

EELISA Student Scientific Competition 2024

Optical flow-based yaw rate estimation and fault detection on drones

Author:

Gabor Jevuczo



Budapest University of Technology and Economics
Faculty of Transportation Engineering and Vehicle Engineering

Advisor:

Peter Bauer Ph.D.

May 22, 2024

Contents

1	Introduction	1
2	Optical flow equations in the camera coordinate system	2
2.1	Coordinate frames	2
2.2	Transformations	3
2.2.1	Transformation between the body and camera frame:	4
2.3	Pinhole camera model	5
2.4	Feature point dynamics	5
2.5	Equations	6
2.6	Solution for system of equations	8
2.7	Singularity examination	8
3	Video Processing	11
3.1	Flight test trajectories	11
3.2	Oriented FAST and Rotated BRIEF	12
3.3	Lucas-Kanade method	13
4	Sensor fault detection	16
4.1	Angular rate sensor fault types:	16
4.2	Fault detection method	16
4.3	Tuning of threshold values	17
4.4	Fault detection results	20
5	Summary	23

Notations

OF	-	Optical Flow
FAST	-	Features from Accelerated Segment Test
SURF	-	Speeded-Up Robust Features
SIFT	-	Scale-Invariant Feature Transform
ORB	-	Oriented FAST and Rotated BRIEF
FPS	-	Frames Per Second
BRIEF	-	Binary Robust Independent Elementary Features
ROC	-	Receiver Operating Characteristic
FA	-	False Alarm
MD	-	Missed Detection
TP	-	True Positive
FP	-	False Positive
TN	-	True Negative
FN	-	False Negative
Tol1	-	Threshold of the residual
Tol2	-	Threshold of the error counter

Chapter 1

Introduction

Nowadays the quick spread and size growth of drones require the use of redundant systems. The angular rate sensors have an important role in the control of drones, therefore fast fault detection is needed for the safety of people and properties. Most drones are equipped with a camera to perform various functions (e.g., video capturing, goggles) therefore it is convenient to use for the detection of sensor errors.

The aim of my research was to estimate the yaw rate from a gimbal-stabilized downward-looking camera's images with the help of optical flow and to detect artificially generated sensor faults on real flight data using the obtained angular rate.

First, the equations were derived, which establish the relation between the optical flow and the velocities of the aerial vehicle with the help of the pinhole camera model. These equations were previously derived for a fixed-wing unmanned aerial vehicle with a mostly forward looking tilted camera placed ahead of the center of gravity. I present the actualization of these equations for the current problem in the first part.

The definition of optical flow is presented in the second part. To calculate the optical flow, it is necessary to find the same feature points on several consecutive frames. However, due to the use of approximating methods, it is important to filter out outliers and incorrect pairings, for which forward-backward optical flow calculation is used. The obtained optical flow and the equations derived in the previous section can be used to estimate the yaw rate.

In the last part, I describe the examined sensor faults, the fault detection method and results. These faults were generated on real flight data and a counter and threshold-based fault detection method is presented. For this method, the angular rate calculated from the camera image and the measured angular rate were compared, using the data of two separate sensors, one of which contains the generated faults. This comparison is evaluated in every time step, and if the result is above a certain threshold, then the value of a counter is increased. If it is not, then it is decreased by a predetermined value. If the counter reaches a certain threshold then a fault can be declared and from this point on, we consider the sensor to be faulty.

Chapter 2

Optical flow equations in the camera coordinate system

In this chapter the connection between the camera image and the feature points detected from the environment are presented. The equations in this chapter were first derived by Joseph J. Kehoe et al. in their work [1] where they also demonstrated multiple methods to solve the obtained equations. Based on their achievement Szabolcs Kun and Peter Bauer introduced an approximate two step linear solution [2] and in their subsequent work [3] they showed that the equations are applicable for sensor fault detection. The main goal of my present work is to analyze the equations applicability for yaw rate fault detection considering real flight data and real camera images.

2.1 Coordinate frames

To understand how the vehicle moves related to the feature points in the surrounding environment, first the definition of the coordinate frames is required. The applied coordinate frames are shown in Figure 2.1. Throughout my work three coordinate frames are considered:

- Earth coordinate frame:

Also called as North-East-Down coordinate system. This frame is fixed to the earth surface and its x_e axis points towards the geodetic north (N) (also known as true north, the most northern point of the ellipsoid), y_e axis towards the geodetic east (E), and z_e axis points directly down (D) towards the centre of the Earth (perpendicular to the tangent plane).

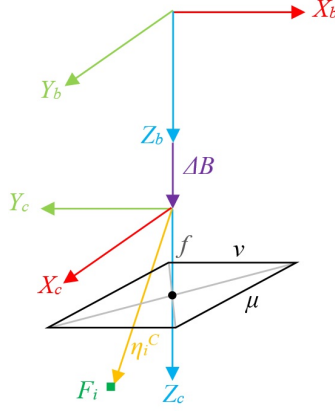


Figure 2.1: The applied coordinate systems.

- Body coordinate frame:

The body coordinate system is fixed to the center of gravity (CoG) of the aerial vehicle, its x_b axis points towards the front of the vehicle, y_b axis points to the right side of the vehicle and z_b axis points downwards.

- Camera coordinate frame:

This coordinate system is fixed to the camera sensor. In which we can interpret the information provided by the camera, for example we can calculate the optical flow in this frame. The x_c axis points towards the right of the vehicle, y_c to the rear of the vehicle, and z_c downwards. Its position in the body frame is described by the ΔB vector, which shows that the camera is mounted approximately vertically below the CoG ($\Delta B = [0.0285 \ -0.0115 \ 0.175] \ [m]$).

2.2 Transformations

To obtain a more simple form of the equations later on, they are derived in the camera coordinate system. Therefore the body velocity, body angular rate and the camera position (ΔB) vectors are transformed to the camera frame determined by the Z-Y-X Euler angle rotations. This means three successive rotations with the Euler angles which are the yaw angle (Ψ), pitch angle (Θ) and roll angle (Φ). These angles describe a fixed sequence of rotations, the first rotation is about the z axis with the yaw angle, the second is about the y axis with pitch angle and the last rotation is about the x axis with roll angle.

$$T_{\Psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_{\Theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \quad T_{\Phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \quad (2.1)$$

2.2.1 Transformation between the body and camera frame:

The velocity and the angular rate are measured in the body frame. In order to transform them into camera frame first transformation to earth frame has to be executed, which can be realized with the following transformation matrix:

$$T_{EB} = (T_{BE})^T = (T_{\Phi} \cdot T_{\Theta} \cdot T_{\Psi})^T \quad (2.2)$$

Following that, transformation from earth to camera system has to be carried out. However the camera is vertically stabilized by a gimbal. Therefore, during earth-camera transformation, we only need to transform with Φ_g gimbal roll and Θ_g gimbal pitch angles, so that the effect of Ψ yaw angle rotation is the same as for the drone. However, the x_c axis points towards the right of the vehicle and y_c towards the rear of the vehicle, and this corresponds to a 90 degree rotation around the z axis. Thus after the transformation with the yaw angle another multiplication with T_{90° is necessary.

$$T_{90^\circ} = \begin{bmatrix} \cos(90^\circ) & \sin(90^\circ) & 0 \\ -\sin(90^\circ) & \cos(90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

$$T_{CE} = T_{90^\circ} \cdot T_{\Psi} \quad (2.4)$$

As a conclusion, the body to camera transformation matrix resulting in the following form (2.5).

$$T_{CB} = T_{CE} \cdot T_{EB}^g \quad (2.5)$$

According to (2.5) the $\underline{v}_b$ velocity vector, the ω_b angular rate vector and the ΔB position vector can be expressed in the camera system.

$$\underline{v}^C = T_{CB} \cdot \underline{v}^B \quad (2.6)$$

$$\omega^C = T_{CB} \cdot \omega^B \quad (2.7)$$

$$\Delta C = T_{CB} \cdot \Delta B \quad (2.8)$$

2.3 Pinhole camera model

The main task of a digital camera is to create a 2 dimensional image from our 3 dimensional world. The most simple camera model is the pinhole camera (camera obscura). The pinhole camera model defines the mathematical relationship between the coordinates of a spatial point (later called feature point) and its projection onto the image plane of the camera.

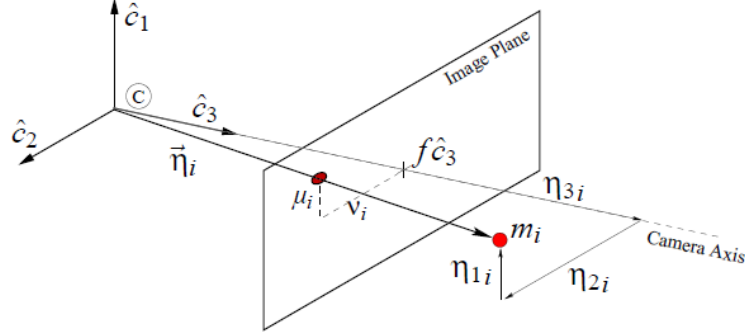


Figure 2.2: Projection of point m to the cameras image plane. [1]

The origin of the camera coordinate system is in the camera's focal point. The camera's image plane is perpendicular to the principal axis. The distance between the origin and the image plane is called the focal length and it is measured along the principal axis [4]. Position of the i -th feature point is described by the $(\eta_i = [\eta_{1i} \ \eta_{2i} \ \eta_{3i}]^T)$ vector. The projection of this point on the image plane is denoted by the $(\mu_i \ \nu_i)$ coordinate pair. The projection is described by the following equations [3]:

$$\mu_i = f \frac{\eta_{1i}}{\eta_{3i}} \quad (2.9)$$

$$\nu_i = f \frac{\eta_{2i}}{\eta_{3i}} \quad (2.10)$$

2.4 Feature point dynamics

Optical flow can be interpreted as the velocity of the feature point's projection on the image plane. Hence differentiating the pinhole camera model, (2.9) and (2.10) equations with respect to time, the optical flow can be obtained. After performing the differentiation, the (2.11) system of equations is acquired [1], in which $(\dot{\mu}_i \ \dot{\nu}_i)$ coordinate pair denotes the optical flow and $\dot{\eta}_i^C$ is the velocity of the feature point expressed in the camera coordinate system.

$$\begin{bmatrix} \dot{\mu}_i \\ \dot{\nu}_i \end{bmatrix} = \frac{f}{\eta_{3i}^2} \begin{bmatrix} \eta_{3i} & 0 & -\eta_{1i} \\ 0 & \eta_{3i} & -\eta_{2i} \end{bmatrix} \dot{\eta}_i^C \quad (2.11)$$

The $(\dot{\mu}_i \dot{\nu}_i)$ coordinate pair can be acquired from the image data, if its possible to detect the feature points on two consecutive frames. In this case the position of the feature points on the image plane can be determined and the optical flow can be calculated with the help of numerical algorithms (see in Chapter 3).

The velocity of the i -th feature point can be determined by the (2.12) equation [3].

$$\dot{\eta}_i^C = \begin{bmatrix} -I_3 & [\Delta^C \times] + [\eta_i^C \times] \end{bmatrix} \begin{bmatrix} V^C \\ \omega^C \end{bmatrix} \quad (2.12)$$

In (2.12) the negative 3x3 identity matrix $(-I_3)$ originates from the phenomenon that in which direction the drone moving the feature points will move in an opposite direction on the camera's image plane. For example when the drone moves forward the feature points move backward.

2.5 Equations

Substituting (2.12) to (2.11) and expressing it, results in the following system of equations:

$$\begin{aligned} \dot{\mu}_i &= \frac{\eta_{2i} f r^C}{\eta_{3i}} - \frac{f u^C}{\eta_{3i}} - f q^C + \frac{\eta_{1i} f w^C}{\eta_{3i}^2} - \frac{f q^C z^C}{\eta_{3i}} + \frac{f r^C y^C}{\eta_{3i}} - \frac{\eta_{1i}^2 f q^C}{\eta_{3i}^2} + \frac{\eta_{1i} \eta_{2i} f p^C}{\eta_{3i}^2} \\ &\quad + \frac{\eta_{1i} f p^C y^C}{\eta_{3i}^2} - \frac{\eta_{1i} f q^C x^C}{\eta_{3i}^2} \\ \dot{\nu}_i &= f p^C - \frac{f v^C}{\eta_{3i}} - \frac{\eta_{1i} f r^C}{\eta_{3i}} + \frac{\eta_{2i} f w^C}{\eta_{3i}^2} + \frac{f p^C z^C}{\eta_{3i}} - \frac{f r^C x^C}{\eta_{3i}} + \frac{\eta_{2i}^2 f p^C}{\eta_{3i}^2} - \frac{\eta_{1i} \eta_{2i} f q^C}{\eta_{3i}^2} \\ &\quad + \frac{\eta_{2i} f p^C y^C}{\eta_{3i}^2} - \frac{\eta_{2i} f q^C x^C}{\eta_{3i}^2} \end{aligned} \quad (2.13)$$

where $(p^C q^C r^C)$ and $(u^C v^C w^C)$ are the drone's angular rate and velocity components respectively, transformed to the cameras coordinate system. $(x^C y^C z^C)$ are the camera position vector coordinates with respect to the body frame, expressed in the camera frame. Furthermore $(\eta_{1i} \eta_{2i} \eta_{3i})$ are the i -th feature point position coordinates in the camera system, $(\mu_i \nu_i)$ are the i -th feature point coordinates on the image plane and $(\dot{\mu}_i \dot{\nu}_i)$ are the i -th feature point's velocity (optical flow) on the image plane.

The (2.13) system of equation contains $6 + 3N$ unknowns for which there are $2N$ equations. However expressing the η_{1i} and η_{2i} feature point position coordinates from the pinhole camera model, (2.9) and (2.10), and substituting into (2.13), the number of unknowns can be decreased [3].

$$\eta_{1i} = \frac{\mu_i f}{\eta_{3i}} \quad (2.14)$$

$$\eta_{2i} = \frac{\nu_i f}{\eta_{3i}} \quad (2.15)$$

$$\begin{aligned} \dot{\mu}_i = & r^C \left(\nu_i + \frac{f y^C}{\eta_{3i}} \right) - q^C \left(f + \frac{\mu_i^2}{f} + \frac{f z^C}{\eta_{3i}} + \frac{\mu_i x^C}{\eta_{3i}} \right) \\ & + p^C \left(\frac{\mu_i y^C}{\eta_{3i}} + \frac{\mu_i \nu_i}{f} \right) + \frac{\mu_i w^C}{\eta_{3i}} - \frac{f u^C}{\eta_{3i}} \\ \dot{\nu}_i = & p^C \left(f + \frac{\nu_i^2}{f} + \frac{f z^C}{\eta_{3i}} + \frac{\nu_i y^C}{\eta_{3i}} \right) - r^C \left(\mu_i + \frac{f x^C}{\eta_{3i}} \right) \\ & - q^C \left(\frac{\nu_i x^C}{\eta_{3i}} + \frac{\mu_i \nu_i}{f} \right) - \frac{f v^C}{\eta_{3i}} + \frac{\nu_i w^C}{\eta_{3i}} \end{aligned} \quad (2.16)$$

Then more assumptions/neglections can be utilized. During the present work it is assumed that the drone fly on a constant altitude and due to the camera frame's z_c axis coincides with the NED frame's *down* axis (because the gimbal holds the camera continuously vertically), therefore the w^C velocity component is assumed to be zero ($w^C = 0$).

The u^C and v^C components are not zero since the drone not only can move to a longitudinal direction but also can do lateral movement too. This is an important difference between a fixed winged UAV and a multicopter, in case of a fixed wing UAV we can assume that $v^C = 0$. Furthermore we assume that besides the yaw rate (r^C), the other two angular rate components are zero ($p^C = 0$, $q^C = 0$). And because the camera is mounted along the z_c axis, approximately vertically under the CoG, the camera position vector's x^C and y^C coordinates can be assumed to be zero (see ΔB vector).

Applying these assumptions/neglections to the (2.16) system of equations results in the following:

$$\dot{\mu}_i = \nu_i r^C - \frac{f u^C}{\eta_{3i}} \quad (2.17)$$

$$\dot{\nu}_i = -\mu_i r^C - \frac{f v^C}{\eta_{3i}} \quad (2.18)$$

The obtained system of equations contains only $3 + N$ unknowns for $2N$ equations. Thus, successfully calculating optical flow for at least three feature points, this system of equations can be solved.

2.6 Solution for system of equations

The (2.17) and (2.18) establish a nonlinear system of equations, where the unknowns are r^C, u^C, v^C, η_{3i} . However considering an instantaneous constant ratio between the velocities $u^C = kv^C$ results in:

$$\frac{f u^C}{\eta_{3i}} = k \frac{f v^C}{\eta_{3i}} \quad (2.19)$$

Expressing $\frac{f v^C}{\eta_{3i}}$ from (2.18) and substituting into (2.19) the result can be substituted into (2.17) resulting in:

$$\dot{\mu}_i = \nu_i r^C + k \mu_i r^C + k \dot{\nu}_i \quad (2.20)$$

This is a bilinear system of equations in which η_{3i} the feature dependent unknown is eliminated. (2.20) can be solved for r^C with an iterative method. In the first iteration k is considered to be zero then the yaw rate r^C is calculated with the least square method. After that the obtained r^C is substituted into (2.20) and k is calculated with the least squares method. In the next iteration the recently calculated k is substituted into (2.20) and r^C is calculated and so on, continuing with the iterations. The number of iterations were limited to 100. In the case of onboard implementation a convergence criterion should be applied to eliminate unnecessary calculations but in this offline calculation all 100 steps were executed, which proved to be sufficient.

2.7 Singularity examination

In (2.19) equation ($k = \frac{u^C}{v^C}$) can lead to singularity if v^C is zero or close to zero. Figure 2.3 shows the k values calculated from the sensor data of Flight1 (flight details are presented in the next Chapter 3), in which the zero crossings of v^C cause large spikes. Note that in Figure 2.3 k is divided by 100.

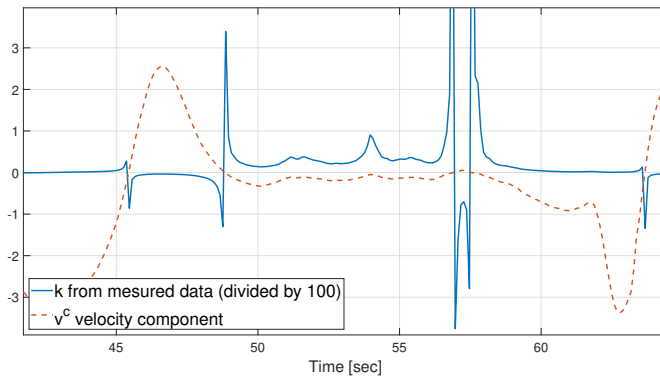
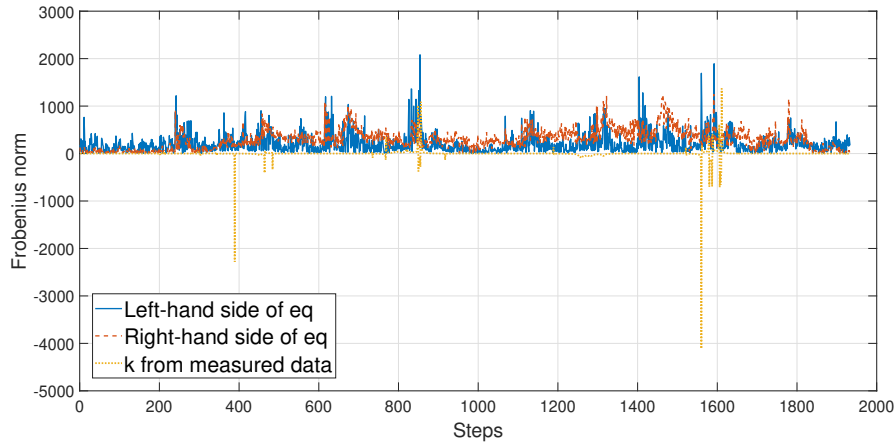


Figure 2.3: k values calculated from the sensor data of Flight1

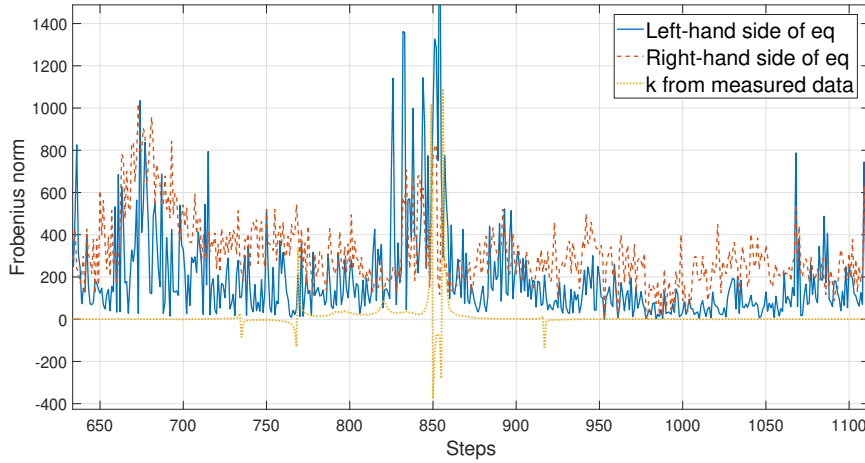
The detection of these singularities were an important task in this work. In the (2.18) equation if v^C value is zero then $(\mu_i r^C + \dot{\nu}_i = 0)$ can be written. Substituting this into a reordered version of (2.20) it can be seen that both sides of (2.21) has to be a zero vector.

$$\dot{\mu}_i - \nu_i r^C = k (\mu_i r^C + \dot{\nu}_i) \quad (2.21)$$

Calculating the Frobenius norm of both sides of (2.21) will show that, these are not zero vectors. This is mainly due to disturbances and noises caused by pixel faults and gimbal angle measurement uncertainties, therefore their norms will not be close to zero even when v^C is zero or close to zero. Figure 2.4 shows the Frobenius norms of both sides of (2.21).



(a) Value of norms throughout the whole flight.



(b) Figure 2.4a zoomed between 650 and 1100 steps

Figure 2.4: The Frobenius norms of (2.21) equation's both sides.

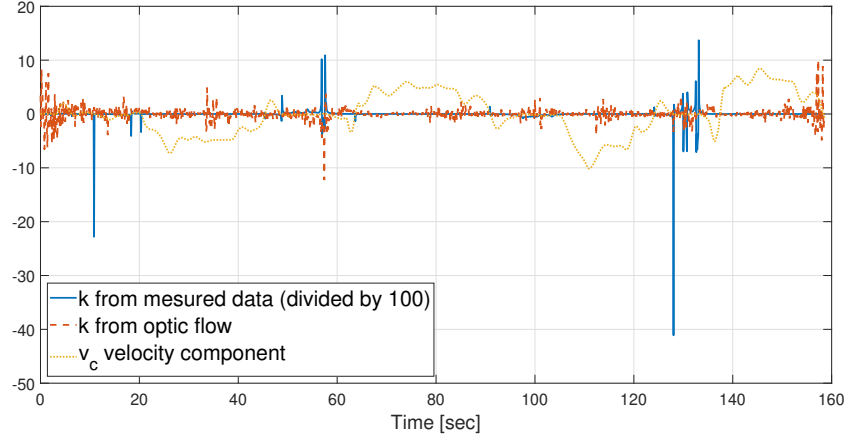


Figure 2.5: The k values calculated from optical flow and measured data

As a result of these noises, singularities did not appear in the calculations. The k values, calculated with the iterative method are plotted together with the k values, calculated from the measured data are shown in Figure 2.5. Note that in Figure 2.5 k is divided by 100 for better visibility purposes. As a result, it is not necessary to detect singularities.

Chapter 3

Video Processing

The aim of this chapter is to obtain $\dot{\mu}_i$, $\dot{\nu}_i$, μ_i , ν_i values for equation (2.20). Bauer and Kun [3], simulated UAV flight data in MATLAB and then they exported it to Unreal-Carla, in which they simulated multiple different environments (like city, forest). They created camera images in Unreal-Carla on which they applied the previously mentioned two step linear method and compared the results with the MATLAB simulation data. Throughout their work the incorrect matches between the tracked points caused uncertainties in the estimation of the angular rates. To eliminate these incorrect matches a backward optical flow was used.

3.1 Flight test trajectories

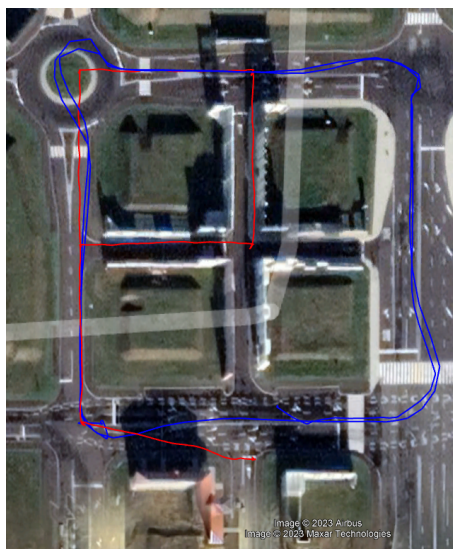


Figure 3.1: Trajectories of multicopter drone flights above ZalaZONE Smart City

For cost efficiency, two flight test trajectories were considered from another project (see [5]). These trajectories were acquired by a DJI M600 multicopter equipped with downward looking camera. The trajectories are shown in Figure 3.1. The first flight (Flight1) is a faster ($\sim 6\text{m/s}$) large square route with longer turns ($\sim 6\text{sec}$) with a moving car in the camera's field of view. The second flight (Flight2) is a slower ($\sim 1\text{m/s}$) small square route with shorter turns ($\sim 4\text{sec}$) without a car in the camera's field of view.

3.2 Oriented FAST and Rotated BRIEF

The *goodFeaturesToTrack* (Shi-Tomasi corner detector) applied in [3] wasn't able to detect enough feature points on real camera images with 10FPS sampling rate (see Figure 3.2). Therefore to detect and describe feature points, the *ORB* (Oriented FAST and Rotated BRIEF) function - implemented in the OpenCV library - was used. An alternative method could be the *SIFT* or *SURF* algorithm but these are patented algorithms, hence *ORB* was applied in this work. The *ORB* function finds the feature points on every level of a multi level image pyramid with the help of the *FAST* corner detector algorithm. After that, with the use of *BRIEF* algorithm it creates the description of the feature points. *ORB* was used with the following parameters, image pyramid with one level and the number of feature points were maximized in 600.



Figure 3.2: Feature points detected by *Shi-Tomasi* corner detector.



Figure 3.3: Feature points detected by *ORB* detector.

The main concept of the *FAST* algorithm is that, it is analyzing p pixel's environment, a circle which consists of 16 pixels with a radius of 3 pixels. All of these 16 pixels then are classified into one of the following two groups. Brighter than p plus a certain threshold t or darker than p minus a certain threshold t . If at least N pixels are darker then $(p - t)$ or N pixels are brighter than $(p + t)$, then it is detected as a feature point.

The *BRIEF* descriptor converts the detected feature points into a binary vector which contains only ones and zeros. Its advantage is that *SIFT* and *SURF* algorithms using floating point numbers to describe the feature points until the *BRIEF* is using binary numbers and this in fact helps with saving memory and computation time.

3.3 Lucas-Kanade method

After detecting and describing the feature points, the next step is to track these features on the successive images. Optical flow (OF) is a motion detection method that can be used to accomplish this task. Its main task is to define a vector for each feature point that is proportional to the movement of the point between two successive images. This requires the images to be processed one by one from the video.

The *OpticalFlowPyrLK* function - implemented in the OpenCV library - was used with the following parameters. The kernel size is 9x9, use an image pyramid with maximum 7 levels, maximum number of iterations is 10 and epsilon is 0.03.



Figure 3.4: Filtered feature points.

With the help of optical flow it is possible to find the location of the feature points on the $(k + 1)$ -th frame, that were previously detected by the *ORB* on the (k) -th frame. However, there are cases when these pairings are incorrect because of pixel intensity changes or too large movements between the two images. To eliminate incorrect matches between the tracked points, as a backward OF, *OpticalFlowPyrLK* was applied on the $(k + 1)$ -th feature points to find them on the (k) -th frame.

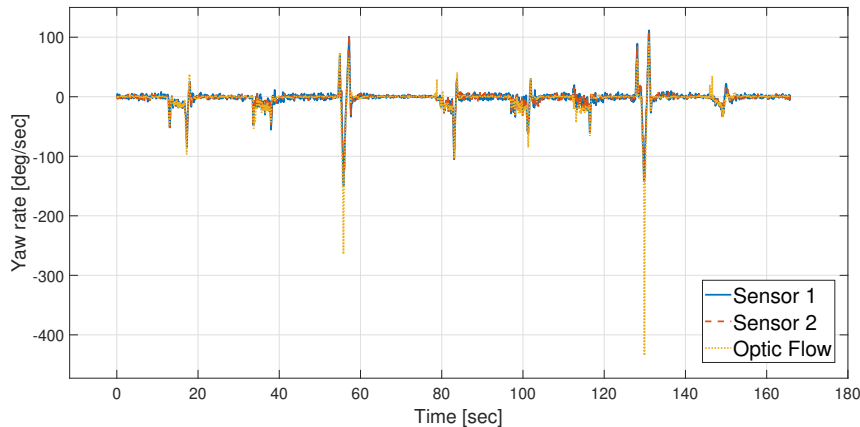
A threshold was applied based on the Euclidean distance between the points detected by the *ORB* and the points calculated from the backward OF. If the distance between the points exceeds the threshold than the incorrect matches are removed and not considered in the calculations later on. This helped to have a better angular rate estimation. Figure 3.4 shows the removed features with cyan blue. Due to the camera lens, a barrel distortion correction was applied on the feature points before OF calculation.

After finding the locations of the features on the $(k + 1)$ -th frame and filtering out the incorrect matches, the optical flow can be calculated with the help of numerical differentiation [3]:

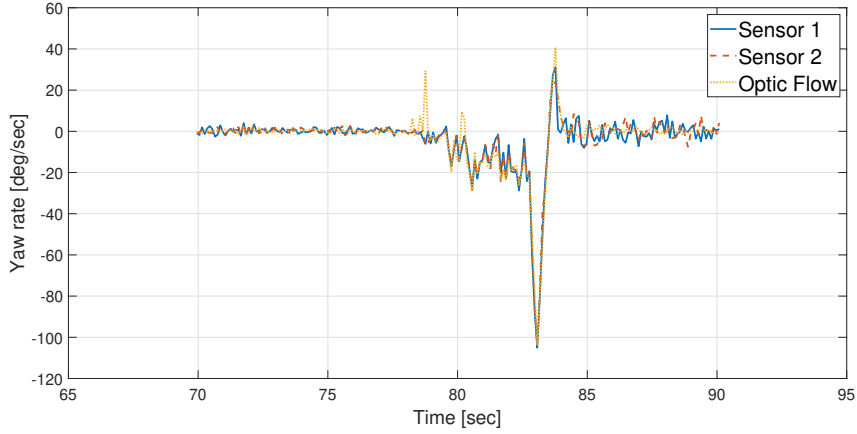
$$\dot{\mu}_i \approx \frac{\mu_i(k + 1) - \mu_i(k)}{\Delta t}, \quad \dot{\nu}_i \approx \frac{\nu_i(k + 1) - \nu_i(k)}{\Delta t} \quad (3.1)$$

where Δt is the time interval between the two consecutive frames.

By doing so the $\dot{\mu}_i$, $\dot{\nu}_i$, μ_i , ν_i values for (2.20) equation are determined. Substituting these into (2.20) and solving with the method shown in Section 2.6 will result in the following (see Figure 3.5 and Figure 3.6).

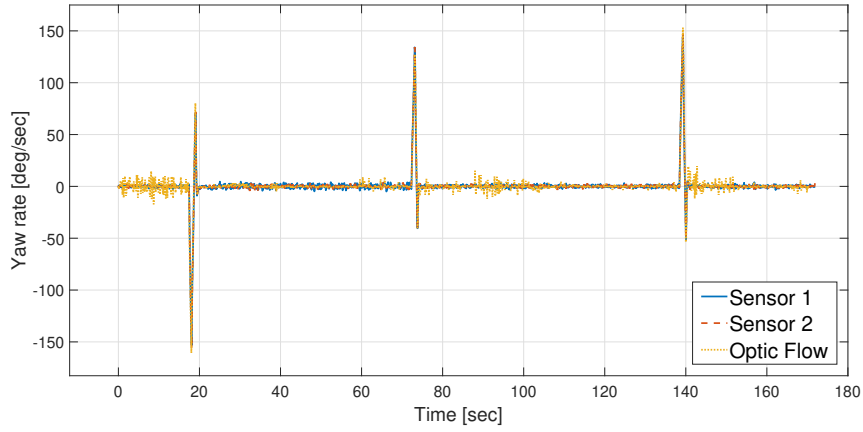


(a) Measured and estimated yaw rate of Flight1

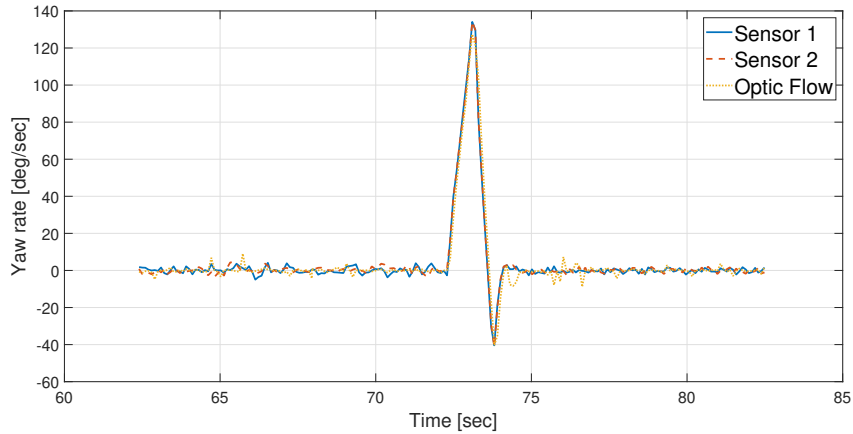


(b) Figure 3.5a zoomed between 70s and 90s

Figure 3.5: Comparison of measured and estimated yaw rate of Flight1



(a) Measured and estimated yaw rate of Flight2



(b) Figure 3.6a zoomed between 62s and 84s

Figure 3.6: Comparison of measured and estimated yaw rate of Flight2

The method gave good estimation for the yaw rates but in Figure 3.5a it can be seen that at the two higher speed turning (58s and 110s) it over estimates the angular rates. However this can be saturated if it is known to what maximum angular velocity the drone is limited to.

Chapter 4

Sensor fault detection

In the last chapter, I describe the considered sensor faults, the fault detection procedure and I present the results of its application.

4.1 Angular rate sensor fault types:

Guillermo Heredia et al. in their work [6] they present an autonomous model helicopter's typical sensor fault types where a detailed description of angular rate sensor errors can be found. Based on [3], four types of sensor faults were considered:

1. Total sensor failure: The sensor gives 0 as its output after the fault occurs.
2. Stuck with constant value sensor failure: The sensor gives the last value as constant output after the fault occurs.
3. Additive-type sensor failure: A constant A rad/s added to the measured value.
4. Multiplicative-type sensor failure: Sensor output is $100+P\%$ of the measured value.

4.2 Fault detection method

The DJI M600 multicopter drone has 3 independent angular rate sensors, from these, sensors 1 and 2 were considered. The sensor faults listed above were generated on the data of sensor 2 representing the faulty sensor. Sensor 1 was kept as a reference for an error free sensor because it is important not to detect an error free sensor as a faulty sensor. Note that the noise and bias characteristics of the two sensors are different so they never perfectly cover each others output. Throughout the tuning and testing two different flight data sets were used which were shown in Figures 3.5 and 3.6. Ahmet Arda Ozdemir et al. in [7] applied a counter-based residual thresholding fault detection method

in which they are using counters and residuals to detect wind turbine faults. Also in their work a detailed description of this method can be found.

In essence, the basis of this method is to form a residual between the angular rate estimated from the optical flow r_{OF} and the angular rate measured by the sensors (see (4.1)).

$$e_{1/2} = r_{sensor1/2} - r_{OF} \quad (4.1)$$

If this residual exceeds the threshold ($Tol1$) than the sensor's counter is incremented with the up-count amount (U), if not, the counter is decremented by the down-count amount (D) considering a 0 lower bound for the counter. These values (U and D) do not necessarily have to be equal. For the counter a fault declaration threshold ($Tol2$) can be introduced and if the counter reaches this threshold then a fault can be declared.

4.3 Tuning of threshold values

The $Tol1$ will determine the sensitivity of the fault detection, which is the smallest deviation that still can be detected. $Tol2$ will help to suppress the false alarms caused by the noise. The benefit of using this counter-base residual thresholding is that it can detect smaller faults compared to a simple threshold.

Tuning is a compromise because the noise on the residuals can cause false alarms if the noise are bigger than $Tol1$, i.e. detecting sensor faults while there is no real error. The other unfavorable case is when the measure of the fault is smaller than $Tol1$, this will result in missed detections i.e. not detecting a fault while the sensor is faulty. Therefore the task of finding the optimal $Tol1$ is a trade-off between the false alarms and missed detections. Tamas Grof and Peter Bauer in their article [8] are solving a similar optimization task.

Based on [8], a probability ($P(data > Tol1|no\ error)$) is defined which expresses how many times the residuals exceeds $Tol1$ while the sensor is fault free (false alarm (FA)). On the other hand, another probability is defined $P(data < Tol1|error)$ which expresses, how many times the residuals do not exceed $Tol1$ while the sensor is faulty (missed detection (MD)).

To obtain MD probability (P MD), stuck with constant value fault was generated on the data of sensor 2 with smaller (Flight1: $11.18^\circ/s$ Flight2: $-9^\circ/s$) and higher (Flight1: $-29.3^\circ/s$ Flight2: $23.45^\circ/s$) error values.

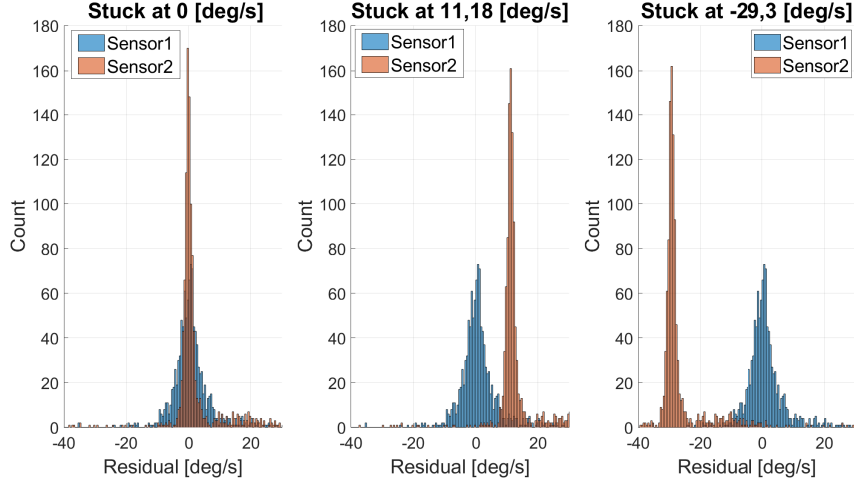


Figure 4.1: Residual histograms of Flight1 with different errors. (Sensor 1 is the faulty sensor, sensor 2 is the fault free sensor)

The total sensor failure was not considered during the tuning, because its residual is not distinguishable from the fault free sensor's residual. This can be seen in Figure 4.1 which shows that the residual histogram with the total sensor type failure can be overlapped with the fault free sensor's residual histogram. When there is no turning the lack of excitation can prevent the detection of small stuck errors while at the same time sensor noise can give false alarms. To eliminate this $P(data < Tol1|error)$ and $(Pdata > Tol1|no error)$ were calculated only when the measured signal exceeds $4^\circ/s$ denoting this threshold as *angularTHS*. This value was obtained experimentally by calculating the curves (like Fig. 4.2) for multiple *angularTHS* values considering the noise on the residuals.

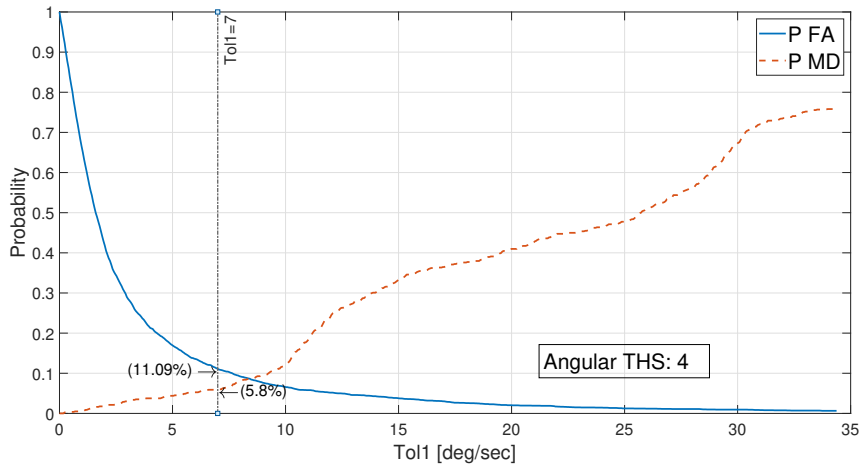


Figure 4.2: FA and MD probability distributions.

The probability of ($P(data > Tol1|noerror)$) is determined by counting how many times the error free sensor's residual exceeds $Tol1$ on both flights and then divide it by the number of all time steps (when evaluation was done). The probability $P(data < Tol1|error)$ can be determined by counting how many times the residuals (containing the faults mentioned above) exceed $Tol1$ and then divide it by the number of all time steps (when evaluation was done). Repeating this procedure to multiple $Tol1$ values between $0^\circ/s$ and $34^\circ/s$ will give a curve which is shown in Figure 4.2. This figure shows that increasing the value of $Tol1$ will lower the ratio of missed detection but it will increase the ratio of false alarms. The two ratios are equal around $8.37^\circ/s$ but I chose $Tol1 = 7^\circ/s$ value because in that case the ratio of missed detection is slightly lower while the ratio of false alarm is higher but with a well selected up-down counter the ratio of false alarms can be lowered.

$Tol2$ will make the decision whether the sensor has a failure or not. The decisions can be classified into 4 groups:

1. True Positive (TP): When the sensor is faulty and judged as faulty.
2. False Positive (FP): When the sensor is fault free and it is judged as faulty.
3. True Negative (TN): When the sensor is fault free and judged as fault free.
4. False Negative (FN): When the sensor is faulty and judged as fault free.

$Tol2$ was determined with the use of Receiver Operating Characteristic (ROC) curve [8]. The ROC curve shows the False Positive rate (FPR) with respect to the True Positive rate (TPR) throughout multiple threshold values (see Fig. 4.3).

FPR is calculated with error-free sensor data by counting how many times the counter exceeds $Tol2$ on both flights and dividing it by the sum of false positives and true negatives.

$$FPR = \frac{FP}{FP + TN} \quad (4.2)$$

To acquire the true positive rate stuck with constant value faults were generated during turns in 25 different times for both flights individually. The TPR can be calculated by counting how many times the counter exceeds $Tol2$ for both flights and divide it by the sum of true positives and false negatives.

$$TPR = \frac{TP}{TP + FN} \quad (4.3)$$

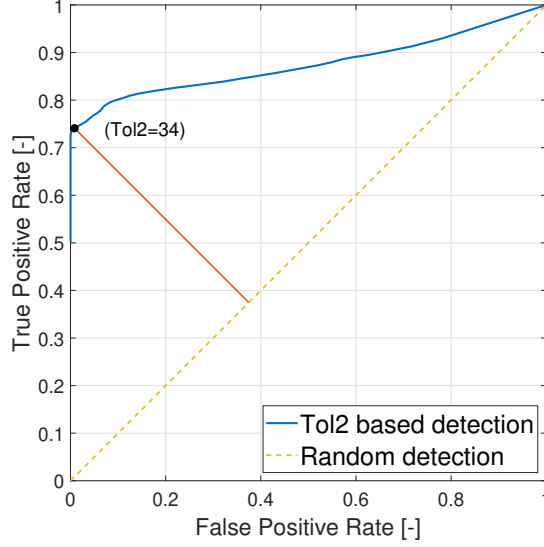


Figure 4.3: ROC curve to find the optimal $Tol2$ value.

Both FPR and TPR were calculated with the previously determined $Tol1 = 7^\circ/s$ only when the sensor data exceeds $angularTHS = 4^\circ/s$ and furthermore with $U = 1$ $D = 1$ parameters. By repeating calculations (4.2) and (4.3) for all $Tol2$ values between 0 and 100 will result the following ROC curve (see Fig. 4.3).

The ideal case would be the (0,1) point of Figure 4.3. Which means that the faulty sensor was always detected and there were no cases when the fault free sensor was judged as faulty. The main goal is to reach this point with the fault detection method as close as possible, i.e. the proportion of True Positive decisions should be as high as possible, while the proportion of False Positive decisions should be as low as possible. The optimal value for $Tol2$ can be found by the largest perpendicular distance from the diagonal line, thus $Tol2 = 34$ was selected. Where the proportion of False Positive decisions is $FPR = 0.0085$ and the True Positive decisions proportion is $TPR = 0.7408$.

4.4 Fault detection results

The fault detection method tuned with previously determined parameters ($Tol1 = 7^\circ/s$ $Tol2 = 34$ $angularTHS = 4^\circ/s$ $U = 1$ $D = 1$) was tested on two different flights.

As fault activation two different cases were considered. The steady case means fault is activated when the angular rate is close to constant and varying case means the fault is activated before large changes in yaw rate.

Table 4.1: Fault detection results.

Error type	Flight1 Varying	Flight1 Steady	Flight2 Varying	Flight2 Steady
Total sensor failure	2.17s	5.367s	67.4377s	69.5392s
Stuck with constant value sensor failure	13.795s	5.367s	67.6409s	69.5392s
Additive-type sensor failure	55.677s	59.677s	X	X
Multiplicative-type sensor failure	51.267s	55.267s	67.5392s	69.5392s

Table 4.1 shows the detection times from the fault activation times in case of successful detection and X marks the missed detections.

In case of additive-type sensor failure this method was able to detect $A = 2.1^\circ/s$ and higher added constant values on Flight1 and with multiplicative-type sensor failure it was able to detect $P = 8\%$ and higher percentages of the measured signal in both flights. The values in Table 4.1 belong to the smallest value of A and P that could be detected, with higher deviations the detection times should decrease.

From Table 4.1 I would like to highlight only 3 detections, one fast detection, one slower but still successful detection and an unsuccessful detection.

The additive-type sensor failure on Flight2 was not detected since the error counter did not reach the $Tol2$ value (see Fig: 4.6) within the examined time frame of Flight2 (only a segment of Flight2 was processed) due to the turns short duration and the 10Hz sampling. Similar happens with the other error types on Flight2, the long detection times are caused by the short turns from which only the second can drive the counter above $Tol2$ while there is about 60s between the two rotations (see Fig: 4.5). The fastest detection of all is shown in Fig. 4.4.

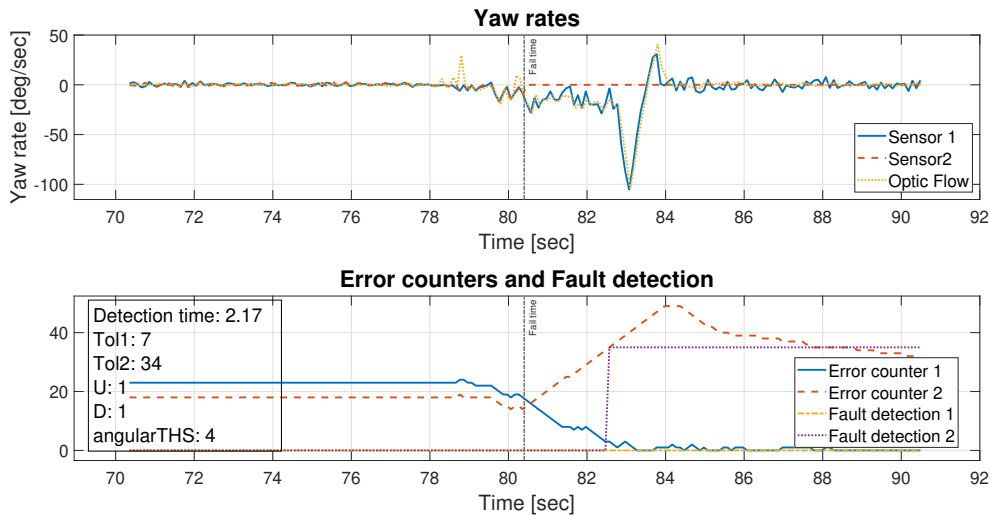


Figure 4.4: Flight1 with total sensor failure.

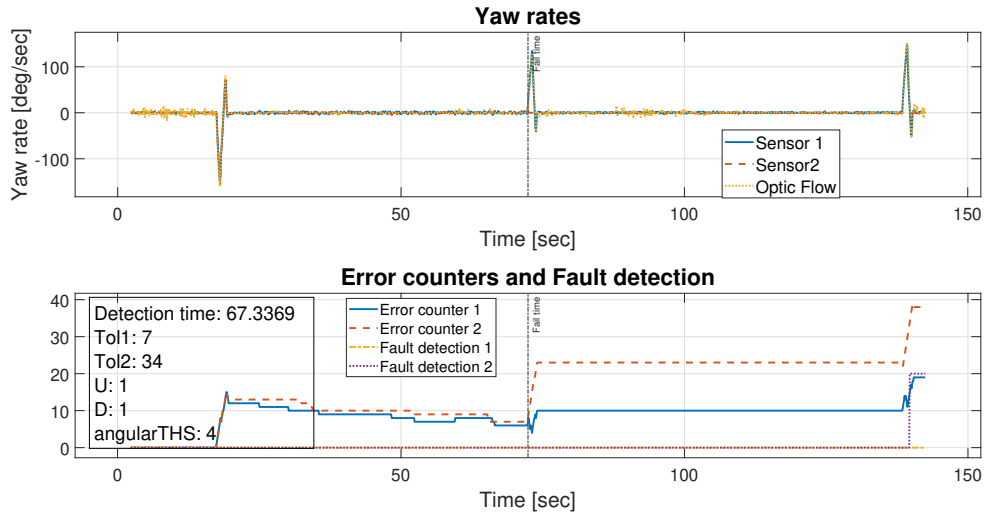


Figure 4.5: Flight2 with total sensor failure.

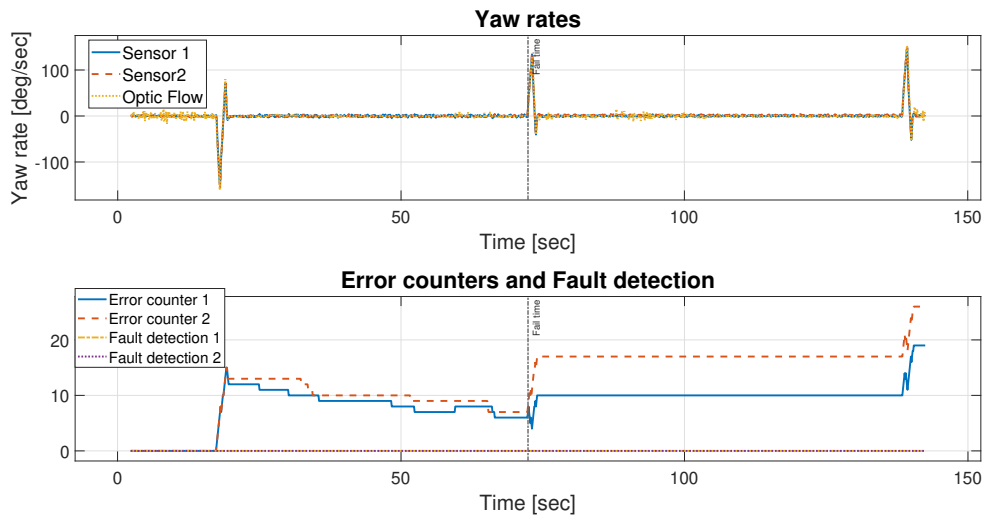


Figure 4.6: Flight2 with additive-type sensor failure.

Chapter 5

Summary

In my work I presented an optical flow based sensor fault detection method applied on real camera images and real flight data. After introducing the equations that describe the connection between the optical flow and angular rates of the vehicle, I modified the system of equations for the case of a gimbal-stabilized camera looking vertically down. After detailing the solution of the system of equations, I presented the image processing required to obtain the solution. To detect sensor errors, I applied a counter-base residual thresholding fault detection method using an error counter and threshold values, which were tuned by using the ROC curve. After that I artificially generated errors on the real flight data. With comparing these to the yaw rate that was estimated from the optical flow I was able to detect various sensor faults. The elaborated method was able to detect $A = 2.1^\circ/s$ and higher added constant values in case of additive-type sensor failure, in case of multiplicative-type sensor failure it was able to detect $P = 8\%$ and higher percentages of the measured signal but with long detection times. In this paper, I only estimated the yaw rate. The aim of the following work is to estimate all the angular velocity components and detect sensor faults, on a fixed-wing unmanned aerial vehicle with a mostly forward looking tilted camera placed ahead of the center of gravity, with using real images and real flight data. The flaw of this method that it was not able to detect small bias errors, or sensor stuck at a small value and this is planned to be solved by calculating the Euler angles from the angular rates and comparing the cumulated rotation with the image-based results.

Bibliography

- [1] Joseph Kehoe et al. “State Estimation Using Optical Flow from Parallax-Weighted Feature Tracking”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. DOI: 10.2514/6.2006-6721. eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2006-6721>. URL: <https://arc.aiaa.org/doi/abs/10.2514/6.2006-6721>.
- [2] Szabolcs Kun and Peter Bauer. “Improvements in Optical Flow-based Aircraft Partial State Estimation”. In: *2021 17th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*. Sept. 2021, pp. 1–4. DOI: 10.1109/CNNA49188.2021.9610822.
- [3] Peter Bauer and Szabolcs Kun. “Optical flow-based angular rate sensor fault detection on UAVs*”. In: *IFAC-PapersOnLine* 55.14 (2022). 11th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2022, pp. 46–51. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2022.07.581>. URL: <https://www.sciencedirect.com/science/article/pii/S2405896322009922>.
- [4] Czúni László Kató Zoltán. *Számítógépes látás [elektronikus dok.] : egyetemi tananyag / írta Kató Zoltán, Czúni László*. Typotex, 2011. URL: <http://nbn.urn.hu/N2L?urn:nbn:hu-108199>.
- [5] Péter Bauer et al. “Encounter Risk Evaluation with a Forerunner UAV”. In: *Remote Sensing* 15.6 (2023). ISSN: 2072-4292. DOI: 10.3390/rs15061512. URL: <https://www.mdpi.com/2072-4292/15/6/1512>.
- [6] G. Heredia et al. “Detection of Sensor Faults in Autonomous Helicopters”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Apr. 2005, pp. 2229–2234. DOI: 10.1109/ROBOT.2005.1570444.
- [7] Ahmet Arda Ozdemir, Peter Seiler, and Gary J. Balas. “Wind Turbine Fault Detection Using Counter-Based Residual Thresholding”. In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 8289–8294. ISSN: 1474-6670. DOI: <https://doi.org/10.3182/20110828-6-IT-1002.01758>. URL: <https://www.sciencedirect.com/science/article/pii/S1474667016449426>.

- [8] Tamas Grof and Peter Bauer. “Voting-based Fault Detection for Aircraft Position Measurements with Dissimilar Observations**This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The research presented in this paper was funded by the Higher Education Institutional Excellence Program. The research leading to these results has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 690811 and the Japan New Energy and Industrial Technology Development Organization under grant agreement No. 062600 as a part of the EU/Japan joint research project entitled ‘Validation of Integrated Safety-enhanced Intelligent flight cONTrol (VISION)’.” In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 14724–14729. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2020.12.1841>. URL: <https://www.sciencedirect.com/science/article/pii/S2405896320324551>.

List of Figures

2.1	The applied coordinate systems.	3
2.2	Projection of point m to the cameras image plane. [1]	5
2.3	k values calculated from the sensor data of Flight1	8
2.4	The Frobenius norms of (2.21) equation's both sides.	9
2.5	The k values calculated from optical flow and measured data	10
3.1	Trajectories of multicopter drone flights above ZalaZONE Smart City . . .	11
3.2	Feature points detected by <i>Shi-Tomasi</i> corner detector.	12
3.3	Feature points detected by <i>ORB</i> detector.	12
3.4	Filtered feature points.	13
3.5	Comparison of measured and estimated yaw rate of Flight1	15
3.6	Comparison of measured and estimated yaw rate of Flight2	15
4.1	Residual histograms of Flight1 with different errors. (Sensor 1 is the faulty sensor, sensor 2 is the fault free sensor)	18
4.2	FA and MD probability distributions.	18
4.3	ROC curve to find the optimal $Tol2$ value.	20
4.4	Flight1 with total sensor failure.	21
4.5	Flight2 with total sensor failure.	22
4.6	Flight2 with additive-type sensor failure.	22

List of Tables

4.1	Fault detection results.	21
-----	----------------------------------	----