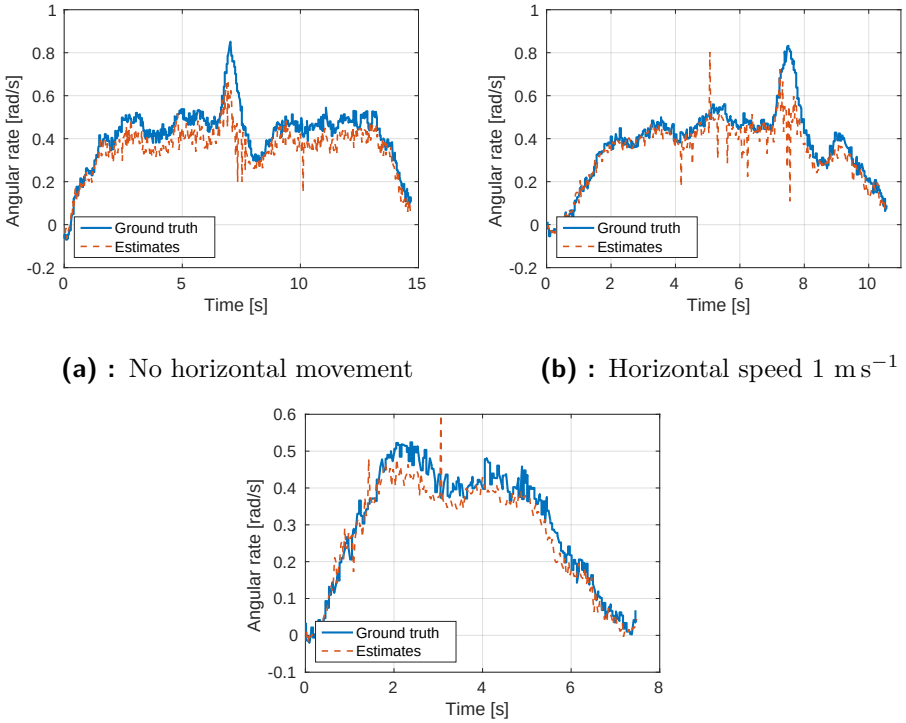
(a) : Yaw rate 0.5 rad s^{-1}, no horizontal movement.

(b) : Yaw rate 0.5 rad s^{-1}, horizontal speed 1 m s^{-1}.



(c) : Yaw rate 0.5 rad s^{-1}, horizontal speed 2 m s^{-1}.

Figure 8.9: Snapshots from a video compiling the yaw rate estimation testing. Highlighted vectors in the image show measured optical flow. The video can be viewed on `https://youtu.be/KTO1fcT1h3M`.



(a) : No horizontal movement

(b) : Horizontal speed 1 $\mathrm{m\,s^{-1}}$

(c) : Horizontal speed 2 $\mathrm{m\,s^{-1}}$

Figure 8.10: Yaw rate estimation with different horizontal speeds along global x-axis. The yaw rate was 0.5 $\mathrm{rad\,s^{-1}}$ same for all the trajectories.

Chapter 9

Conclusion

We conclude that all points of the assignment of the thesis was fulfilled.

We implemented an FFT-based method that uses Phase correlation for optical flow calculation. The calculated optical flow is combined with data from other sensors and filtered with RANSAC inspired method. The implementation is described as a whole in section 5.1, the optical flow calculation method is depicted in 4.2.2, the Motion field equations used for conversion to relative velocity in 4.1 and the filtering procedure in 4.3.

We optimise the parameters in chapter 6 and verify the whole algorithm on datasets obtained from the simulator in chapter 7 and on real-world datasets in chapter 8. We compared our solution with Block matching algorithm in both chapters. Moreover, comparison with *PX4FLOW Smart Camera* was done in chapter 8.

Also, we designed, implemented and fully tested a method that estimates 3D translation and rotation in yaw. The design and implementation are described in section 5.2 and the method is tested on a simulator dataset in chapter 7.2 and on a real-world dataset in section 8.2.

The results from real-world verification show that our FFT (Phase correlation) method provides horizontal velocity estimates with an average error around 0.1 $\mathrm{m\,s^{-1}}$. The results with Block matching algorithm are very similar. Both approaches have at least two times better average accuracy than *PX4FLOW Smart Camera* according to our tests.

The experimentation on real-world data was also done with our method estimating 3D translation and rotation in yaw. It showed that yaw rate could be measured with the average error as small as 0.04 $\mathrm{rad\,s^{-1}}$. This makes the method usable for yaw estimation in magnetometer-denied environments. Horizontal velocity estimates are worse with this approach. The average error goes up to around 0.4 $\mathrm{m\,s^{-1}}$. Vertical velocity can be estimated with an average error of 0.2 $\mathrm{m\,s^{-1}}$.

Apart from higher accuracy and improved reliability, we must stress that the results of the work are highly modular. The software was implemented as a node for Robot Operating System. This means that an adjustment for another UAV running ROS is a matter of minutes. Furthermore, the parameters of the camera, altitude sensor, lens or gyroscope can be changed easily to adjust for the current environment.

Still, more development can be done on the work. As a direct follow-up to the results presented, it is planned that the method will be employed in feedback onboard Model Predictive Control (MPC) in the next months. Furthermore, we would like to focus filtering methods used with yaw velocity estimation, since the measurements sometimes tend to contain outliers.

Appendix A

CD contents

The description of each directory on the CD is written in table A.1.

Directory name	Description
sources	Software source code
thesis	This thesis in pdf format
videos	Videos of experiments

Table A.1: Table describing the contents of each directory on the CD.

Appendix B

Bibliography

[AFF04] F. Aubepart, M. El Farji, and N. Franceschini, *Fpga implementation of elementary motion detectors for the visual guidance of micro-air-vehicles*, 2004 IEEE International Symposium on Industrial Electronics, vol. 1, May 2004, pp. 71–76 vol. 1.

[BK12] M. R. Balazadeh Bahar and G. Karimian, *High performance implementation of the horn and schunck optical flow algorithm on fpga*, 20th Iranian Conference on Electrical Engineering (ICEE2012), May 2012, pp. 736–741.

[BLS16] Tomáš Báča, Giuseppe Loianno, and Martin Saska, *Embedded model predictive control of unmanned micro aerial vehicles*, 21st International Conference on Methods and Models in Automation and Robotics (MMAR), IEEE, 2016.

[BZF13] Adrien Briod, Jean-Christophe Zufferey, and Dario Floreano, *Optic-Flow Based Control of a 46g Quadrotor*, Workshop on Vision-based Closed-Loop Control and Navigation of Micro Helicopters in GPS-denied Environments, IROS 2013, 2013.

[CEU+15] Aasish C, Ranjitha E., Razeen Ridhwan U, Bharath Raj S, and Angelin Jemi L., *Navigation of uav without gps*, 2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE), Feb 2015, pp. 1–3.

[Con15] J. Conradt, *On-board real-time optic-flow for miniature event-based vision sensors*, 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dec 2015, pp. 1858–1863.

[FKC+13] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska, *Low-cost embedded system for relative localization in robotic swarms*, IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2013.

[GSM13] Nils Gageik, Michael Strohmeier, and Sergio Montenegro, *An autonomous uav with an optical flow sensor for positioning and navigation*, International Journal of Advanced Robotic Systems **10** (2013), no. 10, 341.

[Hei17] Adam Heinrich, *An optical flow odometry sensor based on the raspberry pi computer*, Master's thesis, Czech Technical University in Prague, 2017.

[HMTP13] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, *An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications*, Robotics and Automation (ICRA), 2013 IEEE International Conference on, May 2013, pp. 1736–1741.

[HS81] Berthold K.P. Horn and Brian G. Schunck, *Determining optical flow*, Artificial Intelligence **17** (1981), no. 1, 185 – 203.

[IHC05] R. Iyer, R. Holsapple, and P. Chandler, *A new method for the computation of motion from image sequences*, Proceedings of the 2005, American Control Conference, 2005., June 2005, pp. 4113–4118 vol. 6.

[IMB08] R. Iyer, J. Meixner, and R. Buckalew, *On an optical inertial navigation system - part i*, IEEE Transactions on Automatic Control **53** (2008), no. 8, 1850–1863.

[ITH08] R. Iyer, M. Toda, and R. Holsapple, *On an optical inertial navigation system - part ii*, IEEE Transactions on Automatic Control **53** (2008), no. 8, 1864–1875.

[Its17] Itseez, *The open source computer vision library*, `https://github.com/itseez/opencv`„ 2017, [Software library].

[JZS10] M. Joos, J. Ziegler, and C. Stiller, *Low-cost sensors for image based measurement of 2d velocity and yaw rate*, 2010 IEEE Intelligent Vehicles Symposium, June 2010, pp. 658–662.

[KB07] Jonghyuk Kim and Galen Brambley, *Dual optic-flow integrated navigation for small-scale flying robots*, Proc. of Australasian Conference on Robotics and Automation, Brisbane, Australia, 2007.

[KNF+14] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail, *A practical multirobot localization system*, Journal of Intelligent & Robotic Systems **76** (2014), no. 3-4, 539–562.

[MKK+15] V. More, H. Kumar, S. Kaingade, P. Gaidhani, and N. Gupta, *Visual odometry using optic flow for unmanned aerial vehicles*,

2015 International Conference on Cognitive Computing and Information Processing(CCIP), March 2015, pp. 1–6.

[PX413] PX4, *Px4flow smart camera*, `http://pixhawk.org/modules/px4flow`, 2013, [Website; version as of 29th April 2017].

[Sas15] Martin Saska, *Mav-swarms: Unmanned aerial vehicles stabilized along a given path using onboard relative localization*, International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2015.

[SBH16] Martin Saska, Tomas Baca, and Daniel Hert, *Formations of unmanned micro aerial vehicles led by migrating virtual leader*, 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), IEEE, 2016, pp. 1–6.

[SBS16] Vojtěch Spurný, Tomáš Báča, and Martin Saska, *Complex manoeuvres of heterogeneous mav-ugv formations using a model predictive control*, Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference on, IEEE, 2016, pp. 998–1003.

[SBSHM09] John Stowers, Andrew Bainbridge-Smith, Michael Hayes, and Steven Mills, *Optical flow for heading estimation of a quadrotor helicopter*, International Journal of Micro Air Vehicles **1** (2009), no. 4, 229–239.

[SBT+16] Martin Saska, Tomáš Báča, Justin Thomas, Jan Chudoba, Libor Přeučil, Tomáš Krajník, Jan Faigl, Giuseppe Loianno, and Vijay Kumar, *System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization*, Autonomous Robots (2016), 1–26.

[SCP+14] Martin Saska, Jan Chudoba, Libor Přeučil, Justin Thomas, Giuseppe Loianno, Adam Třešnák, Vojtěch Vonášek, and Vijay Kumar, *Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance*, International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2014.

[SKP14] Martin Saska, Zdenek Kasl, and Libor Přeucil, *Motion planning and control of formations of micro aerial vehicles*, 19th World Congress of the International Federation of Automatic Control (IFAC) (2014).

[SKSB17] Martin Saska, Vít Kratký, Vojtěch Spurný, and Tomáš Báča, *Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control.*, ETFA, IEEE, 2017.

[SKV+13] Martin Saska, Tomáš Krajník, Vojtěch Vonásek, Petr Vaněk, and Libor Přeučil, *Navigation, localization and stabilization of formations of unmanned aerial and ground vehicles*, International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2013.

[SKV+14] Martin Saska, Tomas Krajnik, Vojtěch Vonásek, Zdeněk Kasl, Vojtěch Spurný, and Libor Preucil, *Fault-tolerant formation driving mechanism designed for heterogeneous mavs-ugvs groups*, Journal of Intelligent & Robotic Systems **73** (2014), no. 1-4, 603.

[SNSAC15] A. Santamaria-Navarro, J. Solà, and J. Andrade-Cetto, *High-frequency mav state estimation using low-cost inertial and optical flow measurement units*, Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, Sept 2015, pp. 1864–1871.

[SVC+16] Martin Saska, Vojtěch Vonásek, Jan Chudoba, Justin Thomas, Giuseppe Loianno, and Vijay Kumar, *Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles*, Journal of Intelligent & Robotic Systems **84** (2016), no. 1-4, 469–492.

[SVKP12] Martin Saska, Vojtěch Vonásek, Tomáš Krajník, and Libor Přeučil, *Coordination and navigation of heterogeneous uavs-ugvs teams localized by a hawk-eye approach*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2012.

[SVKP14] Martin Saska, Vojtěch Vonásek, Tomáš Krajník, and Libor Přeučil, *Coordination and navigation of heterogeneous mav-ugv formations localized by a 'hawk-eye'-like approach under a model predictive control scheme*, The International Journal of Robotics Research **33** (2014), no. 10, 1393–1412.

[SVP14] Martin Saska, Jan Vakula, and Libor Přeućil, *Swarms of micro aerial vehicles stabilized under a visual relative localization*, IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014.

[Sze10] Richard Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[TG17] Tersus-GNSS, *Precis-bx305 gnss rtk board*, `https://cdn.shopify.com/s/files/1/0928/6900/files/Datasheet_Precis-BX305_EN.pdf?336381172763480191`, 2017, [Datasheet; version as of 7th May 2017].

[XLZ+14] B. Xian, Y. Liu, X. Zhang, M. Cao, and F. Wang, *Hovering control of a nano quadrotor unmanned aerial vehicle using optical*

flow, Proceedings of the 33rd Chinese Control Conference, July 2014, pp. 8259–8264.