Otto von Guericke University Magdeburg
Chair of Intelligent Systems
Institute of Knowledge and Language Engineering

Digital Engineering Project

Analysis of Extended Movement Models for Autonomous Quadrocopters

Anita Hrubos
anita.hrubos@st.ovgu.de

Magdeburg, 2016

Supervisor:
Steup, Christoph
steup@ivs.cs.uni-magdeburg.de

Contents

1 Introduction

The aim of the Swarm Robotics Lab of the Otto von Guericke University is to research swarm intelligence and to develop methods and algorithms in this field. To provide testing possiblities for the researcher a swarm of autonomous quadrotors were designed and built here. The current horizontal movement control of the quadrotors is, however, limited to basic movement commands such as rotation around one axis, which is not sufficient to implement swarm robotics algorithms.

The goal of this project was to analyze and implement movement models for the quadrotors, which make it possible to realize complex movement commands, including position and velocity control.

The project started with analyzing the movements of the FINken v2 quadrotor with an onboard optical flow sensor and it moved later on to the currently developed FINken v3, which was a more stable and robust construction.

This documentation contains the analysis I made, including the testing procedure and evaluation. In chapter 2 I summarize the theoretical background needed for the project, then in chapter 3 the applied sensors are described. The details about the implementation of the movement models can be found in chapter 4 and chapter 5 contains the evaluation of the implemented models.

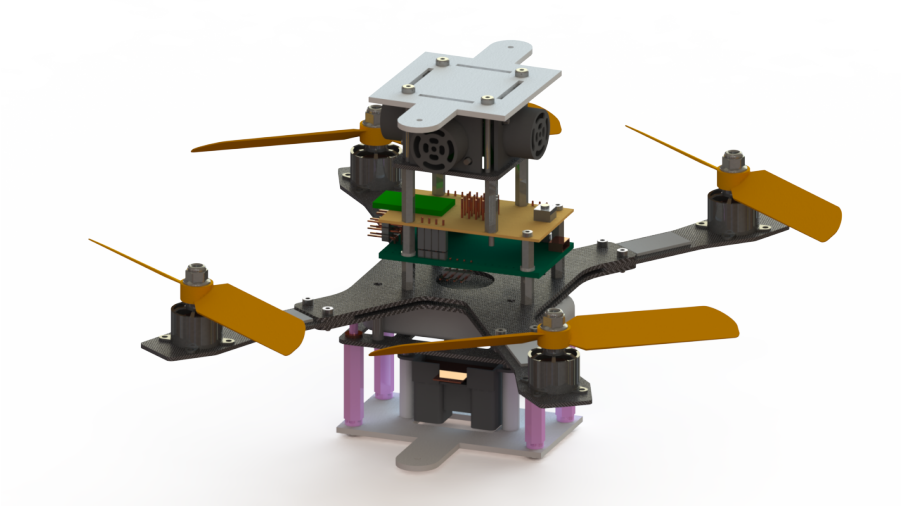


Figure 1.1: FINken v2 with IR sensor

2 Theory

In this chapter the theory needed for the project is summarized. The first section describes the basic concept and applicability of a PID controller, which is one of the most widely applied control algorithms in the field of unmanned aerial vechicles (UAV).

The second section is a short summary about trajectory control, which is the ultimate goal of movement analysis and control projects, since trajectory control allows complex and autonomous movements, which are requisite for swarm algorithms.

2.1 PID controller

The PID controller (proportional-integral-deriative controller) is a simple control loop feedback mechanism.

The control signal (u) is calculated based on the error signal (e), which is the difference between the desired value (r) and the actual output (y). Depending on the type of controller, proportional (K_p), integral (K_i) and/or derivative (K_d) gains can be used. The formula for the control signal is the following:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt + K_d \cdot \frac{de}{dt} \tag{2.1}$$

$$e(t) = r(t) - y(t) \tag{2.2}$$

This control signal is sent to the plant and the new output is measured, based on which the new control signal is calculated.

Increasing the proportional gain reduces the rise time, yet it might result in a huge overshoot. The overshoot can be reduced by adding a K_d gain to the controller. P or PD controllers may have, however, a significant steady-state error, which can be eliminated using an K_i gain. Its disadvante is that it may increase the response time and cause instability.

To realize the velocity and the position control two PD controllers were implemented. The aim was to control the quadrotor when it was in motion and not during hovering. Therefore the K_i gain was not included because it could have eliminated only the steady-state error, which was pointless considering that the quadrotor was permamently in motion. Yet, if the aim is to stabilize the hovering, it would be worth applying a PID controller.

2.2 Trajectory control

For many applications trajectory planning is necessary. This problem includes two parts: trajectory generation and the control of a quadrotor to follow the predefined path.

Trajectory control requires not only reliable velocity and position feedbacks, but also the quadrotor has to be able to follow the control signals quite well. As far as the microcontroller is considered, it must be powerful, since the on-the-fly calculation of the trajectories requires great computational power.

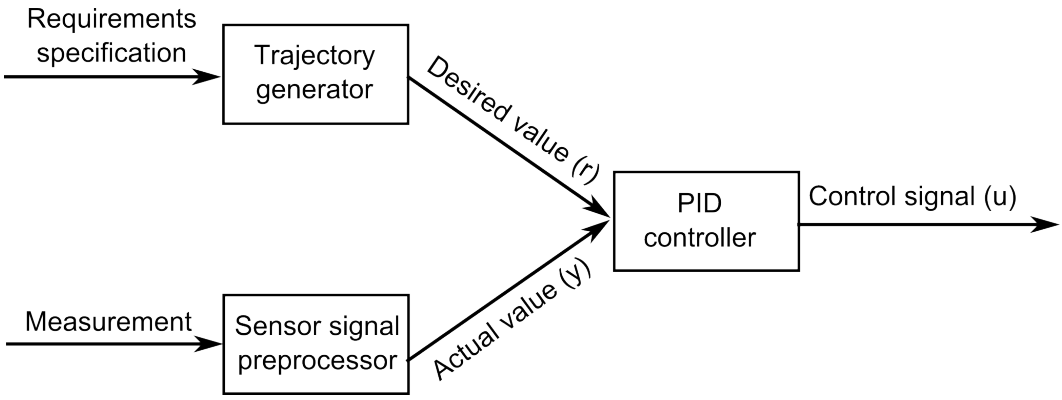


Figure 2.1: General concept for trajectory control

2.2.1 Trajectory generation

The aim of the trajectory generation is to find the most optimal path from point A to point B. To guarantee feasibility of this trajectory some conditions depending on the task and hardware should be satisfied. For example the maximal actuator torque and velocity calculated should not exceed the limitations of the hardware. Furthermore, the optimal trajectory should satisfy the predefined boundary conditions, such as zero velocity at point A and B. After the trajectory is generated, it should be tested whether it is collision-free, especially if there are more (moving) objects in the arena.

2.2.2 Following trajectories

Various strategies exist, which makes it possible to follow a predefined trajectory. In most applications PID controllers or linear quadratic regulators (LQR) are used. The mathematical description of the PID controller was intruced above.

The linear quadratic regulator (LQR) is a feedback controller for systems that can be described by linear differential equations. The control is realised by minimizing the quadratic cost function, which is ususally the sum of the difference between the set point and the measured values. In this way, the deviation from the desired state can be minimized.

However, there are some other methods. For example, [5] uses a predictive control strategy. The basic idea is to predict the only next n future states based on the current state to find the optimal sequence of commands. Then, the first command is applied and the calculation is repeated for the next time step. Optimality is reached by minimizing a cost function, which can be extended to fulfill some other requirements, such as less aggressive flight or slower attitude changes.

3 Sensors

This chapter is a short summary about the relevant on-board sensors which were available during the project and were used to gain feedback about the state of the quadrotor.

To desing and implement motion control and analysis the following data has to be measured and monitored during the flight:

- angular position (roll, pitch and yaw)
- acceleration in directions x and y
- velocity in directions x and y
- position in directions x and y
- altitude (position in direction z)

The angular position and the acceleration were measured by the intertial measurement unit (see section 3.1). To obtain the altitude an IR sensor was mounted to the ground plate by default and there were no sensors for velocity or position measurement in directions x and y. To solve this problem the IR sensor had to be replaced with an optical flow sensor (see section 3.2), which contained a sonar sensor to obtain altitude data.

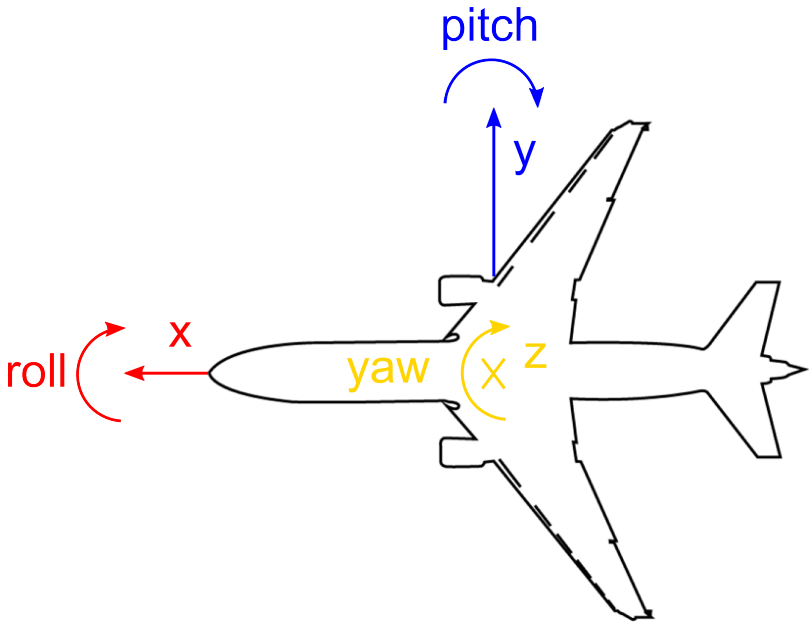


Figure 3.1: Body coordinate system

3.1 Inertial Measurement Unit (IMU)

The IMU is a combination of sensors to measure the angular rate of the aircraft body and the forces acting upon it. It is typically used in unmanned aerial vechicles to track motion and to support navigation.

The quadrotor hardware is equipped with an Aspirin IMU, which has a 3-axis accelerometer,

a gyroscope and a magnetometer on board. This makes it possible to get the acceleration and the angular position state of the quadrotor in 3D space.

3.2 Optical flow sensor

In the project the PX4FLOW optical flow sensor was applied. It estimates the motion of the quadrotor based on sequence of images from the camera. The velocity is derived from the displacement which is calculated based on the displacement of the patterns in the consecutive image frames. The difficulty of the estimation lies in the tracking and matching of the patterns.

Before applying the flow sensor, the following aspects should be checked. First, the focus of the image should be set sharp at the operating distance, which is the set point value of the altitude controller in this project. Otherwise the image is not clear at most of the flight and the sensor values are therefore not good enough. The focus setting can be done using a program called QGroundcontrol[1].

Additionally, the parameters of the flow sensor should be set adequately to the environment. They can be changed using the same Qgroundcontrol program, however, it is important to note that in this way the settings are not written to the ROM, thus they are resetted to the default values at power loss or restart. The solution to this problem is to change the hard coded parameters in the code and re-flash the sensor. In table 3.1 the most important parameters are listed.

Table 3.1: Summary of the most important parameters of the flow sensor

Parameter	**Default value**	**Value applied**	**Note**
LENS_FOCAL_LEN	16	4.7	Focal length of the lens (mm). If they are not set according to the specifications, there will be a constant scaling error.
IMAGE_L_LIGHT	0	0	Low light mode. If the ambient light conditions are poor, it should be set to 1.
BFLOW_GYRO_COM	1	1	Gyro compensation mode.
SONAR_FILTERED	0	0	Kalman filter on the altitude values measured the by sonar. According to tests, the results are much better when the filter is off.

[1]downloaded from the https://pixhawk.org/modules/px4flow website

4 Implementation

The code with comments can be found online at `https://github.com/ovgu-FINken/paparazzi.git` in the modul pos_control under the branch called anita-velocity-control.

To use the position control module the USE_FLOW and POS_CONTROL flags have to be defined in the configuration file under conf/airframes/ovgu. To avoid interference with other modules the following modules have be turned off when using the position controller:

- finken_wall_avoid
- finken_rc_passthrough

4.1 Coordinate systems

To avoid the misinterpretation of the sensor data, their coordinate system had to be always checked. Most of the sensor data was obtained in the coordinate system of the sensor and was not automatically transformed into the body coordinate system.

At first, the aim was to transform everything into the North-East-Down (NED) coordinate system, however, this was not possible because of the incorrectness and drift of the yaw angle measurement. Therefore, the yaw angle was manually set to 0 in the code and it was assumed that the quadrotor does not rotate around its z axis during the flight. But this assumption was not valid and because of this incorrectness in the behaviour of the quadcopter, the motion state estimation was inaccurately when the yaw angle changed during the flight.

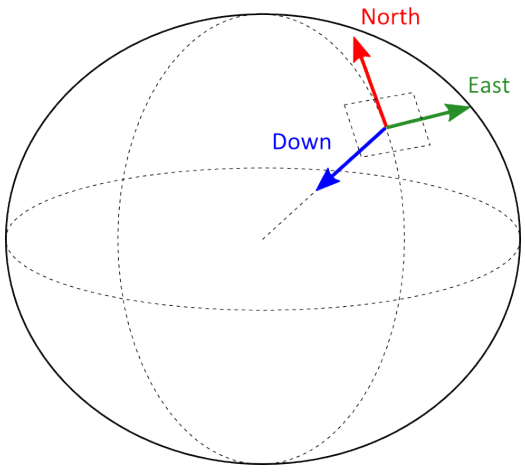


Figure 4.1: NED coordinate system

To overcome this issue, the acceleration and velocity values were transformed into the body coordinate system of the quadrotor (positive x axis pointing forward, positive y axis pointing to the right), which moved with the quadrotor and rotated with it. Of course, a global coordinate system is needed to track the position. This global coordinate system was initialized at the system start. All position values obtained during the flight were relative to this global coordinate system.

4.2 Acceleration

The data from the sensors are processed by different subsystems and there are various ways to obtain the measurement values from Paparazzi.

At the beginning, the accelerations computed by the inertial navigation system (INS) were used. The INS fused the sensor data and calculated not only the acceleration, but also the position, orientation and velocity of the UAV. It processed the raw sensor data from IMU in function ins_int_propagate(). First, the data was transformed from the sensor coordinate system into the local coordinate system of the quadcopter and than into the global NED coordinate system. A compensation to gravity in direction z was implemented, this means that 9.81 m/s^2 was added to the sensor value.

The measurements showed, however, that the acceleration data from the INS subsytem was so noisy that it could not be used directly. Therefore, some filtering methods were applied, including the following moving averaging filter:

$$\overline{a}_n = \frac{(n-1) \cdot \overline{a}_{n-1} + a_n}{n} \tag{4.1}$$

where,

$\overline{a}_n$: average value
a_n: actual value

Various weights were examined for the filter; however, the results were not accurate enough. If the weight was small, the acceleration was still noisy. Increasing the weight, however, introduced an inacceptable time delay in the sensor output.

The tests showed that the noise was partly due to the coordinate transformation by the INS using the noisy yaw data. To overcome this problem the raw sensor data was used later on. This means that the coordinate transformation from the sensor coordinate system into the local coordinate system of the quadcopter had to be implemented, too. Due to the rational mounting it could be easily done in the FINken sensor model:

```
acceleration.x = imu.accel.y;
acceleration.y = -imu.accel.x;
```

Alternatively, the acceleration values estimated by the Kalman filter, which was implemented by David Bodnar, could be used. One problem with it is the maximum achievable frequency of the Kalman filter. Using the current hardware it is only 15 Hz, which might not be enough, depending on the requirements of the application.

4.3 Velocity

To control the velocity of the quadrotor it was essential to have feedback about the actual velocity. Theoretically, it could have been integrated from the acceleration, which was provided by the accelerometer of the IMU, however, the acceleration measurement itself was too noisy. Therefore, it was obtained from the optical flow sensor, which was designed to measure the velocity.

However, for the optical flow calculataion the pattern of the ground was a vital point. To check the visibility of the patterns on the floor the QgroundControl program was used. This application made it possible to download the live video stream from the camera, so the surfaces could be evaluated. During the project various options were tested.

The first tests were performed on the original black mattresses with white covers. The results were fair, but the propellers stirred the air, which could move the covers and indicate incorrect velocities.

The results of the test with the new yellow mattresses were unsuprisingly useless, since they were monochromatic and they had no patterns. The sensor values were constant zeros.

Then the mattresses were covered with white plates with black tapes on them. The contrast between the tapes and the surfaces was high, therefore it was easy for the program to recognize and track the patterns, however, the patterns were too far away from each other and there were too large white spots between them, where the sensor values were zero. The reason for that was the field of view of the lens, which is relative small: it is 83.1°. The conclusion was that the plates cannot be used without further modifications.

The best indoor results were reached with a patterned blanket (see 4.2). It has to be noted that the air stirred by the rotors moved also these blankets a bit and they were not large enough to cover the whole surface of the arena, which reduced the reliability of the measurement.

Figure 4.2: Patterned blanket

In addition to the indoor tests, the optical flow sensor was tested also outdoor. The tiled pavement in the university seemed to be an excellent surface, where the sensor provided good and reasonable velocity values.

Besides the well-recognizable patterns of the surface, good light conditions were necessary, too. According to the indoor tests, the ambient light was not enough in normal light mode (see table 3.1), but the lamps above the arena provided enough light, therefore no additional light

sources were needed, but the lamps had to be always on.

Not only the environment conditions should be met, but the style of the flight should also be taken into consideration. Aggressive flight with rapid angular position changes should be avoided. The reason is that the image seen by the flow sensor shifted also by angular rotations even if the quadrotor did not move. There was a gyroscope compensation mode (see 3.1) in the optical flow sensor, however, it could not filter out each peaks caused by rotation. Also fast vertical movements could cause problems because the software of the flow sensor did not take the scaling of the images into consideration.

4.4 Position

The position of the quadrotor was integrated from the velocity measured by the optical flow sensor. Many numerical methods for precise integration exist; however, due to the limited computational power of the quadrotor the Euler method was applied:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{4.2}$$

where,

y_{n+1}: actual estimation of the position
y_n: estimation of the position in the last step
h: step size
t_n: actual time step

The greatest advantage of this method was its simplicity. On the other hand, its error was relative large. More complex integration methods were also implemented and tested, however, due to the noisy velocity measurement, they could not improve the results significantly.

The solution for the inaccurate position feedback could be to use a position sensor or method, which makes it possible to calculate the position correctly. There are some possible alternative solutions to this problem:

- external camera system to track the quadrotor
- GPS (especially for outdoor applications)
- camera to track markers (e.g. with QR code) indicating absolute position

4.5 Angular position

The measurement of the pitch and roll angles was quite accurate because Paparazzi compensated the measurements using the acceleration data. The main idea is to use the projection of the gravity vector to calculate the tilt angles. Unfortunately, the compensation of the yaw angle is not possible in this way.

In figure 4.3 a test measurement can be seen when the quadrotor stayed on the table with the rotors on. The compensation mentioned above kept the differences in the pitch and roll values within 1°, which made these measurements reliable. On the other hand, there was a significant drift in the yaw value, during this test it was more than 4° in 75 s. According to further measurements, this drift was not constant or deterministic which means that it could not be compensated without any further information.

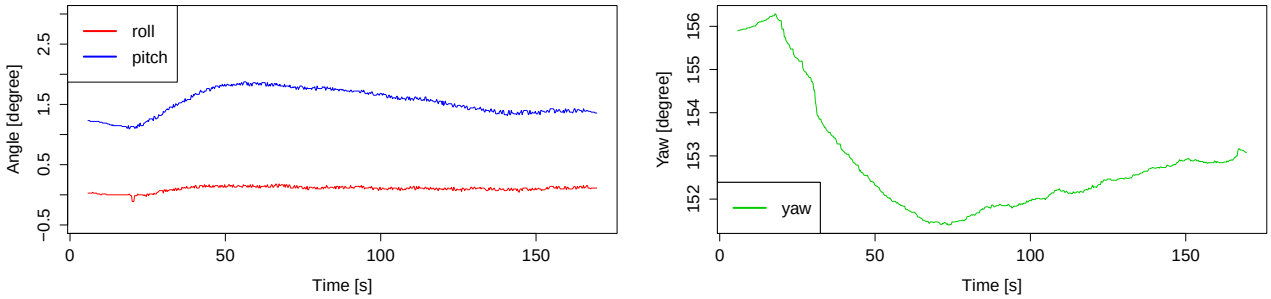


Figure 4.3: Test measurement for the angular position

Another problem is that the angles might have an offset error. They could be eliminated using an exact calibration, however, it has to be mentioned that the offset is not exactly the same during each flight and after some reconstruction (for example after replacement of the IR sensor with the flow sensor) it might change even more drastically. These factors make the calibration difficult.

Before carrying out this measurement, the calibration for the pitch and roll angles were carried out. The result of the roll angle calibration was satisfying, however the offset in the pitch angle could not be eliminated (see section 5.1).

4.6 Control loops in Paparazzi

Paparazzi provides four built-in horizontal control modes:

- GUIDANCE_H_MODE_RATE: using the rate control loop it calculates the roll, pitch and yaw commands in a way that the set point rate values are met
- GUIDANCE_H_MODE_ATTITUDE: using the attitude control loop it stabilizes the three angular positions
- GUIDANCE_H_MODE_HOVER: makes the quadrotor hover at the position of the initialization
- GUIDANCE_H_MODE_NAV: navigates the quadrotor to the set point coordinates in direction x and y

The hover and the navigation mode match partially the goal of the project, so they could have been applied. However, they all use the NED coordinate system, which is a global one. This means, that the values are always transformed using the information about the heading (yaw angle).

As shown and described above, the yaw angle measurement is not reliable in its actual condition. It was not worth rewriting the coordinate transformation and the data aquistion of Paparazzi, because it was not well-documented and the implementation of an own PD controller did not require much effort. The challange was to find the correct parameters and to provide accurate feedback for the controller.

4.7 Controller

In Paparazzi various horizontal control modes can be chosen. The mode „GUIDANCE_H_MODE_ATTITUDE" is ideal in most of the cases because the angles can be set precisely with it. It has some advantages, such as it is easily understandable and verifiable by the user. For the position control of the quadrotor it is adequate to modulate the angular position (pitch/roll). Let us see the following case as an example. If the quadrotor is on the left of the set point destination, it should be moved to the right. In other words, it should lean to the right, that is, the roll angle should be positive. A linear relationship between angular position and the position error can be defined, it is the K_p gain of the position controller.

However, the velocity error cannot be linked to the angular position without modification. Let us see an example. Let us assume that the quadrotor flies to the right with a velocity of 1.5 m/s instead of 1 m/s. This means that the velocity should be decreased. It would be a false reaction to set a negative roll angle. Instead, the roll angle should be decreased and the degree of the reduction should be the K_p gain of the velocity controller.

For this purpose, a control mode should be chosen, where the set point values are the angular rates and not the angular positions. In Paparazzi it could have been done by selecting the „GUIDANCE_H_MODE_RATE" horizontal control mode. To avoid such a significant alternation in the low level, the velocity PD controller was modified, instead. The new control signal was not equal to the current error multiplied by the gains, but it was always incremented with it:

$$u(t) = u(t-1) + K_p \cdot e(t) + K_d \cdot \frac{de}{dt} \tag{4.3}$$

4.7.1 Tuning

Different level of autonomy can be achieved with UAVs. They can be semi/fully-autonomous or manually controlled. The FINken v2 and v3 quadrotors can also be controlled using a remote control, which has 4 sticks, thus the four degrees of freedom can be individually controlled. However, the aim of this project was that the quadrotor could follow predefined trajectories autonomously.

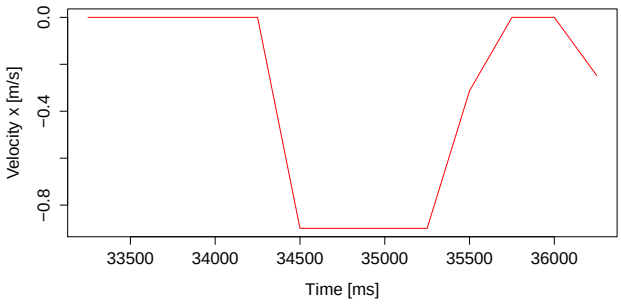
Paparazzi made it possible to easily pass through the signals from the remote control in autonomous mode, so semi-autonomous control could be achieved. This was a great facility which I used while testing. For the directions x and y the gains could be tested individually. To achieve this the altitude-controller and the controller only for one direction were turned on. In this way it was easier to keep the quadrotor in the arena and to avoid walls. Alternatively, the wall avoidance program could have been turned on in one direction.

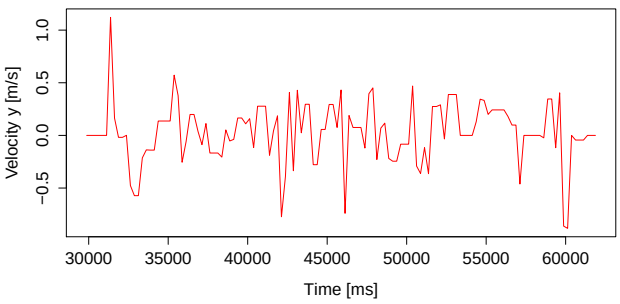
5 Evaluation

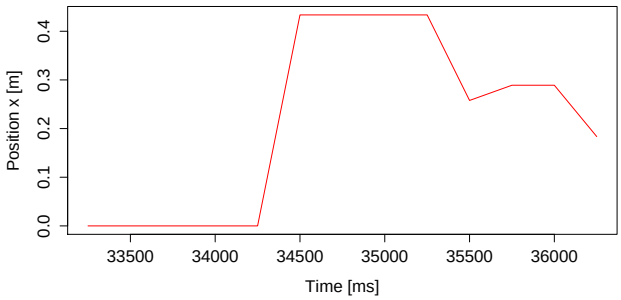
5.1 Position control

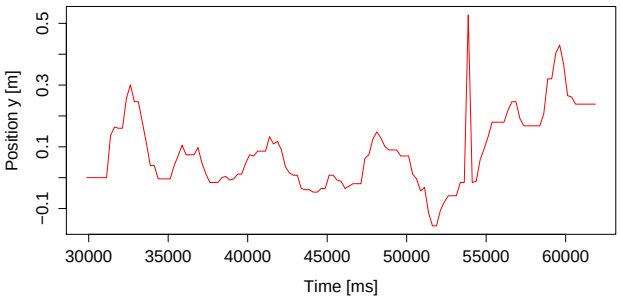
In the following tests the set point positions in both directions were set to 0 m. All tests were performed with the following calibration parameters:

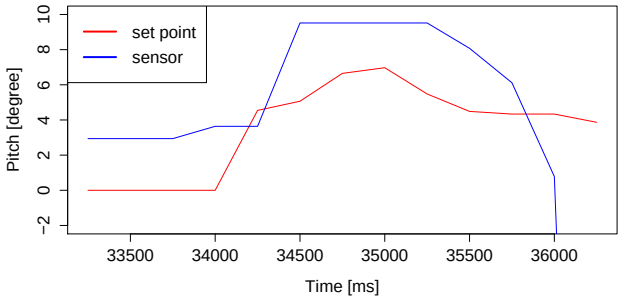
- roll offset: 2°
- pitch offset: -1°
- yaw offset: 0°











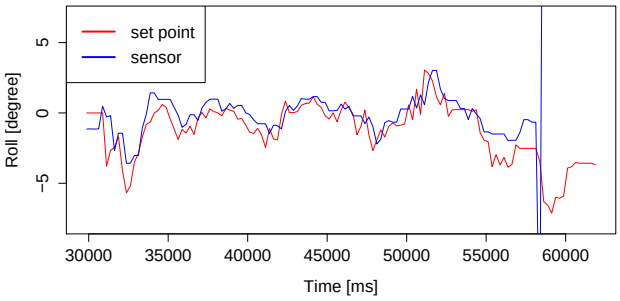


Figure 5.1: Measurement 1

Figure 5.2: Measurement 2

The parameters of the position controller in each test:

- K_p gain in direction x and y: 15
- K_d gain in direction x and y: 1.6

In figure 5.1 it can be seen that the position values became often unreliable at the take-off. The reason was that the optical flow sensor might incorrectly detect horizontal motion by the vertical movements, since the software was not designed to compensate the affects of the drastical scaling or rotation of the image by vertical movements. To compensate this incorrectly estimated initial position of 0.4 m, the quadrotor moved backwards (positive pitch), crashed into the back wall and flipped.

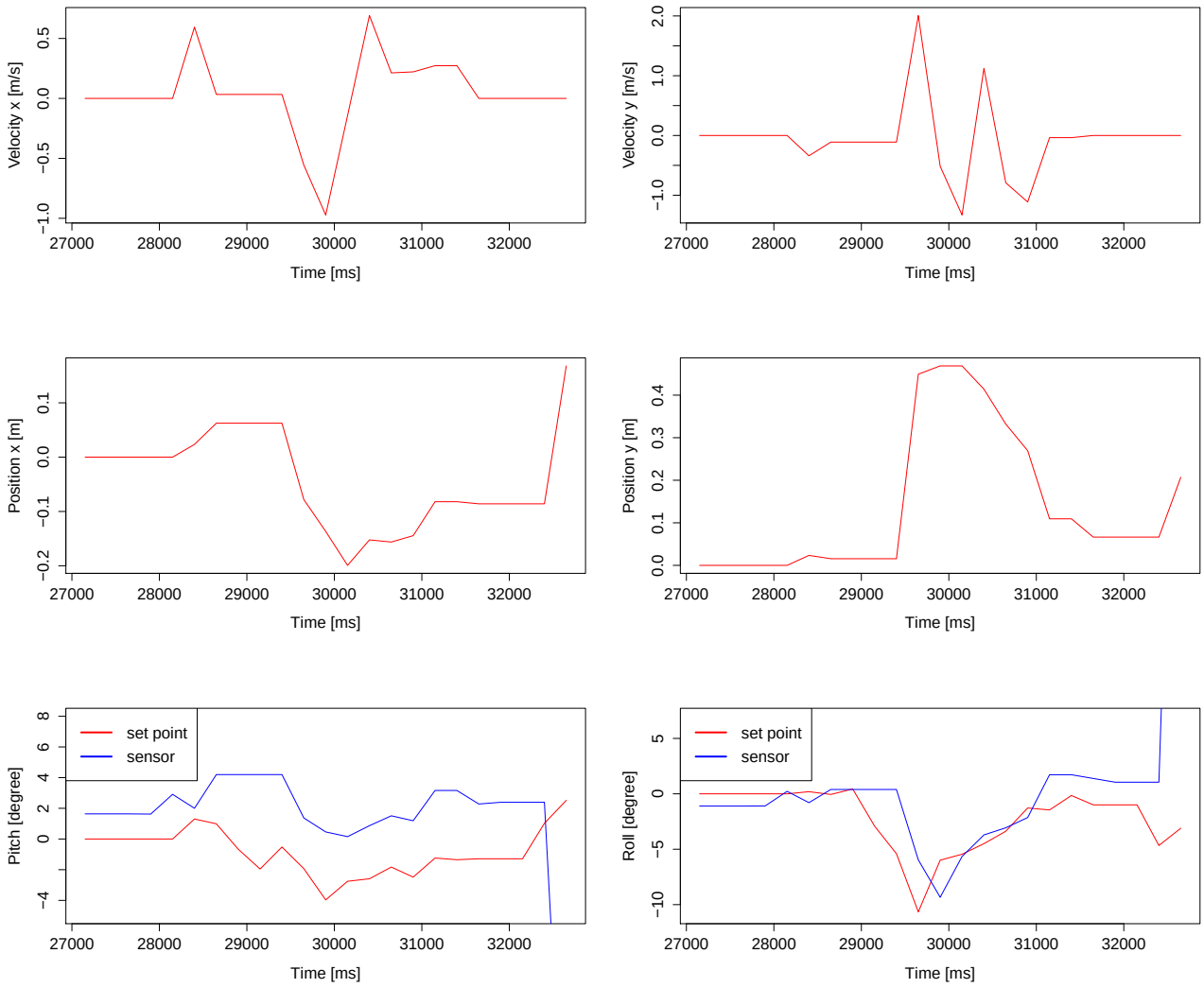


Figure 5.3: Measurement 3

Accoring to the position estimation the quadrotor moved about 0.2 m before it reached the wall and flipped. However, the real displacement was about 1 m. This scaling error was mainly caused by two things. First, the blanket was not large enough to cover the whole arena, thus near the walls there were blind spots (about 0.2 – 0.3 m) where the velocity measurement was not working. Secondly, in many cases the flow sensor did not recognize the horizontal motion and returned zero value.

In the test presented in figures 5.2 a similar example can be observed. During take-off, the quadrotor measured a displacement of about 0.3 m and tried to compensate it, so it moved to the left. The actual displacement until the crash was about 1.5 m to the left.

It is worth checking here how the angular positions differ from the set point values. During the initialization there was a constant offset due to the non-flatness of the ground surface and the error in the calibration. At the take-off the built-in controller compensated and after an overshoot the actual values followed the set point values properly. This was true only in direction y. In direction x there was an about $2 - 3°$ offset between the set point values and the sensor data. To test the effect of the calibration, the measurements were repeated after resetting the calibration values for roll, pitch and yaw to 0. One of the tests can be seen in figure 5.6 which shows that the offset is unchanged, thus it was independent from the calibration parameters. This strange behaviour was present both in FINken v2 and v3, which suggest that the problem was not caused by the sensor or the hardware, but it lied in the Paparazzi software.

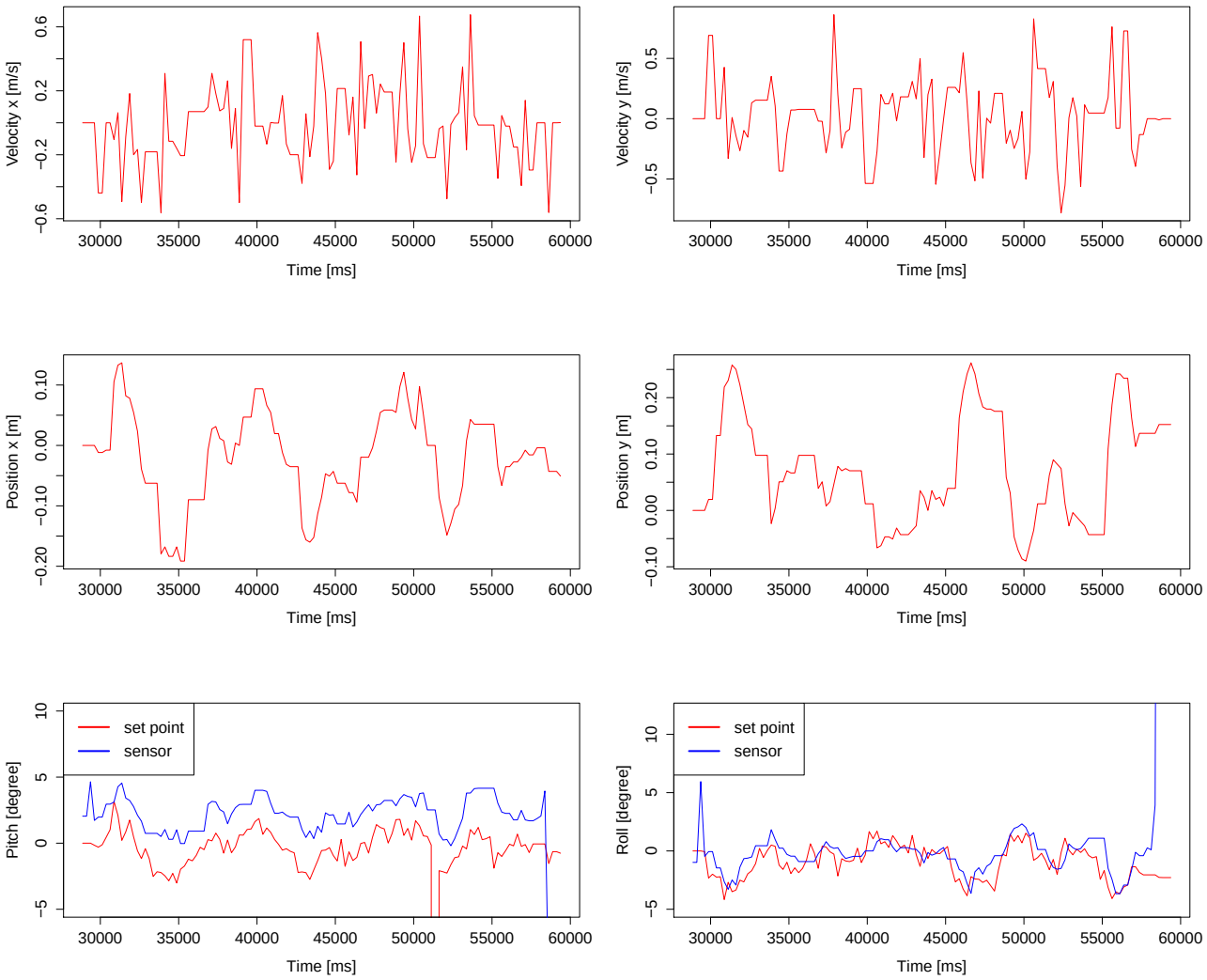


Figure 5.4: Measurement 4

Moreover, the first pitch value right after the start confirmed that there was some problem with this angle. In each measurements the quadrotor compensated the difference between the set point value and the actual observation value, so the roll value always moved toward 0, which was the initial set point. However, exactly the opposite phenomenon could be observed in the pitch angle. The first alternation always increased the difference between the set point and the measurement values. This could be observed in each measurement.

Another problem was that the quadrotor often rotated around its z axis by +45° right after start (during take-off). This happend for example in measurement 3. This might be another reason why the quadrotor measured incorrectly a velocity of 0.5 m/s during take-off, which caused an incorrect displacement, similarly to measurement 1.

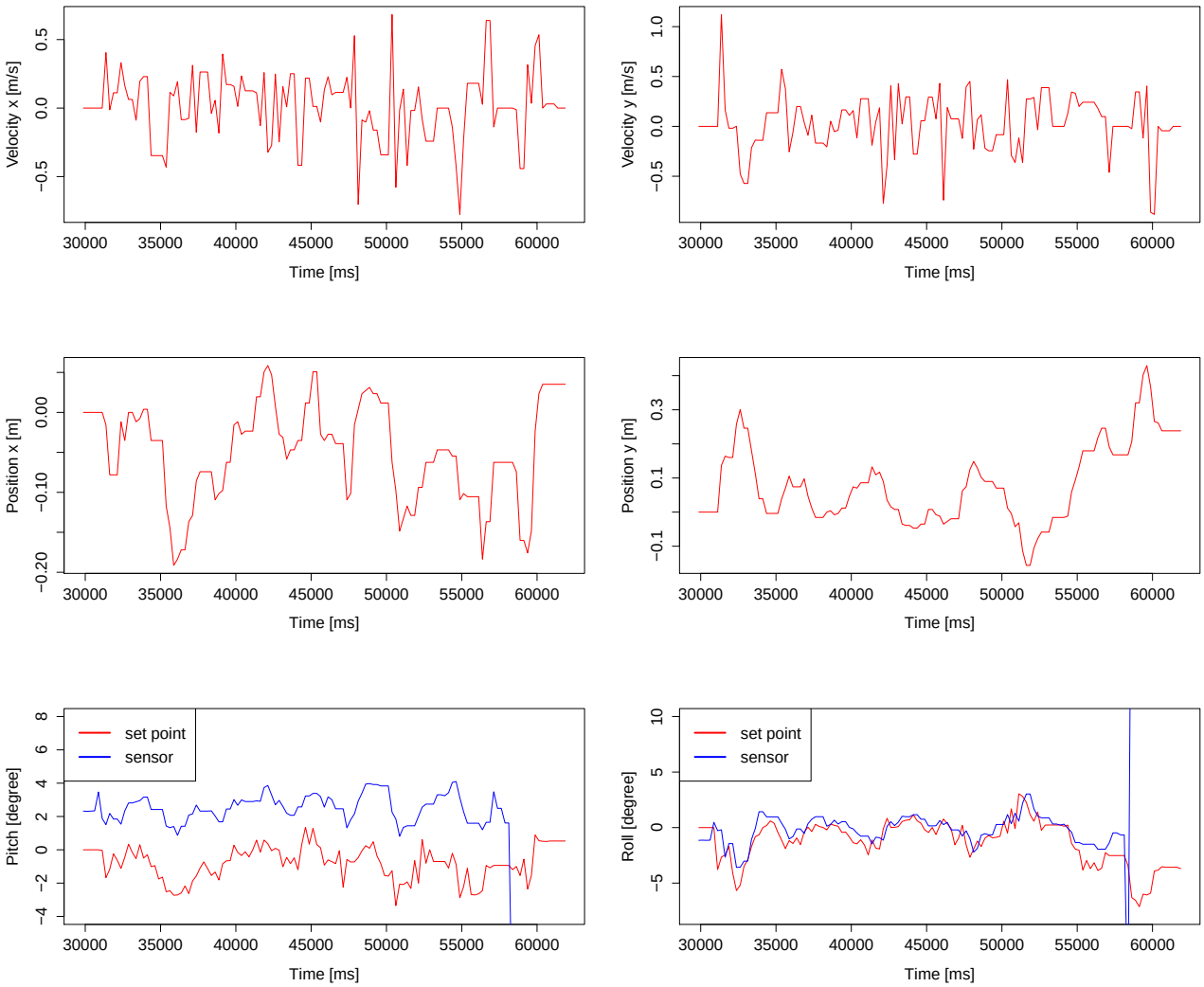


Figure 5.5: Measurement 5

Figure 5.4 shows one of the rare cases when the behaviour of the quadrotor answered the expectations. Both in direction x and y the position measured was around 0 m. However, the actual position was not so satisfying because of two factors. First, the actual displacement was about 1 – 1.5 m. Secondly, the quadrotor rotated around its axis z with about +60° during the flight. Therefore, in the global coordinate system the actual movement was not around the zero position.

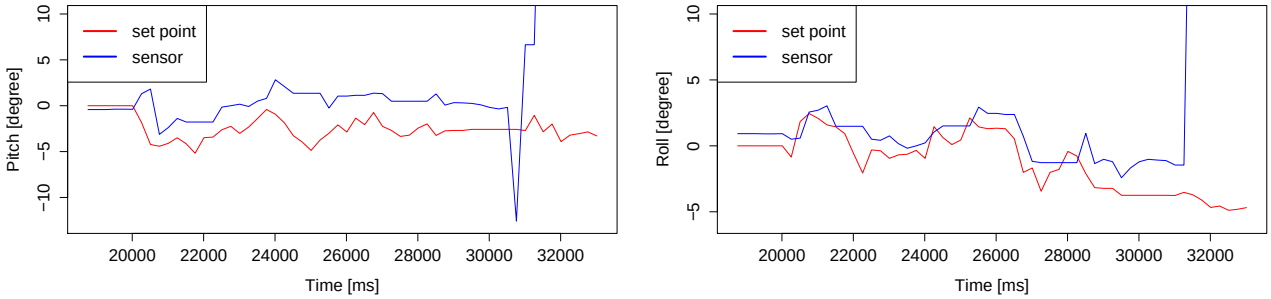


Figure 5.6: Test with calibration values of 0

In figures 5.5 another functioning example similar to the previous one can be seen. However, it is worth checking the behaviour in direction x in both cases. The offset and the receding initial values mentioned earlier were present also in these measurements.

5.2 Velocity control

During the evaluation of the velocity controller, the set point velocities were set to 0 m/s, so the quadrotor was expected to hover. To be able to evaluate larger velocities that 0, the arena should be enlarged or the tests should be carried out outside.

The PD velocity controller was tested using both the acceleration and the velocity change for the calculation of the K_d gain. The sixth measurement (see figure 5.7) shows the latter.

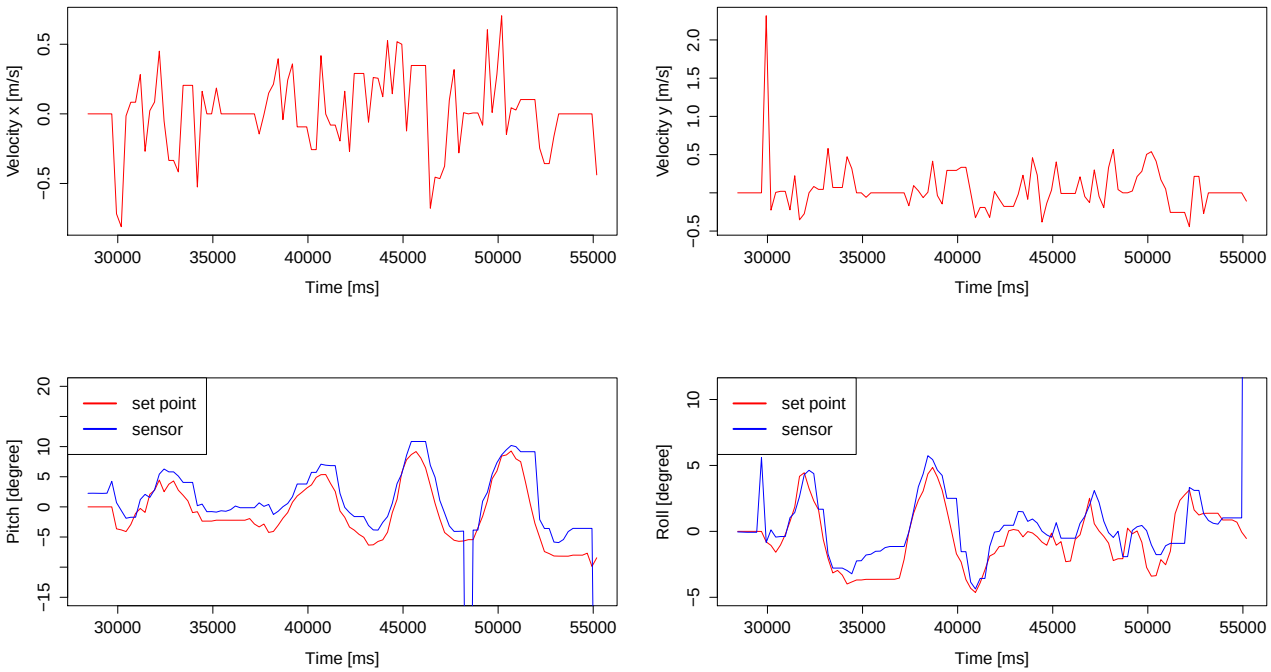


Figure 5.7: Measurement 6

Right after the start the flow sensor measured incorrectly a velocity of more than 2 m/s. After that the quadrotor started flying in a circle and after the 43rd second it flew forwards and backwards twice. In this measurement the yaw angle was quite stable and thereofore the

measurement values reflected the real situation quite well. However, the controller itself could not control the quadrotor in a stable way, the gains ($K_p = 0.8$ and $K_d = 0.01$) were not adequate which caused overshooting.

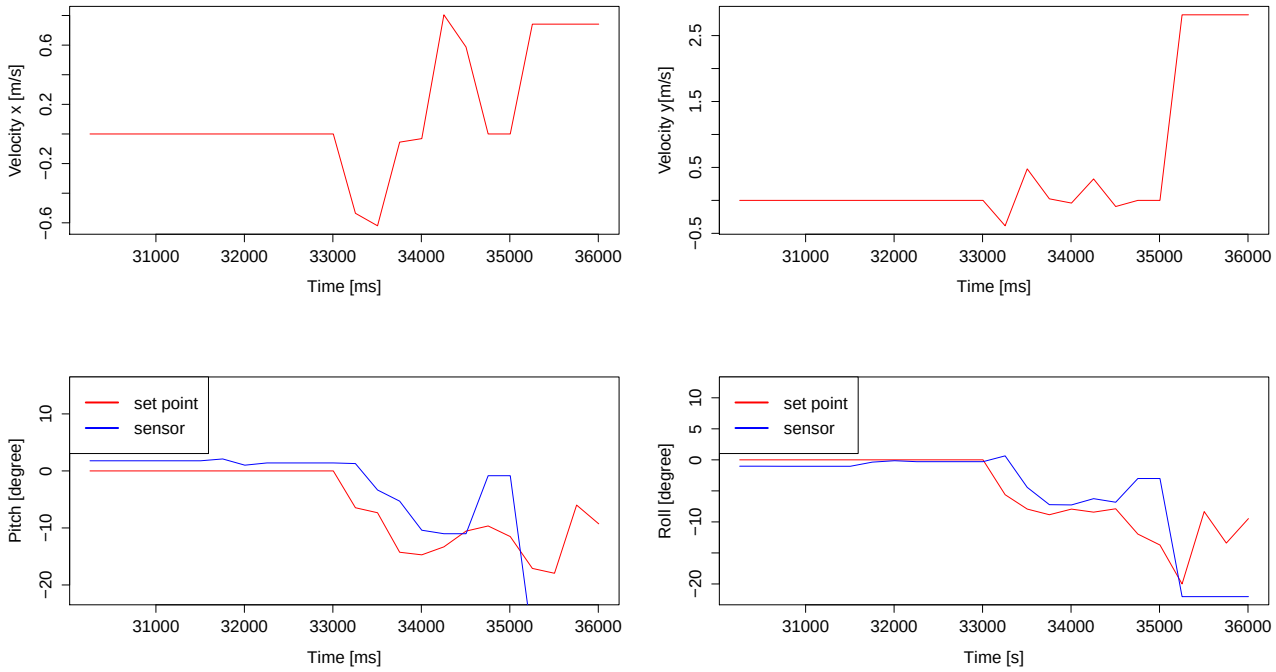


Figure 5.8: Measurement 7

In the seventh measurement the output of the flow sensor was far worse than in the previous case. In reality the quadrotor turned +45° during take-off than flew into the left wall with an increasing velocity. The reason is that the flow sensor measurement was too noisy and the gains based on the unreasable values were always added to the control output (integral behaviour see in section 4.7) that caused acceleration in negative y direction.

This example shows that in case of a faulty velocity measurement the error is integrated and amplified. Because of this, it was decided not to use the velocity change for the calculation of the K_d gain. Instead, the acceleration measurement values directly from the IMU were applied. The main advantage was that they were present even if the flow sensor could not measure anything. On the other hand, the acceleration measrement itself was not free from noise.

In the eighth measurement (see figure 5.9) the accelerations were used to calculate the control output. Right after take-off, an incorrect velocity of -1.5 m/s appeared on the sensor output, which caused the quadrotor to move forward and collide with the front wall. Due to the integrating behaviour the pitch angle could not be compensated in time.

5.3 Altitude measurement

One of the consequencies of using the flow sensor was to have a sonar sensor for altitude measurement instead of the IR sensor.

The quadrotor applied other sonar sensors, which were for measuring the distance to the walls of the areana. Tests showed that the sonar for the altitude measurement was not affected by

the other sonar sensors. However, the wall avoidance application should also be tested using the sonar of the flow sensor to see whether it is incompatible or interferes with the sonar tower.

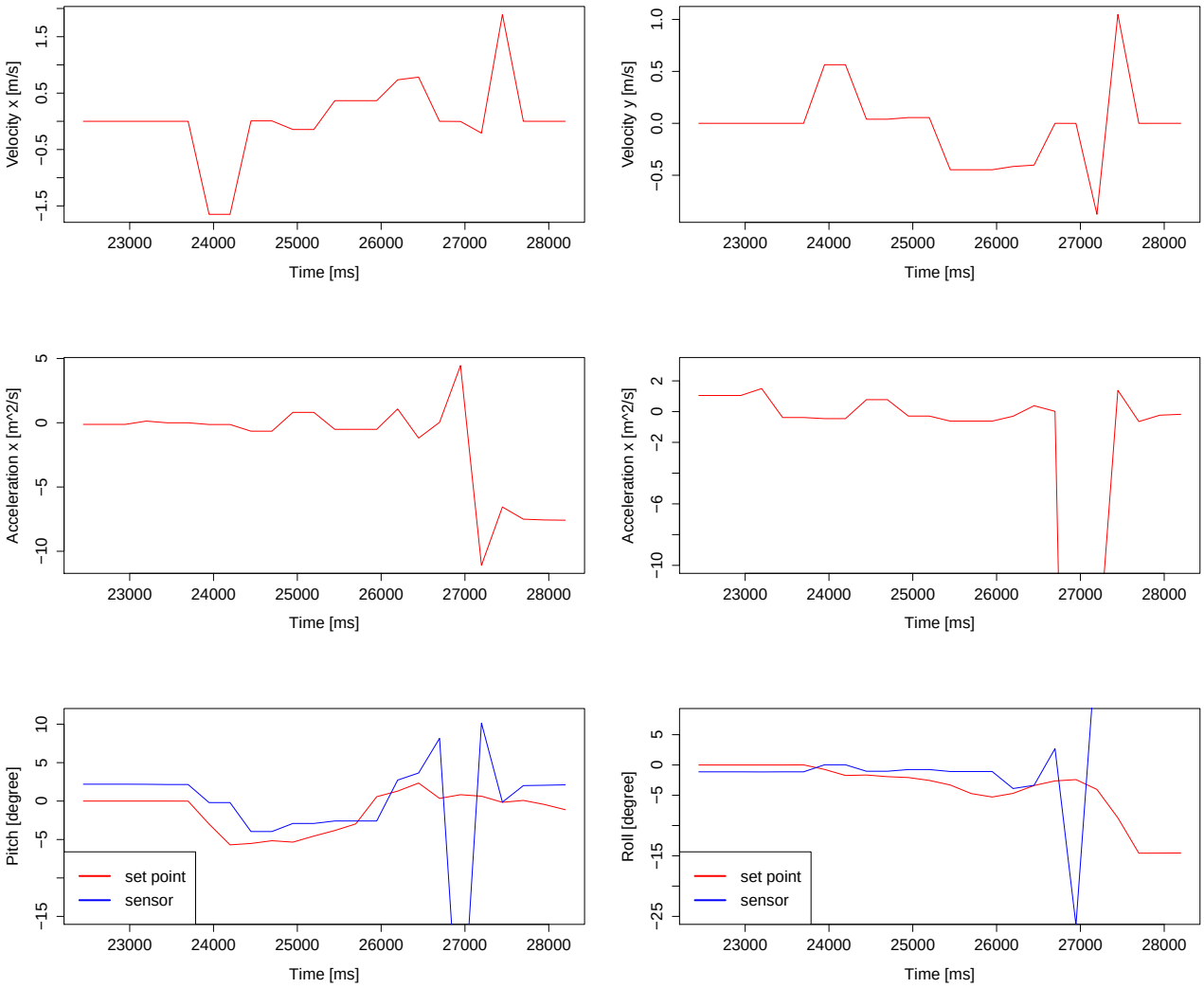


Figure 5.9: Measurement 8

The flow sensor was used both with the FINken v2 and v3. Due to the differences in the structure of the quadrotors, the behaviour of the altitude sonar sensor was different in both cases.

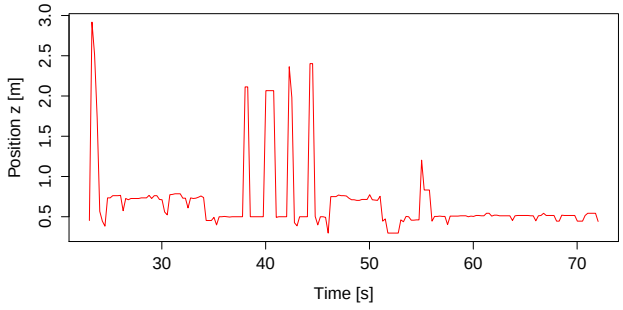


Figure 5.10: Altitude test

On the FINken v2 the sonar sensor provided far better results. After initialization on the ground, the altitude measurement showed 0.29 m and there were only few outliers. In contrast, the output of the sonar sensor on the FINken v3 measured more than 1 m several times after initialization and the results got better only after it was lifted. This suggests that the flow sensor has some filters on-board which cannot be modified through the QGroundControl program. As described before, the flow sensor offeres a Kalman filter on the altitude output, which made the test results even worse.

In figure 5.10 an altitude test could be seen with the FINken v3. In the first 46 seconds the quadrotor was on the table. After that it was moved above the floor, so the actual altitude changed from about 0.1 m to about 0.7 m. At time 52 s the quadrotor was returned to the table and at 55 s it was lifted above the floor again. Then it was placed back onto the table. It can be concluded that the sonar could precisely measure the altitude when it was not on the ground. However, after initialization there were a lot of outliers and even later the altitude output was 0.59 m, which was worse than the 0.29 m measured with the FINken v2.

However, if larger distance bolts were applied at the ground plate, the behaviour of the sensor got better, which suggests that the reason for this behaviour must be the structure and the ground plate of the quadrotor. The suggestion is to redesign the holes of the ground plate considering the reflections from the sonar sensor.

6 Conclusion

The controllers could not work as they should have in most of the cases: only 15 % of the position control tests lasted more than 20 – 25 s, in other cases the quadrotor reached the wall in about 2 – 10 s. The velocity controller produced even worse results: it could keep the quadrotor in the arena for more than 20 s only in 10 % of the tests.

The following list provides a summary of the errors which were found in the behaviour of the quadrotor:

1. There is a non-deterministic drift in the yaw angle measurement
2. The quadrotor often rotates with +45° around its z axis during take-off
3. The flow sensor might measure large velocities at take-off
4. Recalibration might be necessary by each alteration of the quadrotor
5. The calibration procedure is time-consuming
6. There is an offset error in the pitch measurement even after calibration
7. The behaviour of the quadrotor is not deterministic
8. The ground plate of FINken v3 is not optimal for the sonar on the flow sensor
9. The environment for the flow sensor is not optimal
10. The arena is too small to test the velocity controller properly

Being able to obtain reliable position feedback from the quadrotor is one of the most significant improvements that should be done. Moreover, fixing the yaw angle measurement problem would solve the first and probably also the second problems listed. Therefore, my suggestion is to apply a position sensor, which can obtain also the orientation. It could be e.g. an external camera system, which tracks the quadrotors or a camera on board, which detects markers denoting position and orientation in the global NED coordinate system.

Alternatively, the gyroscope of the flow sensor could be used as an independent angular position feedback to improve not only the yaw, but also the pitch measurement.

To increase the stability of the quadrotor and to make it more robust the construction should be redesigned considering symmetry. For example the battery holder should provide a fix place for the battery to block its motion during flight. This might help to ensure a predictable behaviour of the quadrotor.

Another improvement regarding the hardware could be the redesign of the ground plate to ensure reflection-free operation for the altitude sonar sensor

To overcome the last two limitations listed, the controller could be tested outside, or the arena should be extended at least in one direction and the mattresses should be covered or replaced with patterned ones.

Until these problems are not solved, the position and velocity of the quadrotor cannot be controlled and more complex motions (e.g. trajectory control) cannot be realized.

Bibliography

[1] Bouktir, Y. ; Haddad, M. ; Chettibi, T. : Trajectory planning for a quadrotor helicopter. In: *16th Mediterranean Conference on Control and Automation Congress Centre, Ajaccio, France* (2008)

[2] Control Tutorials for MATLAB & SIMULINK: *Introduction: PID Controller Design.* `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID`, . – Online. Accessed: 1-October-2015

[3] Cowling, I. D. ; Whidborne, J. F. ; Cooke, A. K.: Optimal trajectory planning and LQR control for a quadrotor UAV. In: *Proceedings of the International Conference Control* (2006)

[4] Hehn, M. ; D'Andrea, R. : Quadrocopter Trajectory Generation and Control. In: *IFAC World Congress* (2011)

[5] Nikolic, J. ; Burri, M. ; Rehder, J. ; Leutenegger, S. ; Huerzeler, C. ; Siegwart, R. : A UAV system for inspection of industrial facilities. In: *Aerospace Conference, 2013 IEEE* (2013)

[6] PAPARAZZI UAV: *Aspirin IMU.* `http://wiki.paparazziuav.org/wiki/AspirinIMU`, . – Online. Accessed: 8-October-2015

[7] PAPARAZZI UAV: *Communication.* `https://wiki.paparazziuav.org/wiki/Overview#Communications`, . – Online. Accessed: 30-January-2016

[8] PAPARAZZI UAV: *Control Loops.* `http://wiki.paparazziuav.org/wiki/Control_Loops#Horizontal_Control`, . – Online. Accessed: 10-October-2015