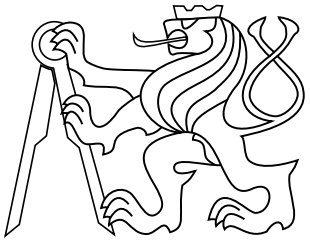
master's thesis

Autonomous Navigation of UAV

Tomáš Sekanina



May 2017

doc. Ing. Martin Hromčík, Ph.D.

Czech Technical University in Prague
Faculty of Electrical Engineering, Departement of Measurement

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering
Department of Measurement
Technická 2, 166 27 Prague 6, Czech Republic

Academic year **2016/2017**

MASTER PROJECT ASSIGNMENT

Student: **Bc. Tomáš Sekanina**

Study programme: **Cybernetics and Robotics**

Specialisation: **Aircraft and Spacecraft Instrumentation**

Title of Master Project: **Autonomous Navigation of UAV**

Systém autonomní navigace UAV zařízení (in Czech)

Guidelines:

The goal of the thesis is to design and implement an algorithm for autonomous navigation of a drone (copter and/or terrestrial rover) to a target specified by GPS coordinates. The work is linked to an existing cooperation of the team from the Faculty of Mechanical Engineering (Tomáš Vyhlídal, Jaroslav Bušek) with Georgia Tech University within the RescueBot project.

1. Get familiar with the state-of-the-art in autonomous navigation of UAVs.
2. Design an algorithm for autonomous navigation of a copter/drone towards a terrestrial target specified by GPS coordinates. Consider the copter has already implemented attitude stabilization subsystem.
3. Perform simulation validation and verification.
4. Implement the algorithm for autonomous drone navigation in the on-board control unit.
5. Verify the functionality of selected functions experimentally in test flights.

Bibliography/Sources:

[1] Blakelock: Automatic Control Systems of Aircraft and Missiles, Princeton University Press, 1985.

Master Project Supervisor: Doc. Ing. Martin Hromčík, Ph.D. (K 13135)

Valid until: September 30, 2018

L. S.

Prof. Ing. Jan Holub, Ph.D.
Head of Department

Prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 13, 2017

Acknowledgement

I would like to thank to doc. Hromčík for introducing me to this project and thus giving me opportunity to work on such interesting topic as well as supervising my work on it. I would also like to express my gratitude to Ing. Bušek for his guidance throughout the project and willingness to help me with encountered problems.

Author's Affirmation

I hereby declare that the submitted thesis is exclusively my own work and that I have listed all used information sources in accordance with the Methodological Guideline on Ethical Principles for College Final Work Preparation.

2017 prague,

Tomáš Sekanina

Abstract

Tato práce se zabývá metodami navigace bezpilotních prostředků, konkrétně vývojem algoritmu pro platformu kvadrokoptéry vytvořenou týmem ČVUT. Především se zaměřuje na filtrační algoritmy, potřebné k navigování letounu. Výsledný produkt by měl přispět k úspěšné účasti na studentské soutěži ARLISS [1]. Favoritem v této oblasti je Kálmánův filtr a proto byly zvoleny dvě jeho varianty – UKF a EKF. Vzhledem k tomu, že ke stabilizaci kvadrokoptéry je využit komerční produkt, který umožňuje řízení přes změnu natočení a výšky, byly také navrženy řídící smyčky potřebné k převodu z chyby pozice na vsuptní příkazy pro autopilot. Jak filtrační, tak regulační část byly ověřeny simulacemi a jejich výsledky vyhodnoceny. Následně byly tyto algoritmy implementovány v C++ pro zvolenou řídící jednotku, kterou je STM32F401.

Klíčová slova

arliss, rescuebot, uav, kalman, EKF, UKF

Abstract

This thesis deals with navigation methods of unmanned aerial vehicles, namely development for quadcopter platform created by CTU team. Above all, it focuses on filtration algorithms necessary for aircraft navigation. The resulting product should contribute to successful participation in student competition ARLISS [1]. Kalman filter is prevailing choice in this field and therefore were chosen two of its variants – UKF and EKF. Since a commercial product is used for stabilization of the quad rotor, which enables control via change of attitude and altitude, there were also designed control loops required for transformation of position error to autopilot's input commands. Both the filtering and regulatory parts were verified by simulations and their results evaluated. Subsequently, these algorithms were implemented in C++ for the selected control unit, which is STM32F401.

Keywords

arliss, rescuebot, uav, kalman, EKF, UKF

Contents

Abbreviations

UAV	Unmanned aerial vehicle
VTOL	Vertical take of landing
GPS	Global positioning system
IMU	Inertial measurement unit
INS	Inertial navigation system
CF	Complementary filter
KF	Kalman filter
EKF	Extended Kalman filter
CTU	Czech Technical University
ARLISS	A rocket launch for international student satellites
MEMS	Microelectromechanical systems
MCU	Microcontroller unit
OS	Operation system
NMEA	National marine electronics association
NED	North-East-Down
AHRS	Attitude and Heading reference system
DCM	Direct cosine matrix
MARG	Magnetic, angular rate and gravity
LRF	Laser range finder
CTU	Czech Technical University

1. Introduction

As the name suggests the main concern of this thesis navigation of autonomous Unmanned aerial vehicles (UAV). In the beginning of chapter 2 is briefly explained background of UAVs – what it is, its history and why is it today such frequent topic. After this necessary preliminary, follows introduction to navigation of aircrafts and current state of the art methods. The theoretical part concludes chapter 4, which is about filtering. This discipline plays major part in navigation since it enables estimating true parameters from distorted sensor readings. Therefore it is essential for localization and control of the aircraft. which gives brief outline of available filtering. There are described 4 different filtration methods, at first simple complementary filter, other two are variations of well-known Kalman filter (KF) and the third one is Madgwick filter.

Following chapter presents motivation for choosing this topic and project of group DICEbot [2], to which results of this thesis should contribute. Since its main aim is participating in an international competition, this part also briefly lays out its rules, because configuration explained in next chapter is relevant to it. The thesis continues with description of designed configuration and its individual parts, namely a microcontroller, autopilot and sensors. Next chapter gives basic information about used sensors and their errors.

The following text explains used process and measurement equations used in KF, with specifics of the implemented UKF and EKF. This is followed by chapter 8 which documents simulations of designed algorithms and their results. This part is followed by description of the final algorithm for the target microcontroller. The last chapter finishes with summary of results and recommandations for future development.

2. Navigation

Before going through past and current navigation methods it is important to first define what is actually UAV and look at its development throughout history, its possibilities and current applications. This covers the first section and rest of this chapter examines different navigation technologies.

2.1. UAV

Unmanned aerial vehicle (UAV) is an aircraft without on board crew but not necessarily not controlled by human. From this point could be established two group by a source of control. It can be done remotely by person or acting on its own, following specified instructions. The first kind can be done e.g. over radio and the aircraft can be either controlled directly such as by adjusting flaps or via some control system which inputs can be i.e. velocities in relevant coordinate frame etc. The later group are autonomous UAVs. This means that such UAV can fulfill its task without any intervention from human pilot. As the name of this thesis implies it takes focus in these.

2.1.1. History

The first occurrences resembling UAVs are documented even before the invention of a plane. As with other scientific advances, the main reason of its progress was the war, the first documented case [3] is bombarding of Venice in August 22, 1849 using hot air balloons. Another, this time peaceful, record is from 1883, when an Englishman named Douglas Archibald measured wind using instrument attached to a kite and 4 years later used it to take camera pictures [4]. The first successful attempts at remote control of aerial vehicle made Nikola Tesla in 1900 with wirelessly controlled airship[3].

As [4] further describes, the first larger advancement happened during World War I. At first the UAVs were basically just an air torpedoes, mechanically designed to hold some course and after desired distance crash to the ground. They were later also used at large for reconnaissance purposes. Great progress was made later throughout several wars in the 20th century in the field of remotely controlled aircrafts, as can be read in [4]. The development of autonomous aircrafts underwent great advancement in the last few decades. One of the milestones maybe being the First International Aerial Robotics Competition in July 29, 1991 [5] which annually continues since then.

Nowadays UAVs range in all sizes and shapes, mostly as multirotors, fixed wing airplanes and others. The next two paragraphs enumerate these different variations and give quick comparison.

2.1.2. Structure

There are two main categories of UAVs: Fixed-wing and multirotors. The first group of aircrafts uses lift generated by its wings during forward motion, while the other its horizontal propellers of which there may be an arbitrary number. The most common ones are with 4 or 6 propellers. The main advantages of rotorcrafts are that they

are capable of vertical take off and landing (VTOL) and can move in all directions or stay in place midair. These features make them suitable for indoor tasks, filming, manipulation with objects etc. The whole construction is simpler, as well as attaching additional instruments to them because their aerodynamics is not such a concern as with a fixed-wing aircraft. Another advantage over them is that it is not affected by some dynamics phenomenons like for example the dutch roll mode. On the other hand cannot travel such distances and fly as long as their counterpart.

The UAV usually consists of its body, motors, source of power – battery or fuel tank, transmitter for communication with the ground, some set of sensors and a control unit, which controls the motors and other adjustable parts of the aircraft. As mentioned earlier it can also carry additional cargo or equipment. The most common attachable instrument is a camera, which can be placed in a gimbal for stabilization and possible control of its attitude.

Besides inertial sensors and camera, the UAV can also be equipped with other sensors. Because of bad vertical accuracy of GPS, another sensor can be incorporated in the filtering process for better altitude estimation. Radar or pressure sensor can improve accuracy in high altitudes while for landing is suitable some precise short range distance sensor, such as ultrasounds or laser.

2.1.3. Current situation

There are ambitions and a lot of research is being invested in using autonomous UAVs for various tasks like 3D mapping, watching over places or exploration of natural disaster stricken areas and looking for survivors. Advancement of technology made possible that some of experiments already got to real application. It is so because the on board systems can nowadays run complex algorithms which allows all kinds of navigation, basic image processing or very precise movement. Lot of research is done in the field of collective robotics, artificial intelligence and indoor navigation for such purposes these days. Unfortunately there are still several problems hindering true utilization of such possibilities. These are mainly short air time because of weight of available power sources, legal reasons and still insufficient technology to apply it in general environment. So the experiments are still mostly done under laboratory circumstances.

2.2. Navigation

Navigation is a term covering three different disciplines. First is a localization, second path planning and at last following of the desired path. Basic principles of aircraft navigation are very similar and originate from maritime navigation. Speaking of first scientific methods, using instruments such as compass and astrolabe, it is documented to go through great advancement already in the 16th century [6]. Today there are much more sophisticated methods and instruments. Even though celestial navigation was largely used in aviation even in the 20th century, it got replaced by GPS in most cases. The prominent methods and current state of navigation in the field of autonomous UAVs is outlined in rest of this chapter.

Path planning and following is similar for different fields of robotics and is also extensively researched. Probably the most significant task for UAVs is position estimation. It is essential for the aircraft to know where it is and other parameters. This task is quite challenging compared i.e. to ground robots because of its high dynamics and different environment while in the air. Nevertheless these information can be obtained

from different sources which vary with application of the UAV. The most basic form is using inertial navigation which is also called dead reckoning.

2.2.1. Inertial navigation

Dead reckoning estimates position from knowing the initial course and location while integrating velocities. This can be easily done in the UAV by using inertial measurement unit (IMU), which is also necessary for control of the aircraft. IMU usually measures angular speed and specific force in all axis and thus should be able to provide attitude and velocity. Unfortunately well known problem in using relative measurements is, that errors of these sensors integrate every computational cycle. Microeletromechanical sensors (MEMS) are generally used in common UAVs, because of their small size, low price and ease of use. Todays high-end MEMS IMUs achieve tactical grade accuracy similar to mechanical gyroscopes, yet still, there is a big trade-off between price and accuracy. While the random walk of gyroscopes can vary from $0.002 - 5°/\sqrt{\text{Hr}}$ and accelerometers horizontal position error $57 - 1100\,\text{km}$ [7] the price can also span from few dollars to thousands.

Even though generally available IMUs are not usable for long term navigation on its own, they have high update rate and are mostly inert to outside world. To compensate for its drawbacks, additional absolute sensors can be integrated in the system, which fixes the estimated values. Purely inertial navigation is than used when the other sensor measurements are not available for some reason. This can be caused by lower sampling frequency then the IMU's or failure of the relevant system. Even though not exactly inertial sensor, magnetometer is usually included in IMUs. It offers absolute readings of attitude, especially the azimuth which typical IMU cannot provide unless aligned beforehand. Unfortunately, it is easily disturbed by other sources of magnetic field so some sort of error detection for faulty measurements and good calibration is required. One such approach was presented in [8] for underwater robot which is likely to encounter metal objects. This paper describes a way of identifying wrong accelerometer and magnetometer readings and eliminating them. Integration of measurements from different sensors is called filtering. It is introduced in chapter 4.

2.2.2. Satellite navigation

For absolute positioning is usually used set of beacons which allows triangulation of the position. In sense of their function these are also satellites. GPS is used in most outdoor UAV applications for exactly this purpose. It free, has great coverage ans its receivers are easily affordable. Besides position it also offers velocity, course over ground and many other parameters. Although GPS got very precise over time even for general public and there are multiple satellite systems to choose from, there are reasons why it is not sufficient without some complementary system as mentioned in previous section. Its update frequency is usually few Hz which is too slow for systems with high dynamics like UAV. Its accuracy is compromised by several factors which are discussed in chapter 3 and altitude and course information is not very accurate to begin with. Nevertheless, the most obvious reason of looking for alternatives or complements to GPS is that its signal is very unreliable or not available at all in some environments.

2.2.3. Visual navigation

The basic navigation methods mentioned in previous paragraphs still persist, at least as a redundancy system, but the camera no longer serves only for capturing interesting

moments. It allows application of complex computer vision algorithms like stereoscopy and other methods, which gave way to new dimensions of utilization.

With the development of computer vision emerged a new trend - using cameras as a means of navigation. Since image processing is quite demanding in terms of data flow and computing, one of the challenges is to make it optimized enough to be able to run on the UAVs on board computer. These methods are also usually used in combination with IMU and eventually other sensors. One of basic and often used methods is navigation using optic flow. In this approach a single camera traces movement of salient features in subsequent frames and deduces from it motion of the UAV. Example of such algorithm using downwards facing camera and IMU for position and velocity control was presented in [9]. Another usage of this method is for distance estimation from obstacles which is very important for usage of UAVs indoors or lower to the ground. Higher magnitude of optic flow means closer distance to the obstacle which makes it is suitable for flying in urban canyons. Comparison of using omnidirectional camera and two cameras with fish-eye lens to cover 360 degrees for this purpose was shown in [10].

Another option which inflicts higher computation load is using stereo vision. It enables to build 3D map of the environment and effective obstacle avoidance. This requires two cameras and can be combined with the optic flow method. Such case is documented in [11]. This paper presented that combination of these two methods using stereo camera facing forward and two sideways looking wide-range cameras was superior to using only one of the methods and eventually only a pair of cameras with fish-eye lens directed forward could be used, where outer sides of the images would be used for optic flow and the inner sides for stereo. Another paper [12] documented usage of stereo camera and rotating laser range finder (LRF) in GPS denied environment. The LRF made it possible to build accurate 3D map of cluttered environment and navigation in it.

2.3. Localization

The following chapters refer to terms which need to be explained first. Since the task at hand is concerned with 6-DOF rigid body, its location is defined by rotation and position in 3 axes. However both can be expressed in different ways.

First of all there are multiple reference frames used in aircraft navigation. The most significant being [13]:

- Inertial coordinate system – Frame in which applies Newton's laws of motion
- Geographic coordinate system – Earth centered, earth fixed (ECEF). Standard GPS coordinates: latitude, longitude, elevation.
- Navigation coordinate system – Local level: North, East, Down (NED).
- Body coordinate system – Frame aligned with the vehicle, mainly to describe orientation of on-board sensors reading and NED frame.

ECEF and NED frame are outlined in figure 1. The same frames are used also with rotations, only it is usually expressed as a rotation of one frame with respect to another.

As described in [14], it can be done by Euler angles, direct cosine matrix (DCM) or quaternion and some other ways but these are the most common. Euler angles are easy to imagine but suffer from singularities and dependency on their order. On the other hand quaternions with their unusual algebra enable easy and straightforward rotations. In the used implementation is attitude described in Euler angles, namely roll, pitch and yaw.

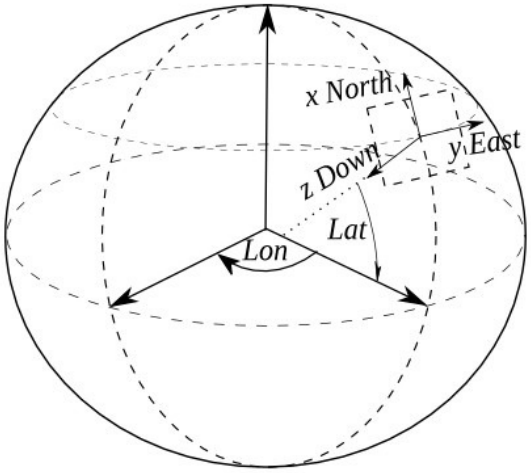


Figure 1. Quadrotor coordinate frames, from [15].

3. Sensors

For successful operation of quad rotor, the control algorithm needs to know its precise position in space and its velocities. That includes space coordinates $[x, y, z]$, its rotation angles $[\varphi, \theta, \psi]$ as well as their derivations (angular and translational velocities). However usage of commonly available sensors makes it challenging task, because neither of the measurements are very precise and only by combining the data and filtering them provides usable information. This process is described in the next chapter on multiple algorithms. But first it is necessary to introduce functions and features of each used sensor.

3.1. IMU

The inertial measurement unit consists of 3 angular rate sensors, 3 accelerometers and a 3 magnetometers. Even though MEMS angular rate sensor is not technically gyroscope, these two terms are used interchangeably. With combination of these three triads, it is then possible to measure acceleration, angular velocity and magnetic field in all axes. It mainly serves for estimation of attitude, velocities and accelerations for the controller but also complements GPS in providing position data, especially in case of GPS signal outage.

Because of Earth's gravitation field, accelerometer can provide relatively reliable information about pitch and roll when the device is not moving. On the other hand magnetometer can measure rotation in all axis and is not affected by motion. Its greatest disadvantage is that it is very easily influenced by other sources of magnetic field, another is that the Earth's magnetic field differs with location so it needs to be taken in account as well. Gyroscope provides accurate information about rotation velocities and by integrating its data relative attitude could be estimated as well but its biases tends to drift and there is always some additive error when integrating. Errors of these devices are throughly described in [16] so the following paragraphs give only a brief insight to this problematic.

3.1.1. IMU errors

All the mentioned sensors suffer from instrumentation errors which are listed in table 1. Scale factor, bias and misalignment of the sensor can be compensated easily by formula 1, but they have to be determined first. $\boldsymbol{x}$ is a true value of the measured $\hat{\boldsymbol{x}}$ after compensation, $\boldsymbol{M}$ corrects misalignment of sensor, $\boldsymbol{S}$ is scale factor and $\boldsymbol{b}$ is bias. This can be easily illustrated by an error ellipse which as in figure 2. The red ellipse represents raw measured values when rotating the sensor around one axis, while the true data and desired shape after compensation is the blue circle. Displacement from origin is caused by bias, rotation by nonorthogonality and not being perfect circle by scale factor. In a 3D situation the sensor is usually rotated around until the measurements form an ellipsoid and using some fitting algorithm its parameters are calculated. This enables then to obtain the real value. However some parameters, especially gyroscope

biases, tend to more or less change over time, so it is sufficient only for high grade sensors.

$$\begin{bmatrix} x_x \\ x_y \\ x_z \end{bmatrix} = \begin{bmatrix} 1 & M_{xy} & M_{xz} \\ M_{yx} & 1 & M_{yz} \\ M_{zx} & M_{zy} & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & \frac{1}{S_z} \end{bmatrix} \left(\begin{bmatrix} \hat{x}_x \\ \hat{x}_y \\ \hat{x}_z \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \right) \tag{1}$$

Bias	Offset of measured values.
Scale factor	Disproportion of the measured values.
Nonorthogonality	Misalignment of sensor axis.
Sensor noise	Random noise in sensor measurements. Usually said to be zero-mean white noise.

Table 1. Common IMU sensor errors

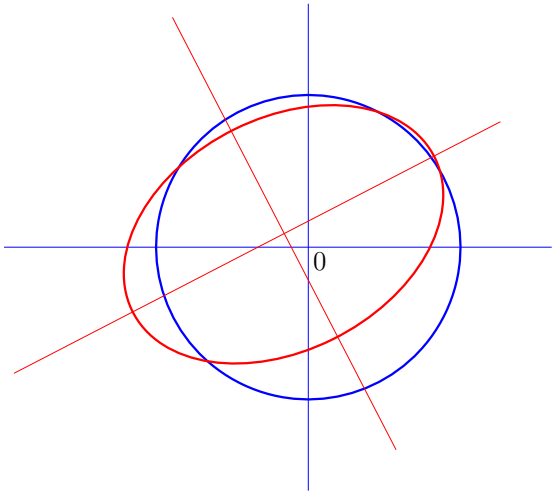


Figure 2. Sensor error ellipse. Blue – compensated measurement, Red – distorted measurement

Accelerometer and gyroscope

Both device are affected by already mentioned errors, gyroscope can be also affected by Earth spin but it is neglected with used grade of sensors and each are also influenced by temperature. The temperature dependency can be usually compensated by measuring values in different temperatures beforehand and find a formula via some polynomial fitting to recalculate the values. Calibration of accelerometer is relatively straightforward since it is known that if static, norm of the values should be always equal to Earths gravitation acceleration.

3.1.2. Magnetometer

Magnetometer calibration is similar to accelerometers in terms of scale factor, bias and misalignment, thought the measurements can be done even on moving sensor. Problem with magnetometer is that Earths magnetic field is very weak and easily disturbed. Such distortions are called soft and hard iron. These and another sources of errors are

deeply discussed in [17], but the most common and with greatest influence are these two. Hard iron effects are caused by permanent magnets near the sensor and behave just like bias so it is only necessary to do the calibration with the sensor attached in its place. Soft iron is caused by ferromagnetic compounds with changing intensity of their field. This can be modeled to some extent or a characteristics for i.e. motors on the robot can be measured to compensate for.

3.2. GPS

The Global Positioning System is a satellite-based navigation system. The client device calculates distance to chosen available satellite from time it takes message to travel over there. Using such information from at least 4 satellites it is able to calculate unique position in latitude, longitude and altitude. GPS has also multiple error sources which can be divided into two main groups:

- Common mode errors
  - Satellite clock error
  - Orbital error
  - Ionospheric error
  - Tropospheric error
- Non-common mode errors (receiver specific errors)
  - Clock errors
  - Multipath
  - Noise

Common mode errors are minimized by specialized methods and extensions of the space part i.e. differential GPS. The later group depends on grade of the receiver which has usually included some mechanisms to suppress these segments.

The greatest vulnerability of this system is its need of a clear view of at least 4 satellites.[1] Due to surroundings of the receiver such as high buildings, view of the sky can be quite obstructed and view of the satellites with it. Because of this or another cause, full GPS outages might occur. In such cases, the system has to rely on information from another sensors to determine its position. This and demand for higher accuracy is why there are filters which allow incorporating data from multiple sensors to make the readings more reliable. Such filters are discussed in the next chapter.

[1] 3 for just horizontal localization.

4. Filtering

Since there are no perfect sensors, each of them has different traits and all measurements are affected by errors, especially in aviation, are often used multiple devices to estimate a single value. Pure estimation of values could be done via simple averaging but using more advanced methods allows modeling state of the system and its errors. Basic filtering algorithm which could be used for e.g. inertial navigation is complementary filter, but it is too simple to provide sufficient results. The most of todays navigation algorithms are based on the Kalman filter (KF), which is named after Rudolf E. Kálmán. Because KF is only for linear problems it had to be adapted for nonlinear tasks such as attitude estimation. Nowadays there are countless implementations which are based on this filter. Other frequent filters in navigation are particle filter, information filter or Madgwick filter.

4.1. Complementary filter

The complementary filter does not require any difficult computations so it is easy to implement and fast to run. It can be used in a situation where there are two sensors s_1 and s_2 [18]. s_1 is disturbed by high frequency noise and the other one by low frequency noise. For each sensor a filter is used, low pass $F(s)$ for s_1 and high pass $G(s)$ for s_2. Where $G(s)$ is complement of $F(s)$, thus $G(s) = 1 - F(s)$. Then we can express the estimated value as in equation 2.

$$\hat{x} = xF(s) + xG(s) = xF(s) + x[1 - F(s)] \tag{2}$$

Such filter could be used i.e. for combining data from accelerometer and gyroscope. There also exists a non-linear version. Since it is very simple and not implemented it is not discussed any further. More information can be found in the cited paper.

4.2. Kalman Filter

Kalman filter is a discrete, recursive and optimal algorithm for estimation of linear dynamic systems. Because it is recursive, it uses only the last estimate for the next computation and does no need to store large number of measurements. It also does not require any demanding computations. These two factors make it very efficient and suitable for running on embedded systems where HW performance is limited. The algorithm is briefly derived according to [19] in the following section.

4.2.1. KF algorithm

KF applies to linear, discrete-time, dynamical systems described by figure 3. This diagram can be divided into two blocks: Process and measurement part. Symbols used in following derivation are in table 2. The goal of this process is to estimate state $\boldsymbol{x}$ as precise as possible. Vector $\boldsymbol{y}_n$ contains all observed values at the given time step. These states values have their own models and are each distorted by different noises.

Both noises are assumed to be additive, white, Gaussian with zero mean and mutually uncorrelated. These two parts are derived separately in rest of this subsection.

Symbol	Definition
$\boldsymbol{x}_n$	state vector of the system
$\boldsymbol{y}_n$	set of measured data
$\boldsymbol{F}_{m,n}$	transition matrix from state $\boldsymbol{x}_n$ to $\boldsymbol{x}_m$
$\boldsymbol{w}_n$	process noise
$\boldsymbol{H}_n$	measurement matrix
$\boldsymbol{v}_k$	measurement noise
n	index defining iteration step

Table 2. Symbol definitions for Kalman filter derivation

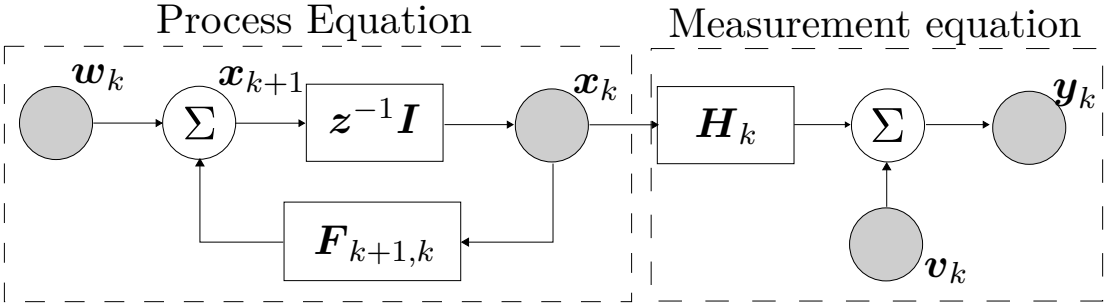


Figure 3. Digram of Kalman filter for discrete linear system [19]

Process equation

Having block diagram of the system, process equation is defined as:

$$\boldsymbol{x}_k = \boldsymbol{F}_{k,k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \tag{3}$$

where M is dimension of the state space. Process noise $\boldsymbol{w}_k$ is defined as:

$$E[\boldsymbol{w}_n\boldsymbol{w}_k^T] = \begin{cases} \boldsymbol{Q}_k & \text{for } n = k \\ \boldsymbol{0} & \text{for } n \neq k \end{cases} \tag{4}$$

Where $E[A]$ is an expected value of A.

Measurement equation

Equation for the measurement block is:

$$\boldsymbol{y}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k, \tag{5}$$

where N is dimension of the measurement space. Covariance matrix of the measurement noise $\boldsymbol{v}_k$ is defined as:

$$E[\boldsymbol{v}_n\boldsymbol{v}_k^T] = \begin{cases} \boldsymbol{R}_k & \text{for } n = k \\ \boldsymbol{0} & \text{for } n \neq k \end{cases} \tag{6}$$

The goal of Kalman filter is to find a minimum mean-square error estimate of $\boldsymbol{x}_i$ using observed data $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$. There are three possible variants with different names as follows:

$$
\begin{aligned}
&i = k \text{ – filtering,}\\
&1 \leq i < k \text{ – smoothing,}\\
&i > k \text{ – prediction.}
\end{aligned}
$$

Elaborate derivation follows in the cited book [19]. The final system then looks like set of equations 3 and 5:

$$
\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{F}_{k,k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}\\
\boldsymbol{y}_k &= \boldsymbol{H}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1}
\end{aligned}
\tag{7}
$$

Life-cycle of the algorithm

The initial state is:

$$
\begin{aligned}
\hat{\boldsymbol{x}}_0 &= E[\boldsymbol{x}_0]),\\
\boldsymbol{P}_0 &= E[(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])^T].
\end{aligned}
\tag{8}
$$

And next steps are calculated from formulas:

$$
\begin{aligned}
&\text{State estimate propagation} && \hat{\boldsymbol{x}}_k^- = \boldsymbol{F}_{k,k-1}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}},\\
&\text{Error covariance propagation} && \boldsymbol{P}_k^- = \boldsymbol{F}_{k,k-1}\boldsymbol{P}_{k-1}\boldsymbol{F}_{k,k-1}^T + \boldsymbol{Q}_{k-1},\\
&\text{Kalman gain matrix} && \boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^T[\boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1},\\
&\text{State estimate update} && \hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k^-),\\
&\text{Error covariance update} && \boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^-.
\end{aligned}
\tag{9}
$$

However as noted in the beginning, it is not possible to use KF for nonlinear problems, which gives rise to adaptations of the former algorithm. The most common version is the Extended Kalman Filter (EKF) but there are also others among which is likely the most notable Unscented Kalman Filter (UKF).

4.2.2. Extended Kalman Filter

Kalman filter allows filtering of linear systems, however most real-life problems, including estimation of position of quad rotor, are non-linear. For such models linearization is a must. Combination of linearization and KF is called Extended Kalman filter. Using equation 7 new non-linear system is defined:

$$
\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{f}(k, \boldsymbol{x}_k) + \boldsymbol{w}_k,\\
\boldsymbol{y}_{k+1} &= \boldsymbol{h}(k, \boldsymbol{x}_k) + \boldsymbol{v}_k.
\end{aligned}
\tag{10}
$$

Same rules apply for process and measurement noise and their covariance matrices. Only transition matrices $\boldsymbol{F}_k$ and $\boldsymbol{H}_k$ are replaced by non-linear, possibly time-variant, transition matrix functions $\boldsymbol{f}(k, \boldsymbol{x}_k)$ and $\boldsymbol{h}(k, \boldsymbol{x}_k)$. When the state space model is linearized, standard KF is used. Common linearization consists of two steps. At first partial derivations of transition matrices are calculated and then evaluated for the current state:

$$
\begin{aligned}
\boldsymbol{F}_{k+1,k} &= \left.\frac{\partial \boldsymbol{f}(k, \boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k},\\
\boldsymbol{H}_k &= \left.\frac{\partial \boldsymbol{h}(k, \boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-}.
\end{aligned}
\tag{11}
$$

These values are then used in first order Taylor polynomial:

$$
\begin{aligned}
\boldsymbol{F}(\boldsymbol{k}, \boldsymbol{x_k}) &\approx \boldsymbol{F}(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\boldsymbol{k}}) + \boldsymbol{F}_{\boldsymbol{k+1,k}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\boldsymbol{k}}),\\
\boldsymbol{H}(\boldsymbol{k}, \boldsymbol{x_k}) &\approx \boldsymbol{H}(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\boldsymbol{k}}^-) + \boldsymbol{H}_{\boldsymbol{k+1,k}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\boldsymbol{k}}^-).
\end{aligned}
\tag{12}
$$

For the final approximation of the non-linear state equations 10, two new formulas are needed:

$$\begin{aligned} \bar{\boldsymbol{y}}_k &= \boldsymbol{y}_k - \{\boldsymbol{h}(k, \hat{\boldsymbol{x}}_k^-) - \boldsymbol{H}_k \hat{\boldsymbol{x}}_k^-\}, \\ \boldsymbol{d}_k &= \boldsymbol{f}(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) - \boldsymbol{F}_{k+1,k} \hat{\boldsymbol{x}}_k. \end{aligned} \tag{13}$$

From which approximation of non-linear equations finally can be obtained as:

$$\begin{aligned} \bar{\boldsymbol{x}}_{k+1} &\approx \boldsymbol{F}_{k+1,k} \boldsymbol{x}_k + \boldsymbol{w}_k + \boldsymbol{d}_k, \\ \bar{\boldsymbol{y}}_k &\approx \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k. \end{aligned} \tag{14}$$

Then the final algorithm for the system is defined by equations 10.

Life-cycle of the algorithm

Firstly the initialization sequence is the same as in KFs formulas 8:

$$\begin{aligned} \hat{\boldsymbol{x}}_0 &= E[\boldsymbol{x}_0]), \\ \boldsymbol{P}_0 &= E[(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])^T]. \end{aligned}$$

And every step new computation of states and covariance matrices is done:

$$\begin{aligned} &\text{Calculate state matrix jacobian} && \boldsymbol{F}_{k,k-1} = \partial \boldsymbol{f}(k, \boldsymbol{x}) / \partial \boldsymbol{x}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}}, \\ &\text{State estimate propagation} && \hat{\boldsymbol{x}}_k^- = \boldsymbol{f}(k, \hat{\boldsymbol{x}}_{k-1}), \\ &\text{Calculate measurement matrix jacobian} && \boldsymbol{H}_k = \partial \boldsymbol{h}(k, \boldsymbol{x}) / \partial \boldsymbol{x}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-} \\ &\text{Error covariance propagation} && \boldsymbol{P}_k^- = \boldsymbol{F}_{k,k-1} \boldsymbol{P}_{k-1} \boldsymbol{F}_{k,k-1}^T + \boldsymbol{Q}_{k-1}, \\ &\text{Kalman gain matrix} && \boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T [\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1}, \\ &\text{State estimate update} && \hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (\boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_k^-), \\ &\text{Error covariance update} && \boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^-. \end{aligned} \tag{15}$$

4.2.3. Unscented Kalman Filter

As is thoroughly explained in [20] the EKF's accuracy suffers from linearization of the transformation equations which might not be usable at all in some cases. It further presents advantages and steps of the Unscented Kalman Filter. This variation of KF is named after the unscented transform, which allows using nonlinear equations while only approximating a probabilistic distribution. It does so by selecting a so called sigma points which hold information about mean a covariance of the data set. These are afterwards transformed using the nonlinear equations and the estimated probabilistic distribution is calculated from them.

Unscented transformation

As [21] states, "the unscented transformation(UT) is a method for calculating the statistics of a random variable which undergoes nonlinear transformation". It further describes its steps starting with the selection of sigma points as:

$$\begin{aligned} \mathcal{X}_0 &= \bar{\boldsymbol{x}} \\ \mathcal{X}_i &= \bar{\boldsymbol{x}} + (\sqrt{(L+\lambda)\boldsymbol{P}_x})_i \qquad i = 1, \ldots, L \\ \mathcal{X}_i &= \bar{\boldsymbol{x}} - (\sqrt{(L+\lambda)\boldsymbol{P}_x})_{i-L} \quad i = L+1, \ldots, 2L, \end{aligned} \tag{16}$$

$$\lambda = \alpha^2(L + \kappa) - L$$

where $\boldsymbol{x}$ is the random variable in L-dimensional space, with mean $\bar{\boldsymbol{x}}$ and covariance $\boldsymbol{P}_x$. α and κ are parameters defining spread of the sigma points. Each sigma point is also associated with its weight. The weights are calculated by following equations:

$$\begin{aligned} W_0^{(m)} &= \frac{\lambda}{L+\lambda} \\ W_0^{(c)} &= \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \\ W_i^{(m)} &= W_i^{(c)} = \frac{1}{2(L+\lambda)} \qquad i = 1, \ldots, 2L, \end{aligned} \tag{17}$$

where β is parameter incorporating prior knowledge of the distribution. Optimal value for Gaussian is 2. Having the sigma points, they are transformed using the nonlinear function g:

$$\mathcal{Y}_i = g(\mathcal{X}_i) \quad i = 0, \ldots, 2L \tag{18}$$

Finally the estimated mean and covariance can be calculated as

$$\begin{aligned} \bar{\boldsymbol{y}} &\approx \sum_{i=0}^{2L} W_i^m \mathcal{Y}_i \\ \boldsymbol{P}_y &\approx \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_i - \bar{\boldsymbol{y}})(\mathcal{Y}_i - \bar{\boldsymbol{y}})^T. \end{aligned} \tag{19}$$

Selection of the sigma points plays a significant role. Their number affects computation cost so the lesser the better, but it is necessary for them to properly embody the probabilistic distribution of the original set. [22] derives method of selecting them and proves that the minimal sufficient number is $2L$ points when properly selected, though $2L + 1$ points are usually used.

Life-cycle of the algorithm

The initialization sequence of UKF is the same as in KFs formulas 8:

$$\begin{aligned} \hat{\boldsymbol{x}}_0 &= E[\boldsymbol{x}_0]), \\ \boldsymbol{P}_0 &= E[(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])(\boldsymbol{x}_0 - E[\boldsymbol{x}_0])^T]. \end{aligned}$$

And at every cycle, unscented transformation is calculated for new set of sigma points for process and measurement separately, from which new mean and covariance estimates

are computed:

$$
\begin{aligned}
&\text{State sigma points by 16} && \boldsymbol{\mathcal{X}}_{k-1},\\
&\text{Transformed state sigma points} && \boldsymbol{\mathcal{X}}_k^- = f(\boldsymbol{\mathcal{X}}_{-1}),\\
&\text{State apriory estimate} && \hat{\boldsymbol{x}}_k^- = \sum_{i=0}^{2L} W_i^x \boldsymbol{\mathcal{X}}_{k,i},\\
&\text{State covariance apriory estimate} && \boldsymbol{P}_{x_k}^- = \sum_{i=0}^{2L} W_i^c (\boldsymbol{\mathcal{X}}_{k,i}^- - \hat{\boldsymbol{x}}_k^-)(\boldsymbol{\mathcal{X}}_{k,i}^- - \hat{\boldsymbol{x}}_k^-)^T + \boldsymbol{Q}_k,\\
&\text{Transformed measurement sigma points} && \boldsymbol{\mathcal{Y}}_{k,i} = h(\boldsymbol{\mathcal{X}}_{k,i}^-),\\
&\text{Measurement estimate} && \hat{\boldsymbol{y}}_k = \sum_{i=0}^{2L} W_i^x \boldsymbol{\mathcal{Y}}_{k,i}\\
&\text{Measurement covariance} && \boldsymbol{P}_{y_k} = \sum_{i=0}^{2L} W_i^c (\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k)(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k)^T + \boldsymbol{R}_k,\\
&\text{Transformed cross covariance} && \boldsymbol{P}_{x_k,y_k} = \sum_{i=0}^{2L} W_i^c (\boldsymbol{\mathcal{X}}_{k,i}^- - \hat{\boldsymbol{x}}_k^-)(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k)^T\\
&\text{Kalman gain matrix} && \boldsymbol{K}_k = \boldsymbol{P}_{x_k y_k} \boldsymbol{P}_{y_k}^{-1},\\
&\text{State estimate} && \hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k),\\
&\text{State covariance estimate} && \boldsymbol{P}_{x_k} = \boldsymbol{P}_{x_k}^- - \boldsymbol{K}_k \boldsymbol{P}_{y_k} \boldsymbol{K}_k^T.
\end{aligned}
\tag{20}
$$

4.2.4. Other versions

Besides EKF and UKF exist also other various adaptation of KF. Thorough comparison of EKF and several versions of UKF is in [23]. Often researched trend are adaptive filters which adjust the covariance matrices on the fly. Examples of such algorithms can be Adaptive Kalman Filter [24], Adaptive Extended Kalman Filter [25] or Adaptive Unscented Kalman Filter [26].

4.3. Madgwick filter

Another interesting approach for estimating attitude using 6 and 9-DOF IMUs published Sebastian O. H. Madgwick in his paper [27]. It uses quaternions and is said to be computationally less demanding than EKF while maintaining similar results.

4.3.1. The algorithm

MARG version of Madgwicks filter is displayed in figure 4. There are two inputs from magnetometer and accelerometer which serve for estimation of attitude using the gradient descend algorithm and a third input from gyroscope which provides information about rate of change of rotation. These two estimates are combined for the final attitude estimate which is then incorporated in a feedback. Group 1 implements magnetic distortion compensation and Group 2 takes care of gyroscope bias drift.

4.3.2. Orientation frames

The gyroscope measures angular rate in sensor frame which is defined by vector

$$
{}^S\boldsymbol{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}. \tag{21}
$$

The rate of change in the earth frame is than

$$
{}^S_E\dot{\boldsymbol{q}}_{\omega,t} = \frac{1}{2} {}^S_E\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^S\boldsymbol{\omega}_t. \tag{22}
$$

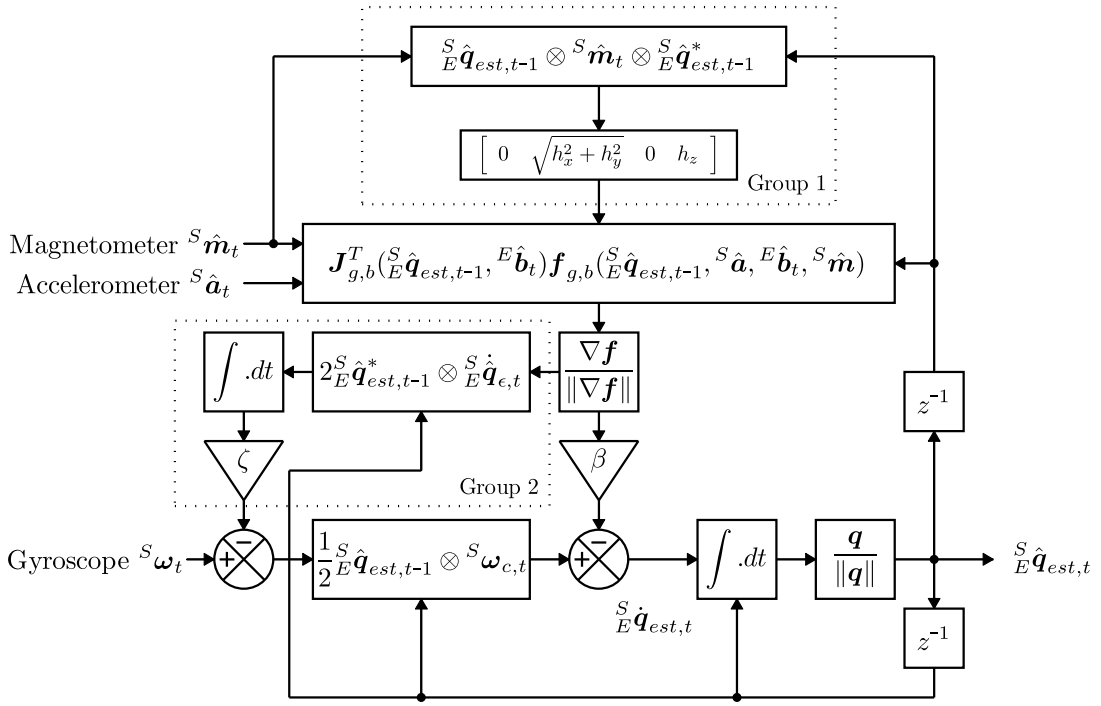


Figure 4. Block diagram of Madgwicks MARG filter algorithm [27]

Finally the estimated quaternion is obtained by integration of angular rate:

$$\overset{S}{E}\hat{\boldsymbol{q}}_{est,t} = \overset{S}{E}\hat{\boldsymbol{q}}_{est,t-1} + \overset{S}{E}\dot{\hat{\boldsymbol{q}}}_{est,t}\Delta t \tag{23}$$

To calculate orientation in the Earth frame from accelerometer and magnetometer it is necessary to use some optimization algorithm to obtain complete solution. Madgwick in his paper uses a gradient step algorithm.

4.3.3. Gradient descend algorithm

The optimization problem is solved by minimizing the objective function $\boldsymbol{f}\left(\overset{S}{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right)$ defined by equation 24, using gradient-descend algorithm.

$$\boldsymbol{f}\left(\overset{S}{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) = \overset{S}{E}\hat{\boldsymbol{q}}^{*} \otimes {}^{E}\hat{\boldsymbol{d}} \otimes \overset{S}{E}\hat{\boldsymbol{q}} - {}^{S}\hat{\boldsymbol{s}} \tag{24}$$

Where ${}^{E}\hat{\boldsymbol{d}}$ denotes reference direction in the earth frame and ${}^{S}\hat{\boldsymbol{s}}$ measured direction in sensor frame.

In the objective function for accelerometer ${}^{E}\hat{\boldsymbol{d}}$ is substituted by gravity vector

$${}^{E}\hat{\boldsymbol{g}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

and ${}^{S}\hat{\boldsymbol{s}}$ with accelerometer measurement

$${}^{E}\hat{\boldsymbol{a}} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix}.$$

In case of the magnetometer ${}^{E}\hat{\boldsymbol{d}}$ is substituted by

$${}^{E}\hat{\boldsymbol{b}} = \begin{bmatrix} 0 & b_x & 0 & b_z \end{bmatrix}$$

and ${}^{S}\hat{\boldsymbol{s}}$ by normalized magnetometer measurement

$${}^{S}\hat{\boldsymbol{m}} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix}.$$

The objective functions $\boldsymbol{f}_g$ for accelerometer and $\boldsymbol{f}_b$ for magnetometer are defined by equations 25 and 26, which can be obtained from the general formula after some simplifications which are described in the original paper.

$$\boldsymbol{f}_g({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{S}\hat{\boldsymbol{a}}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{25}$$

$$\boldsymbol{f}_b\left({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{S}\hat{\boldsymbol{a}}\right) = \begin{bmatrix} 2b_x(0.5 - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\ 2b_x(q_1q_3 + q_2q_4) + 2b_z(0.5 - q_2^2 - q_3^2) - m_z \end{bmatrix} \tag{26}$$

Jacobians $\boldsymbol{J}_g({}^{S}_{E}\hat{\boldsymbol{q}})$ and $\boldsymbol{J}_b({}^{S}_{E}\hat{\boldsymbol{q}})$ can be easily obtained by derivation of their respective objective functions. The next orientation estimate is calculated from equation 27 using step size μ.

$${}^{S}_{E}\boldsymbol{q}_{\nabla,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} - \mu_t \frac{\nabla \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right)}{\left\| \nabla \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_k, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) \right\|}, \; k = 1, 2, \ldots n \tag{27}$$

$$\nabla \boldsymbol{f} = \begin{bmatrix} \boldsymbol{J}_g({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}) \\ \boldsymbol{J}_b({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{E}\hat{\boldsymbol{b}}) \end{bmatrix}^T \begin{bmatrix} \boldsymbol{f}_g({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}, {}^{S}\hat{\boldsymbol{a}}) \\ \boldsymbol{f}_b\left({}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1,}, {}^{E}\hat{\boldsymbol{b}}, {}^{S}\hat{\boldsymbol{m}}\right) \end{bmatrix} \tag{28}$$

Magnetic distortion

To compensate magnetic distortion, ${}^{E}\hat{b}_t$ is defined by equation 29

$${}^{E}\hat{\boldsymbol{b}}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \tag{29}$$

$${}^{E}\hat{\boldsymbol{h}}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^{S}\hat{\boldsymbol{m}}_t \otimes {}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{est,t-1} \tag{30}$$

Gyro bias drift

The gyro scope bias changes slowly over time with temperature and motion. Apart from Kalman filter, Madgwick estimates gyroscope bias drift via integral feedback of the error in the rate of change orientation. The compensated gyroscope measurement is than calculated as the figure suggest via set of equations 31.

$$\begin{aligned} {}^{S}\boldsymbol{\omega}_{\epsilon,t} &= 2\, {}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{est,t-1} \otimes {}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} \\ {}^{S}\boldsymbol{\omega}_{b,t} &= \zeta \sum_t {}^{S}\boldsymbol{\omega}_{\epsilon,t}\Delta t \\ {}^{S}\boldsymbol{\omega}_{c,t} &= {}^{S}\boldsymbol{\omega}_t - {}^{S}\boldsymbol{\omega}_{b,t} \end{aligned} \tag{31}$$

Algorithm configuration

There are only two gains to be tuned in the filter β and ζ. The first represents all mean zero gyroscope measurement error and the latter rate of convergence to remove gyroscope measurement errors which are not mean zero. These can be determined using following equations.

$$\beta = \left\| \frac{1}{2} \hat{\boldsymbol{q}} \otimes \begin{bmatrix} 0 & \tilde{\omega}_\beta & \tilde{\omega}_\beta & \tilde{\omega}_\beta \end{bmatrix} \right\| = \sqrt{\frac{3}{4}} \tilde{\omega}_\beta \tag{32}$$

$$\zeta = \sqrt{\frac{3}{4}} \tilde{\omega}_\zeta \tag{33}$$

5. The project

In previous chapters were described application and history of UAVs, evolution of navigation methods and filtration techniques used today, with detailed look on some filtering algorithms. All of this comes as a background for the following part of this thesis where these algorithms are applied on a real project. In this chapter is described motivation behind this effort and purpose of the project.

5.1. Motivation

The practical part of this thesis is built on an ongoing development of CTUs team [2] which represents its university with it in an international ARLISS [1] competition. This event takes place every year in Black Rock Desert, Nevada and CTU's team already participated in the previous runs. Rules of the competition are introduced in the next section.

The project engages mainly in the fields of electrical and mechanical engineering and aviation. Having the mechanical part done from previous work of another student, it is welcomed opportunity to build own solution almost from scratch, while the field of work is highly relevant to studied program. Another great possibility is, if possible, to eventually personally participate in the competition. This is conditioned by successful solution of this task of course.

5.1.1. ARLISS competition

The competition is held for students interested in aerospace engineering, so they can get practical experience with building a complete solution for this kind of task. Teams from all over the world gather in late September to compare their skills in two categories: CanSat and Open class. The CTU's robot is built for the later.

Figure 5. Launch of a rocket carrying the device during ARLISS competition [28]

5.1.2. Open class category

The assignment in this category is to create a device able to autonomously navigate to a target specified by GPS coordinates located somewhere in 6 km radius from the place of launch. At first, the device is launched circa 3 km high, as can be seen in figure 5, and released. Then it has to travel to the provided location and stop there with 10 m accuracy. The robot has to fit in the carrier device and so there are size and weight restrictions, which are listed in table 7.

Dimension	Maximum value
Diameter	146 mm
Height	254 mm
Weight	1.8 kg

Table 3. List of restrictions for robot design.

There are two main categories of robots: ground and flying. Most teams choose to build a rover – robot which moves over the ground, however CTU team's solution is a quad copter. Its design specification including the frame and electronics are described in chapter 6.

6. HW set-up

Taking propositions of the competition in account, it is kind of a unique application. Hence building a custom made solution is a logical approach, because mass produced goods would hardly fill all needs and would cost multiple times over. Schema of the HW set-up is in figure 6. Each part of the design is described in rest of this chapter.

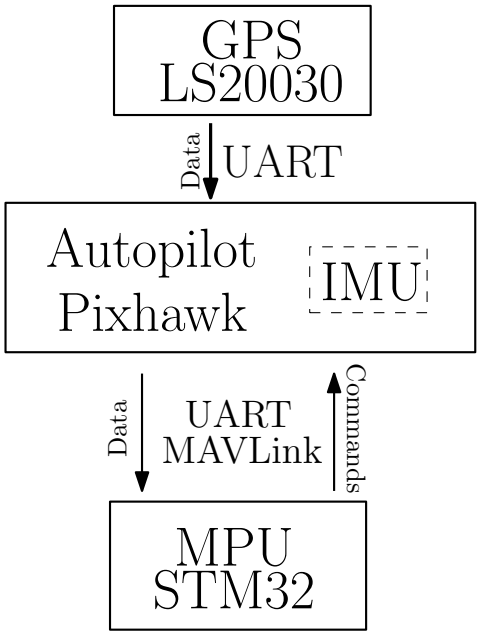


Figure 6. Custom quadrotor of CTU team for ARLISS competition

6.1. Quad rotor

The quad rotor was built by DICEbot team [2] at the Faculty of Mechanical Engineering of CTU. Since the quad rotor has to fit in a small tube, the beams with propellers have to be folded to be in parallel. Therefore it was necessary to implement a mechanism which would spread these supports to their working position. The quad rotor is equipped with a parachute which unfolds right after the robot is released and besides stabilization during initial fall it is also attached to mechanism which opens up unfolds the structure. The built frame is in picture 7.

6.2. Microcontroller

There are countless options when it comes to choosing a platform that could control a quad rotor but there are 2 main choices in general. Firstly there are small computers with OS such as Raspberry Pi [29] or BeagleBone Black[30] and secondly microcontrollers with market leader Arduino[31] or STM32[32]. The price is similar for both

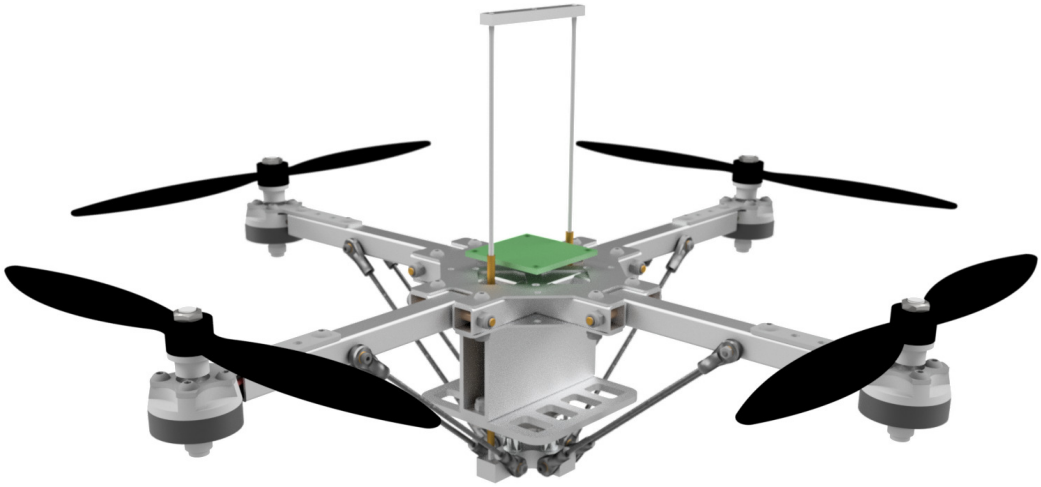


Figure 7. Custom quadrotor of CTU team for ARLISS competition

groups depending on its class but where computers have superior performance to MCUs, with power consumption it is exactly the other way around. This fact makes MCUs more suitable for robotic applications if the provided performance is sufficient and run time on battery is critical factor. MCUs also provide more deterministic behavior when it comes to timing compared to usually used UNIX distributions, though there are also real-time systems for specific purposes.

Considering this comparison, using a microcontroller for the ARLISS competition is obvious choice since airtime of quad rotor is already short due to its motors. Then there are another 2 options: There are universal development kits with large amounts of GPIOs, multiple peripherals and communication interfaces or there is a possibility to build an ad-hoc board based on CPU of choice which would be minimalistic. Since this is a pilot project and a lot can change in next phases of development, multi-purpose MCU kit is used for this work. However building some lighter and more application specific solution is desirable for further development.

Because of its availability, SW support and performance, development kit based on the high-performance ARM© Cortex©-M4 32-bit RISC core operating at a frequency of up to 84 MHz[33] was selected. Comparison to high-end non-USA Arduino version Genuino is in table 4.

Peripherals	**STM32F401VC**	**Genuino Zero**
Price(official)[€]	13	42.90
Flash memory	256 kB	256 kB
SRAM	64 kB	32 kB
GPIO	81	20
Maximum CPU frequency	84 MHz	48 MHz

Table 4. Comparison of ST32MF401VC[33] and Genuino Zero[34]

The STM32F401VC kit is a universal high-performance controller which covers all needs of the application in terms of CPU frequency and communication interfaces

needed for attaching additional sensors. Its Cortex©-M4 core features a Floating point unit which is quite advantageous in this kind of application and has also good software support with many necessary libraries already existing.

6.3. Stabilization

Since stabilization of quadcopter is a task of its own and this thesis is mainly concerned with navigation a Pixhawk Autopilot [35] is used for this. It is a multi-purpose module which can be used with different vehicles, sensors and communication protocols. It built on STM32F427, which is MCU similar to the microcontroller used for running the main algorithm. The system also hosts two two sets of accelerometers and angular rate sensors plus one magnetometer and barometer. Another useful features are several integrated backup function which make it suitable for use in aircrafts. It also comes with an extensive software support and large user community.

It has three main modes of use – manual, assisted and auto. These then have their own variations depending on type of the aircraft. For this application was chosen the assisted *ALTCTL* mode, where the control for multirotors is done by pitch, roll, yaw commands and climb rate, otherwise the autopilot keeps leveled flight and altitude.

Besides its internal sensors, Pixhawk also reads data from external GPS and eventually other devices and transmits them to the MPU. Communication with the unit is done by serial interface using MAVLink protocol [36]. Using MAVLink messages the microcontroller reads raw sensor data and issues commands to the autopilot.

This autopilot has also native navigation algorithms and would be able to complete the whole task on its own, but in this case it is strictly used only for stabilization and reading sensor data. Nevertheless, it would be logical to completely remove it in future and implement the stabilization into the microcontroller.

6.4. Sensors

The internal sensors of Pixhawk:

- 16 bit 3D angular rate sensor ST Micro L3GD20H
- 14 bit 3D accelerometer and 3D magnetometer ST Micro LSM303D
- 3D accelerometer and gyroscope Invensense MPU 6000
- MEAS MS5611 barometer

Since there are multiple sensors of the same kind the autopilot has redundant measurements and user can choose from two different IMU messages. Their selected parameters are in table 5.

The device has also special connector for communication with a GPS unit which is in this case LS20030. It is briefly described in the next paragraph.

6.4.1. LS20030

LS20030[40] is a complete GPS receiver including embedded passive antenna and GPS receiver circuits. It can acquire up to 60 satellites and its update frequency is 1 Hz by default but can be changed up to 10 Hz. In such case either baud rate has to be raised as well or types of incoming NMEA messages reduced. NMEA is a standard of communication protocol transmitting GPS information. The communication consists of so called sentences, which are independent of each other and start with a 5-letter prefix. It contains information about the device and type of the content. Meaning of

Sensor	Feature	Value
MPU-6000 gyroscope	Full-scale range Sensitivity	± 250, ± 500, ± 1000, $\pm 2000\,^\circ\,s^{-1}$ ± 131, ± 65.5, ± 32.8, ± 16.4 LSB/°/s
MPU-6000 accelerometer	full-scale range Sensitivity	± 2, ± 4, ± 8, ± 16 g 16 834, 8192, 4096, 2048
L3GD20H gyroscope	Full-scale range Sensitivity	± 245, ± 500, $\pm 2000\,^\circ\,s^{-1}$ ± 8.75, ± 17.5, ± 70, ± 70 mdps/digit
LSM303D accelerometer	Full-scale range sensitivity	± 2, ± 4, ± 6, ± 8, ± 16 g ± 0.061, ± 0.122, ± 0.183, ± 0.732 mg/LSB
LSM303D magnetometer	Full-scale range Sensitivity	± 2, ± 4, ± 8, ± 12 G ± 0.08, ± 0.16, ± 0.32, ± 0.479 mG/LSB

Table 5. Parameters of Pixhawk's internal sensors [37] [38] [39]

all provided NMEA sentences is in the cited datasheet. The module uses serial port so UART is used for data transmission to the autopilot.

Feature	Value	
Update rate	1 Hz, up to 1 Hz	
Acquisition time (open sky)	Hot start Cold start	< 1 s (typical) 32 s (typical) without AGPS < 15 s (typical) with AGPS
Position accuracy	Autonomous SBAS	3 m 2.5 m (depends on accuracy of correction data)
Protocol support	NMEA0183 ver 3.01	9600 bps(default), 8 data bits, no parity, 1 stop bits (default)1Hz: GGA, GLL, GSA, GSV, RMC, VTG

Table 6. LS20030 parameters[40]

7. Implementation

This chapter describes models and principles used in the final filtering algorithms including steps specific for each of them. The Madgwick filter was tested just briefly, because it allows only attitude estimation and some other filter would be required anyway to incorporate position and velocity. As the most suited for the task seemed to be EKF and UKF and therefore were selected.

The implemented filters use the loosely coupled architecture, which means that the GPS and IMU measurements are independent of each other. Taking in account unreliability of the magnetometer and sources of magnetic field on the quad rotor, the only measurements used besides GPS data are from accelerometer and angular rate sensor.

7.1. Navigation equations

A model is core part of the KF. State space of the process model differs with implementations and accuracy of the system. State equitations differ with applied approximations. Depending on used sensors and required precision, multiple errors of each sensor can be modeled. Although price for more accurate model with more states is lengthy and difficult tuning. The measurement model is basically given by used sensors and their approximations. In the implemented algorithm were used slightly altered equations from [41]. Both implemented filters are done according to the steps described in chapter 4. There only few things that are specific for each and are mentioned in subsections 7.4 and 7.5.

Symbol	Meaning
$\boldsymbol{p}_{NED} = [p_N, p_E, p_D]$	Position in NED frame
$\boldsymbol{v}_{NED} = [v_N, v_E, v_D]$	Velocity in NED frame
$\boldsymbol{a}_b = [a_x, a_y, a_z]$	Specific force in body frame
$[\varphi, \theta, \psi]$	Attitude - roll, pitch, yaw
$\boldsymbol{\omega}_b = [\omega_x, \omega_y, \omega_z]$	Angular velocity
$\boldsymbol{b}_{acc} = [b_{ax}, b_{ay}, b_{az}]$	Accelerometer bias
$\boldsymbol{b}_{gyr} = [b_{gx}, b_{gy}, b_{gz}]$	Angular rate sensor bias
$\boldsymbol{p}_{GPS_{NED}} = [p_{GPS_N}, p_{GPS_E}, p_{GPS_D}]$	GPS position measurement
v_{GPS}	GPS speed over ground measurement
$\boldsymbol{f}_s = [f_{s_x}, f_{s_y}, f_{s_z}]$	Accelerometer measurement
$[\omega_x, \omega_y, \omega_z]$	Angular rate sensor measurement
T	Period
τ_{gyr}	Angular rate sensor correlation time
τ_{acc}	Accelerometer correlation time
g_N	Gravitation constant

Table 7. Table of symbols for section 7

.

7.1.1. State model

$$\boldsymbol{x} = \left[\, p_N,\, p_E,\, p_D,\, v_N,\, v_E,\, v_D,\, a_x,\, a_y,\, a_z,\, \varphi,\, \theta,\, \psi,\, \omega_x,\, \omega_y,\, \omega_z,\, b_{ax},\, b_{ay},\, b_{az},\, b_{gx},\, b_{gy},\, b_{gz} \,\right] \tag{34}$$

The state model used in the implemented filters consists of position, velocity, acceleration, attitude, angular velocity and biases of accelerometer and angular rate sensor. The set of equations 35 describes this model. And equations 36 the discrete step update.

$$\begin{aligned}
\dot{p} &= vt + \frac{1}{2}at^2 \\
\dot{v} &= at \\
\dot{a} &= -\frac{a}{\tau_{acc}} + \frac{1}{\tau_{acc}}v \\
\dot{\varphi} &= \omega_x + \omega_y \sin\varphi \tan\theta + \omega_z \cos\varphi \tan\theta \\
\dot{\theta} &= \omega_y \cos\varphi - \omega_z \sin\varphi \\
\dot{\psi} &= \omega_y \frac{\sin\varphi}{\cos\theta} + \omega_z \frac{\cos\varphi}{\cos\theta} \\
\dot{\omega} &= -\frac{1}{\tau_{gyr}}\omega + \frac{1}{\tau_{gyr}}w \\
\dot{b}_{acc} &= 0 + v \\
\dot{b}_{gyr} &= 0 + w
\end{aligned} \tag{35}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \begin{bmatrix}
T \times \boldsymbol{v}_{NED} + \frac{1}{2}T^2 \times \boldsymbol{C}_b^n \times \boldsymbol{a}_b \times g_N \\
T \times \boldsymbol{C}_b^n \times \boldsymbol{a}_b \times g_N \\
-\frac{T}{tau_{acc}} \times \boldsymbol{a}_b \\
T \times \begin{bmatrix} 1 & \cos\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \end{bmatrix} \times \boldsymbol{\omega_b} \\
-\frac{T}{\tau_{gyr}} \times \boldsymbol{\omega_b} \\
\boldsymbol{0}_{3\times 1} \\
\boldsymbol{0}_{3\times 1}
\end{bmatrix} \tag{36}$$

7.1.2. Measurement model

$$\boldsymbol{y} = \left[\, p_{GPS_N},\, p_{GPS_E},\, p_{GPS_D},\, v_{GPS},\, f_{s_x},\, f_{s_y},\, f_{s_z}\ \omega_x,\, \omega_y,\, \omega_z \,\right] \tag{37}$$

The measurement consist of position and velocity from GPS, accelerometer and angular rate sensor readings, so the measurement model is derived in the same fashion. The measurement estimation is calculated in a following way:

$$\hat{\boldsymbol{z}}_k = \begin{bmatrix}
\boldsymbol{p}_{NED} \\
\boldsymbol{v}_{NED} \\
\boldsymbol{a}_b + \boldsymbol{b}_{acc} + \boldsymbol{C}_n^b \times \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \\
\boldsymbol{\omega}_b + \boldsymbol{b}_{gyr}
\end{bmatrix} \tag{38}$$

And $\boldsymbol{C}_b^n$ is a transformation matrix from body to NED frame:

$$\boldsymbol{C}_b^n = \begin{bmatrix}
\cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\
\cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\
-\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta
\end{bmatrix} \tag{39}$$

Reverse transformation is done using transpose of this matrix.

7.2. Initiation phase

Before the take-off or launch in case of the competition, the copter is left still to lock on GPS and get its approximate position. This is taken as origin in the navigation frame, from which are calculated variables for the transformation between the two frames as shows equations 40. Then the calculation of NED coordinates from GPS is done using set of equations 41. LAT and LON stands for latitude and longitude, ALT is altitude. The subscript in stands for initial value.

$$\begin{aligned}
a &= 6378137 && \ldots \text{Earth semi-major axis (m)} \\
b &= 6356752.3142 && \ldots \text{Earth semi-minor axis (m)} \\
e &= \sqrt{1 - b^2/a^2} && \ldots \text{Earth eccentricity} \\
r_N &= a(1 - e^2)/\sqrt{(1 - e^2 \sin(\frac{\pi}{180} LAT_{in})^2)^3} && \ldots \text{Curvature radius in prime vertical} \\
r_M &= a/\sqrt{1 - e^2 \sin(\frac{\pi}{180} LAT_{in})^2} && \ldots \text{Meridian radius of curvature}
\end{aligned} \tag{40}$$

$$\begin{aligned}
p_N &= (LAT - LAT_{in})/\frac{180}{\pi} \arctan(r_N^{-1}) \\
p_E &= (LON - LON_{in})/\frac{180}{\pi} \arctan((r_N \cos(\frac{\pi}{180})^{-1}) \\
p_D &= -ALT + ALT_{in}
\end{aligned} \tag{41}$$

7.3. GPS measurement frequency

Since output rate of GPS is lower that IMU's, it is necessary to do some extra operations in steps without GPS update. Between each GPS update the navigation system si basically in dead reckoning mode. Thus during this phase biases are not updated and last 6 rows of $\boldsymbol{K}$ are zero. The same goes for the position and velocity rows in innovation vector.

7.4. EKF

7.4.1. Process covariance update

The process covariance matrix is every cycle calculated from the original matrix $\boldsymbol{Q}$ by formula 42.

$$\boldsymbol{Q}_k = 0.5(\boldsymbol{F}_{k,k-1} \boldsymbol{P_k} \boldsymbol{F}_{k,k-1}^T + Q); \tag{42}$$

7.5. UKF

7.5.1. Sigma points

To ease the computation load a little bit, cholesky factorization *chol()* is used instead of square root. This is possible because the covariance matrix is always positive definite if

the filter is stable. So the sigma points are computed using set of equations 43 from [42].

$$\begin{aligned} P_{chol} &= \mathrm{chol}(P) \\ A &= cP_{chol} \\ Y &= \begin{bmatrix} x & x & \cdots & x \end{bmatrix}_{L \times L} \\ X &= \begin{bmatrix} x & Y + A & Y - A \end{bmatrix} \end{aligned} \tag{43}$$

7.6. Trajectory

Since the UAV's air time is quite limited it would be desirable to find optimal trajectory. However multiple facts play role here. In higher altitude is lower air density. To show this on real numbers, follows comparison of point of release of the quad rotor and at the ground. In altitude of 3 km is around 44 % less than at the sea level according to model of standard atmosphere [43] as can be seen from eq 44.

Therefore thrust of the propellers significantly drops with higher elevation, which causes energy outtake to increase. From this point of view it would be for the best to release the parachute above the ground and fly horizontally. On the other hand while keeping some descend rate and moving forward, the required thrust will decrease. But when the thrust is lowered in order to slowly sink, while keeping constant attitude, the forward speed will drop as well, resulting in longer air time and again increased energy outtake. On the contrary drag of the aircraft is proportional to air density, which is another element in this equation.

Finding optimal solution for this situation would probably require knowing relevant characteristics of the quad rotor and correct differential equations describing the situation. Then the problem could be solved by some optimization method. For now is selected starting altitude 200 meters and straight flight path towards the goal.

h(m)	**T(K)**	**p(Pa)**	$\boldsymbol{\rho}$**(kg m^{-3})**
0	288.15	101,325	1.2250
500	284.90	95,461	0.9421
1,000	281.65	89,874	0.8870
1,500	278.40	84,556	0.8345
2,000	275.15	79,495	0.7846
2,500	271.90	74,682	0.7371
3,000	268.65	70,108	0.6919
3,500	265.40	65,764	0.6490

Table 8. Table of selected values from model of the standard atmosphere[43]

$$\frac{100}{\rho_{h=0}} \cdot (\rho_{h=0} - \rho_{h=3000}) = \frac{100}{1.225} \cdot (1.225 - 0.6919) \doteq 43.5\,\% \tag{44}$$

8. Simulations

For completing the task at hand two different algorithms are necessary. As the previous chapters indicate, these are navigation and control. First of all aircrafts's attitude, position and other attributes are required. Filtering of sensor data and estimating correct values is role of the navigation part and the control part is in charge of moving the quad rotor to the desired position.

Even though the autopilot can switch into an automatic mode anytime, it is necessary to test and tune the individual algorithms by simulations. This should, ease the development and at least to some extent, prevent potential collision and damage. In this chapter are described simulation tests which were done. The simulations were mostly done in Matlab [44] and Simulink [45].

8.1. Navigation

Both Extended and Unscented Kalman filters were tuned with data from a flight of different aircraft because it was not possible to obtain sensor readings on sufficient rate from autopilot logs. As reference was used system with higher grade sensors. In the two following subsections are results of these tests. Since the autopilot is professional commercial product and has implemented filtration algorithm as well, it is used as a reference in the testing process. These tests were done purely in Matlab.

8.1.1. Results

Each following section shows different aspects of the results for both filters and comments on the obtained values. From each chart was selected just a part to be showed so that the individual values can be compared. One cycle of the EKF took 0.70642 ms and UKF 1.1 ms.

Position

It is clearly visible from figure 8 that estimating of position is good for both algorithms. This is supported by calculation of RMSE and STD for each filter. These values are given in table 8.1.1

	filter	N	E	D
RMSE[m]	EKF	3.52	3.17	2.6
RMSE[m]	UKF	3.49	3.18	2.62
STD[m]	EKF	2.84	2.96	3.02
STD[m]	UKF	2.82	2.96	3.03

Velocity

In figure 9 is comparison of estimated velocities in NED frame.

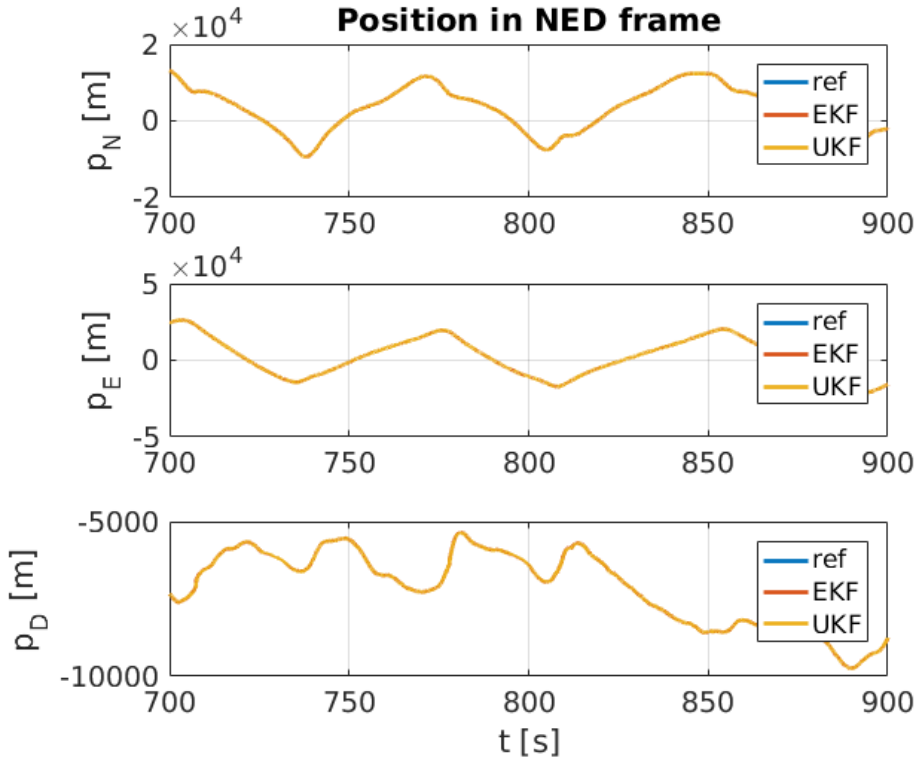


Figure 8. Position estimate comparison.

Attitude

Figure 10 shows estimation of roll pitch and yaw. RMSE and STD values are in table 8.1.1.

	filter	Φ	Θ	Ψ
RMSE[°]	EKF	2.16	3.34	5.95
RMSE[°]	UKF	2.78	3.08	6.28
STD[°]	EKF	1.98	2.61	5.15
STD[°]	UKF	2.83	5.06	8.73

Biases

Biases of the accelerometer are in figure 11 and gyroscope's in figure 12. This is an important parameter because it reflect how well is the filter tuned.

8.2. Control

For the control simulation was used Simulink model from [46]. This model consists of quad rotor model with attitude and altitude PD regulator. Since the autopilot is driven in the same fashion, it is suitable for testing control algorithms for the MCU. Designed regulators and results of the simulations are described in rest of this chapter.

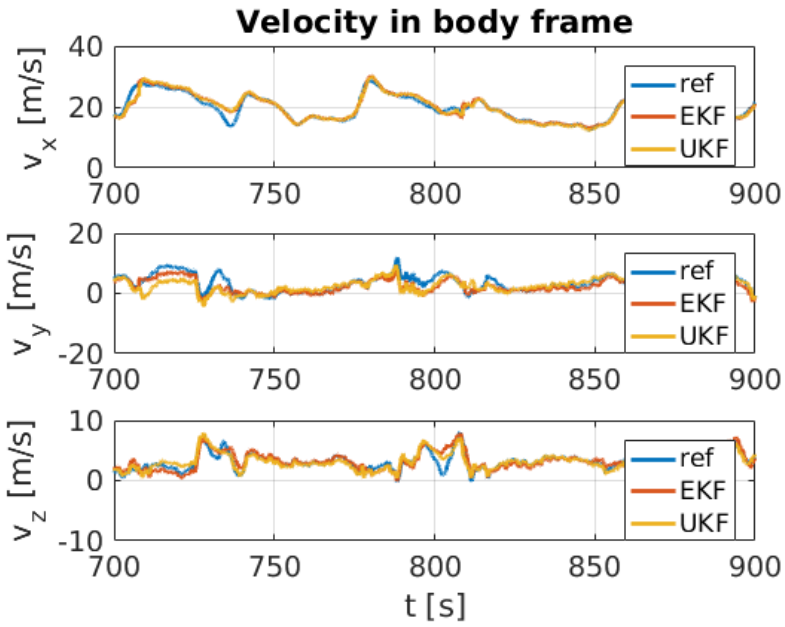


Figure 9. Comparison of estimated velocity

8.2.1. Vertical

The original PD regulator already has vertical control, but it does not bring satisfying results, so different PID regulator is used. The fact that vertical speed value is available was taken advantage of and it is used instead of derivative of the current value in the regulator. The structure can be seen in figure 13.

8.2.2. Horizontal

The horizontal position is affected by pitch and roll of the quad rotor. Because the original model has only attitude control and position control is required, it was necessary to introduce regulator from position to attitude. Direction the UAV is facing is irrelevant in this application so x and y coordinate errors can be simply conversed to pitch and roll commands. Both of these rotation angles have their own P regulator with feedback from their respective position and velocity. The whole navigation Simulink block is in figure 14.

The block *speedErrBody* simply transforms position error from navigation frame to body frame using another *n2b* block which uses formula 45. *speedSat* block only saturates maximum angle command.

$$\begin{bmatrix} x \\ y \end{bmatrix}_{body} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_{navigation} \tag{45}$$

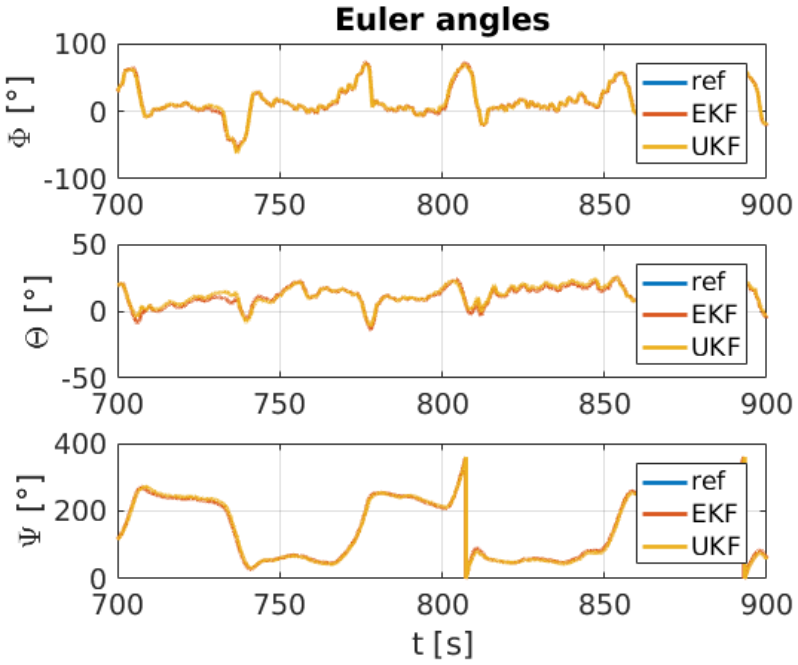


Figure 10. Comparison of estimated attitude

8.2.3. Tests

Since the route is arbitrary, one of the options, which would suffice for the competition, was used for the test. The reference and actual coordinate does not match until reaching the goal, because the immediate reference position in every moment is chosen some distance in front of the UAV along the planned path. The chosen path goes along the X axis from 1500 to 10 meters under -45°. The initial point is considered as the moment when the parachute is thrown away. The results for all axis are shown in figure 15 and altitude profile of the trajectory is in figure 16.

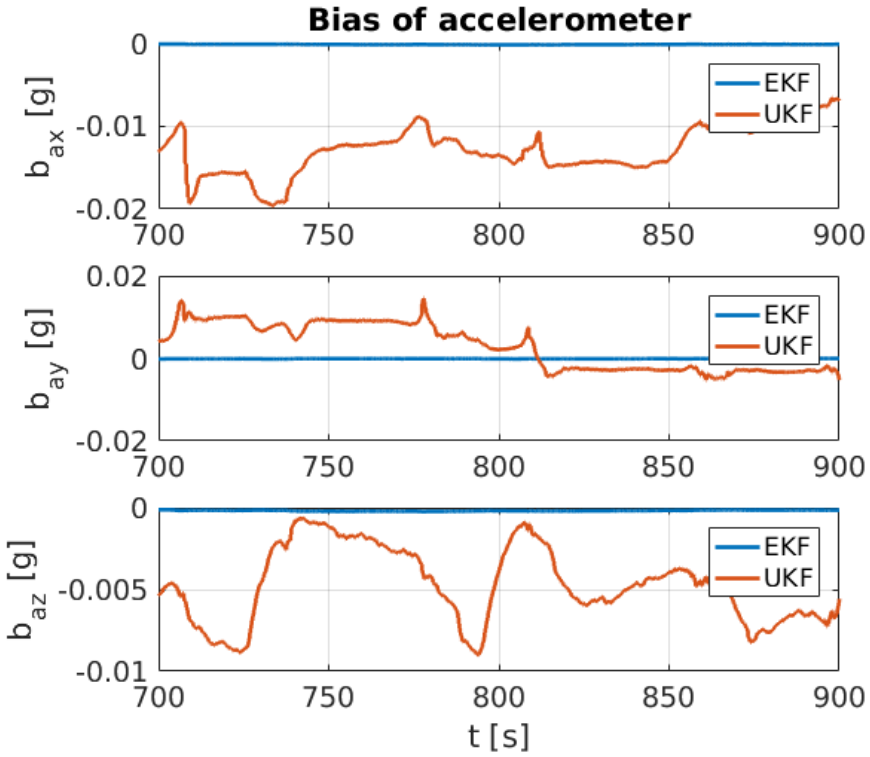


Figure 11. Biases of accelerometer

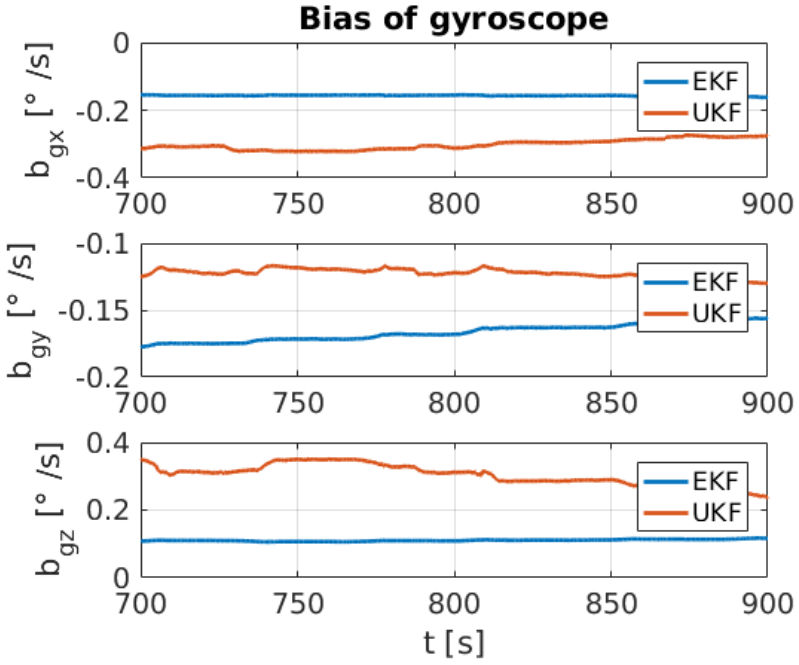


Figure 12. Biases of gyroscope

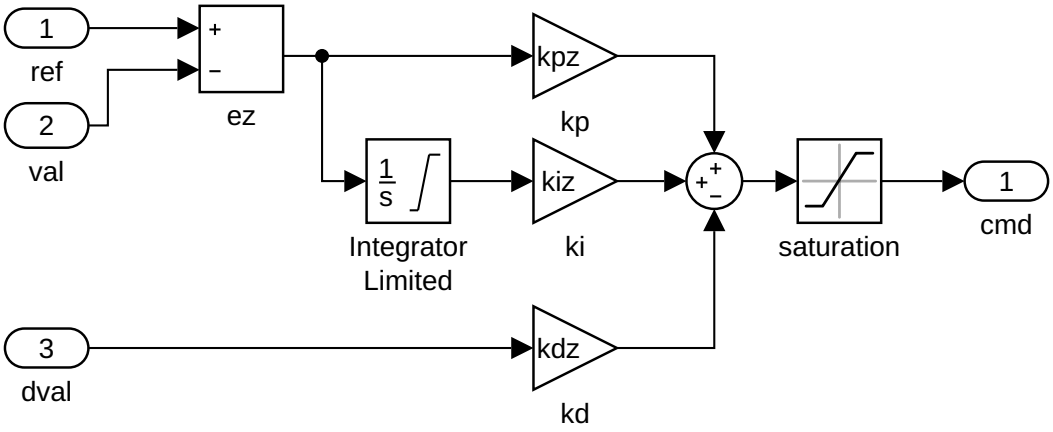


Figure 13. Vertical position regulator

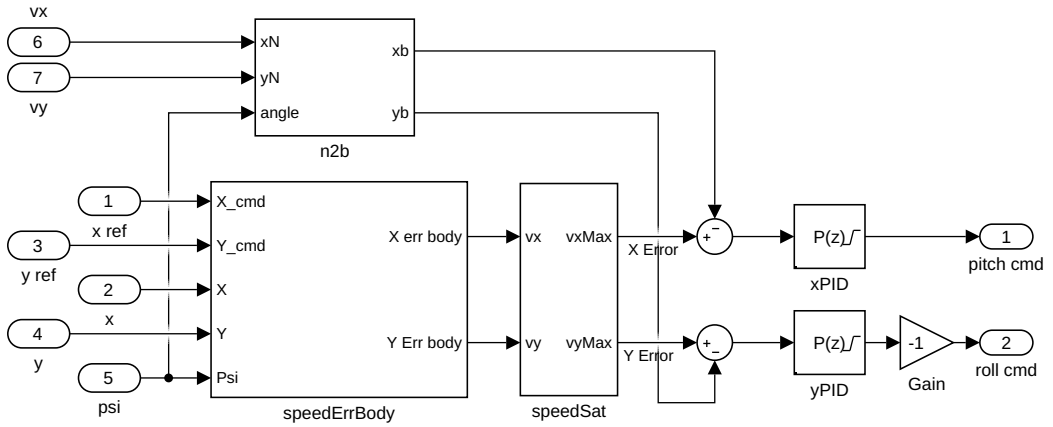


Figure 14. Horizontal position to attitude regulator

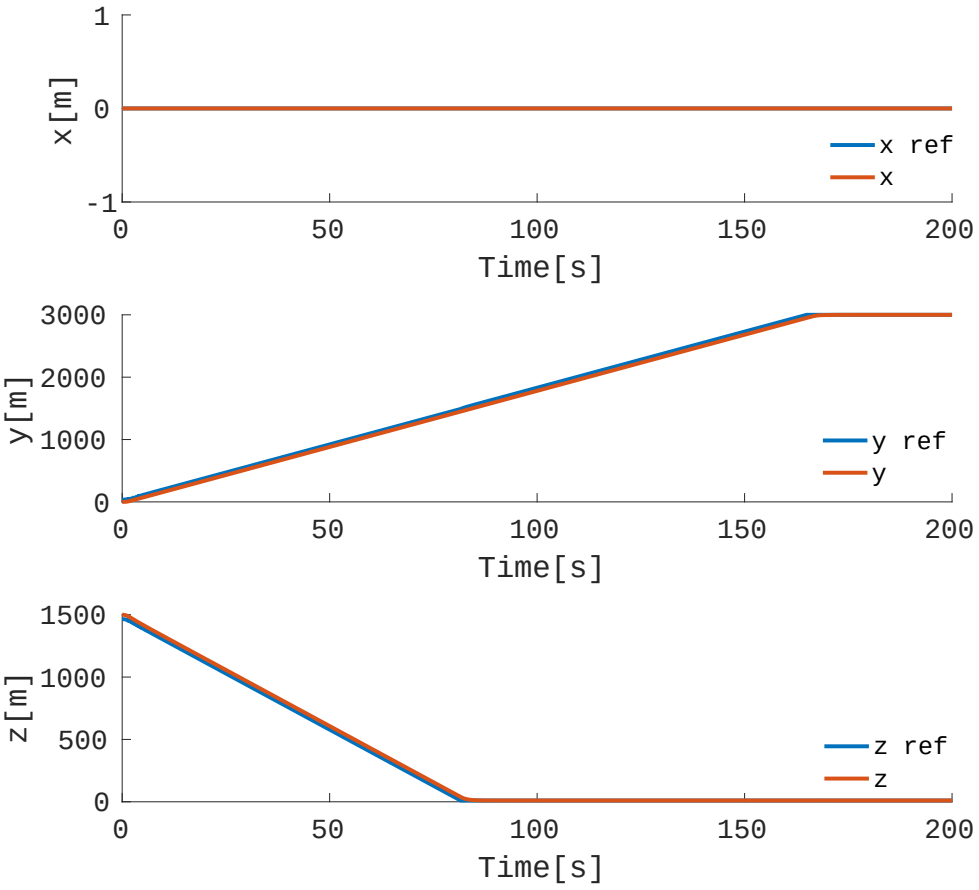


Figure 15. Graph of references and actual values for all axis

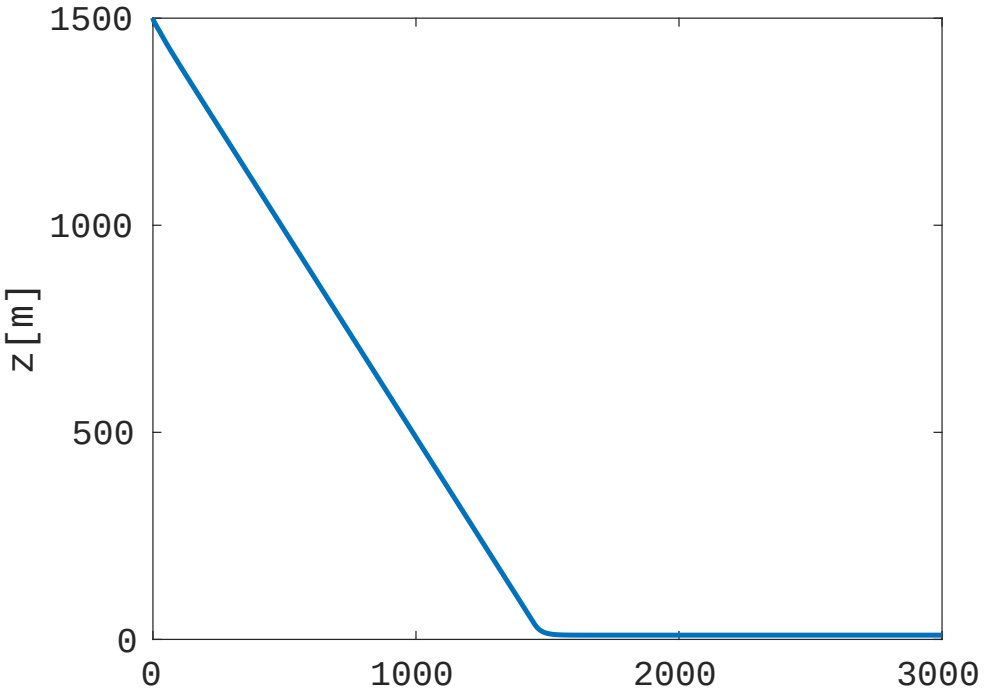


Figure 16. Z-Y view of the trajectory

9. Algorithm

Both EKF and UKF are implemented for the target platform – STM32f401. Since it is 32-bit MCU the obvious choice is C/C++. Though C is lower-level language and would be maybe more suitable for such application, C++ was chosen because it has better library support for matrix calculations, specifically Eigen [47]. Template of the project was generated by STM's CubeMX [48] which uses HAL driver. It is a low level hardware abstraction layer, which provides implementation of most of the necessary functions for using the MCU and its peripherals. It should also ease portability of the code.

9.1. MCU setup

While using embedded systems such as this one, the actual implementation precedes configuration of the device and its peripherals. In this section is laid out the applied setup.

9.1.1. Clock

As a source of clock is used on-board external 8 MHz crystal. Then the internal PLL and prescalers are configured so that the system clock is 72 MHz as well as all timers and peripheral's clock, excluding *APB1* which runs on a half of the frequency.

9.1.2. UART

Communication with the autopilot is done via serial interface, so UART2 device is configured to handle this. It is in Rx/Tx 8n1 mode with baud rate 57 600 bit s^{-1} using pins PA2 and PA3. To be able to continually transmit and receive messages in non-blocking mode, global interrupt is enabled. Another uart device UART6 is configured for eventual data output in transmit mode and baud rate 9600 bit s^{-1}.

9.1.3. Timer

Since the autopilot needs to receive heartbeat messages to keep the connection alive, timer TIM3 is configured to trigger interrupt every second. To obtain this behavior its prescaler's value is set to 36000 and it counts up to 1999.

9.2. Classes

There were implemented 3 main classes with key functionalities. Them being *Telemetry*, *KF* and *Control* and each of them plays major part in the algorithm.

9.2.1. Telemetry

This class handles communication with the autopilot. It keeps configuration of the ongoing MAVlink communication and other variables necessary for incoming data. Besides just reading the messages also provides methods for preparing data structures for subsequent functions and encoding commands generated by the *Control* class into MAVLink message and sending it back to the autopilot. It also implements method for arming, changing mode and sending the heartbeat which is being called from handler of the TIM3 interrupt.

9.2.2. KF

Base class for Kalman filter which implements methods and variables common for *EKF* and *UKF* which subclass it. It also defines structures and methods required for transforming coordinates between NED and ECEF frame. The main method here is pure virtual function *update()* which differs for the two derived classes *EKF* and *UKF*.

EKF

Besides *update()* function implements methods for calculating jacobians of the process and measurement equations.

UKF

This class has four member functions – *update()*, one for calculating sigma points and the other two for unscented transformation of process and measurement sigma points.

9.2.3. Control

The *Control* class defines important structure *PID* which represents real discrete PID regulator, although all components does not need to be used. Instance of this structure is created for each axis in a way similar to the design used in simulation. In the update step of the control part of the algorithm are then calculated new values for every regulator.

9.3. Main

Besides the two mentioned interrupt handlers, *main()* function controls the whole flow of the program. In the beginning the peripherals and interrupts are configured. Then required variables are defined and initialized afterwards which an infinite while loop is entered. In there, 4 main steps are taken. At first is created measurement vector from data which where decoded in the UART interrupt. These are passed to either EKF or UKF instance, depending on the configuration. State vector with position, attitude and velocities calculated there is then handed over to the control part. Command calculated by *Control* class instance is then converted by *Telemetry* and send via uart back to the autopilot. The cycle then repeats itself as is described in figure 17.

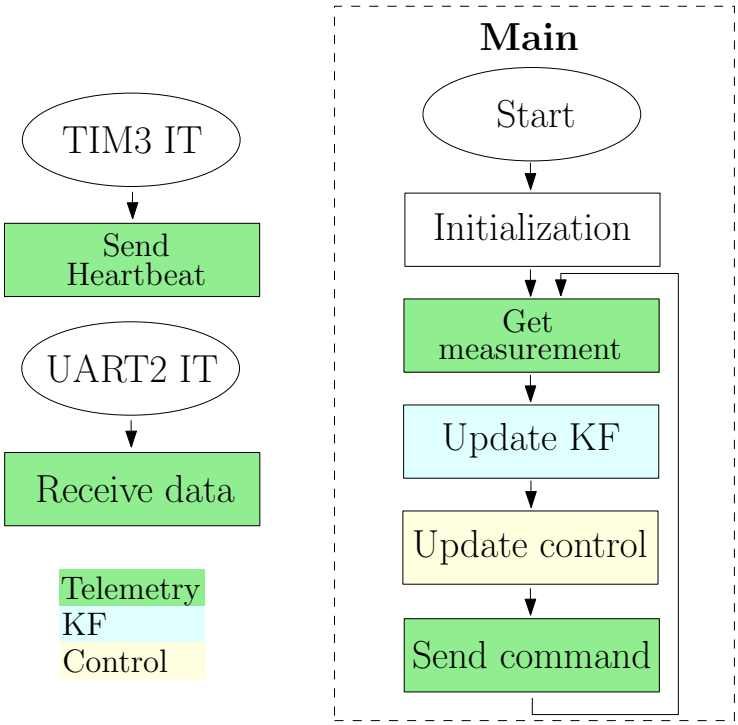


Figure 17. Horizontal position to attitude regulator

10. Conclusion

In the introductory theoretical part of this thesis was given overview of the problematics connected to navigation of autonomous UAVs. Further were enumerated various past as well as recent navigation methods and latest trends in this field. As a preliminary to actual filtering algorithms were briefly mentioned sensor errors and their compensation.

As a background and motivation for this topic in the first place was in chapter 5 introduced the ARLISS competition from which the practical part of the assignment originates. First of all was laid out selected hardware configuration which is supposed to be used. This includes selection of microcontroller STM32F401 as the navigation unit and autopilot Pixhawk for stabilization of the quad rotor and sending sensor readings to the MCU.

The main part of this thesis is concerned with filtering methods. There was given detailed description of Kalman filter and its descendants Extended Kalman filter and Unscented Kalman filter. Explanation of Madgwick filter was given as well though this algorithm was not implemented beyond some basic tests because it only estimates attitude. On the other hand EKF and UKF were tested by simulations and compared to each other. In the end of the thesis is described implementation of the designed algorithms for the target platform. As a complement to the code was generated PDF and HTML page with documentation. This should ease eventual continuation on this work.

Up to the last task was the implementation successful. However the implemented algorithms are supposed to be verified by actual test flights. This did not happen due to multiple problems, which appeared just in the final phase of writing of the thesis. First was caused by Linux version of APM Planner [49]. This is a software for configuration of the autopilot and can as well work as a ground control station. Even thought being the latest version, the program stopped displaying parameters of the unit and its configuration. Also flashing of new firmware was failing from the begging, so in case something went wrong, it was necessary to use another computer. This obstructed preparation for the testing. Next problem was actual sending commands to the autopilot, which did not react to most of them for some reason. Third problem turned out to be insufficient output frequency of measured values. Although the used baud rate should be capable of handling such data stream when unnecessary messages are disabled and it is stated multiple times by other users that it is possible to set the output rate up to 100-200 Hz, the maximum which was ever measured is around 25Hz.

Nevertheless part of the communication with Pixhawk was successful and API for finishing this step was prepared. It is now possible to send heartbeat, the autopilot got armed by the algorithm so it should be ready for a take off and mechanism for reading MAVLink messages works well too.

10.1. Results

In chapter 8 were given RMSE and STD values for attitude and position estimations for each filter. Even though UKF is supposed to have lower errors, EKF brings better

results here. This might be also caused by the algorithm being better tuned. But not only higher errors, 1 UKF cycle also took approximately 0.3936 seconds longer than EKF, which might matter in a long run.

The estimated velocity is clearly not as accurate as attitude and position, but it is not as important and it is more difficult to estimate because it is integrated by accelerometer and GPS provides only speed over ground. From the biases it is visible that even in this area shows EKF better results, though not so significantly, but it might be reason why does it give better accuracy.

10.2. Future work

Since there appeared a problem with sensor data rate from the autopilot, the data should be read directly from external sensors. This will require new IMU, but its output rate will be more than sufficient and under direct control of the MCU. Another suitable step would be replacing the universal development kit STM32F401VC6T with a custom minimalistic board. For the quad rotor to be really complete solution from scratch it will be necessary to get rid of the autopilot and implement stabilization directly to the main program. This a large task but it will make the device much more compact since everything will be on only one board and the project will not depend on external tools such as the APMPlanner.

But regarding the nearest feature, it is important to get proper sensor readings and overcome the problem with controlling the autopilot so that the quad rotor is ready for the competition.

Appendix A.

CD content

Since main part of this thesis is computer code and data, everything was put in an attached CD. This appendix lists its structure and contents. This is list is not complete directory tree but gives narrowed structure and description of each directory.

- sekanina.pdf ... This thesis
- simulations ... Data and code used in simulations
- code ... Source and header file for the C++ project
- docs ... Directory with documentation to the code
  - doc.pdf ... Documentation as pdf
  - HTML ... Documentation as HTML
- copter401.ioc ... CubeMC project with configuration of the board

Bibliography

[1] *ARLISS A Rocket Launch for International Student Satellites*. 2017. URL: http://www.arliss.org/.

[2] *DICEbot*. 2017. URL: http://control.fs.cvut.cz/dicebot.

[3] *Remote Piloted Aerial Vehicles : An Anthology*. Feb. 2003. URL: http://www.ctie.monash.edu/hargrave/rpav_home.html#Teleautomata.

[4] Paul Gerin Fahlstrom and Thomas James Gleason. *Introduction to UAV systems*. 2012.

[5] G. A. Steinlage. "Autonomous aerial vehicle development". In: *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference@m_NAECON 1992*. May 1992.

[6] Daniela Bleichmar et al. *Science in the Spanish and Portuguese Empires, 1500-1800*. Dec. 18, 2008.

[7] *Inertial Measurement Units and Inertial Navigation*. URL: http://www.vectornav.com/support/library/imu-and-ins (visited on 30/03/2017).

[8] Riccardo Costanzi et al. "An Attitude Estimation Algorithm for Mobile Robots Under Unknown Magnetic Disturbances". In: *IEEE/ASME TRANSACTIONS ON MECHATRONICS* (Aug. 2016).

[9] Farid Kendoul, Isabelle Fantoni, and Kenzo Nonami. "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles". In: *Robotics and Autonomous Systems* (Feb. 20, 2009).

[10] Stefan Hrabar and Gaurav S. Sukhatme. "A Comparison of Two Camera Configurations For Optic-Flow Based Navigation of a UAV Through Urban Canyons". In: (2004).

[11] Stefan Hrabar et al. "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". In: (Aug. 2005).

[12] *A stereo and rotating laser framework for UAV navigation in GPS denied environment*. IEEE, Oct. 2016.

[13] Jan Roháč. *Lecture Reference frames. Principles of navigation. Attitude representation*. 2017.

[14] James Diebel. "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors". Stanford University, Oct. 20, 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.5134&rep=rep1&type=pdf (visited on 06/27/2016).

[15] Basic Air Data. *NED frame image*. URL: http://www.basicairdata.eu/knowledge-center/background-topics/coordinate-system/.

[16] Saurabh Godha. "Performance Evaluation of Low Cost MEMS-Based IMU Integrated With GPS for Land Vehicle Navigation Application". UCGE Report. Schulich School of Engineering, Jan. 2006.

[17] Valérie Renaudin, Muhammad Haris Afzal, and Gérard Lachapelle. "Complete Triaxis Magnetometer Calibration in the Magnetic Domain". Schulich School of Engineering, University of Calgary, Oct. 2010.

[18] Walter T. Higgins, Jr. "A Comparison of Complementary and Kalman Filtering". Arizona State University, Aug. 1974.

[19] Simon Haykin. *Kalman Filtering and Neural Networks*. John Wiley & Sons, 2001. Chap. 1. ISBN: 0-471-22154-6.

[20] Simon J. Julier and Jeffrey K. Uhlmann. "Unscented Filtering and Nonlinear Estimation". In: *Proceedings of the IEEE* (Nov. 4, 2004).

[21] Eric A. Wan and Rudolph van der Menve. "The unscented Kalman filter for nonlinear estimation". In: (Oct. 4, 2002).

[22] Simon Julier and Jeffrey K. Uhlmann. "A General Method for Approximating Nonlinear Transformations of Probability Distributions". University of Oxford, Nov. 1, 1996.

[23] S. Kolås, B.A. Foss, and T.S. Schei. "Constrained nonlinear state estimation based on the UKF approach". In: *Computers and Chemical Engineering* (Mar. 21, 2008).

[24] Weiguang Yang Yuanxi; Gao. "An Optimal Adaptive Kalman Filter". In: *Journal of Geodesy* (July 2006).

[25] Leopoldo Jetto, Sauro Longhi, and Giuseppe Venturini. "Development and Experimental Validation of an Adaptive Extended Kalman Filter for the Localization of Mobile Robots". In: *IEEE Transactions on Robotics nad Automation* (Apr. 1999).

[26] Jianhua Cheng et al. "An Adaptive Unscented Kalman Filtering Algorithm for MEMS/GPS Integrated Navigation Systems". In: *Journal of Applied Mathematics* (Nov. 13, 2014).

[27] Sebastian O.H. Madgwick. *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*. Apr. 30, 2010. URL: `https://www.samba.org/tridge/UAV/madgwick_internal_report.pdf` (visited on 06/27/2016).

[28] *Photography from ARLISS competition*. 2017. URL: `http://control.fs.cvut.cz/dicebot/wp-content/uploads/2015/05/cropped-cropped-IMG_20140910_1240051.jpg`.

[29] *Raspberry Pi*. May 6, 2016. URL: `https://www.raspberrypi.org/`.

[30] *BeagleBone Black*. May 7, 2016. URL: `https://beagleboard.org/black`.

[31] *Arduino*. May 6, 2016. URL: `https://www.arduino.cc/`.

[32] *STM32*. May 6, 2016. URL: `http://www2.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus.html`.

[33] STMicroelectronics. *DS9716: ARM©Cortex©-M4 32b MCU+FPU, 105 DMIPS, 256KB Flash/64KB RAM, 11 TIMs, 1 ADC, 11 comm. interfaces*. Version 5. STMicroelectronics. Aug. 2015. URL: `http://www2.st.com/content/ccc/resource/technical/document/datasheet/9e/50/b1/5a/5f/ae/4d/c1/DM00086815.pdf/files/DM00086815.pdf/jcr:content/translations/en.DM00086815.pdf` (visited on 04/30/2015).

[34] *Arduino Zero*. May 7, 2016. URL: `https://www.arduino.cc/en/Main/ArduinoBoardZero`.

[35] *Pixhawk Autopilot*. URL: https://pixhawk.org/modules/pixhawk (visited on 05/07/2017).

[36] *MAVLink Micro Air Vehicle Communication Protocol*. URL: http://qgroundcontrol.org/mavlink (visited on 05/07/2017).

[37] *L3GD20H*. 2017. URL: http://www.st.com/en/mems-and-sensors/l3gd20h.html.

[38] *LSM303D*. 2017. URL: http://www.st.com/en/mems-and-sensors/lsm303d.html.

[39] *MPU-6000*. 2017. URL: https://store.invensense.com/ProductDetail/MPU6000-InvenSense-Inc/420595/.

[40] *Datasheet of GPS smart antenna module, LS20030 3*. 79. LOCOSYS Technology Inc. 2006. URL: http://www.locosystech.com/product.php?zln=en&id=20 (visited on 05/07/2016).

[41] Jan Roháč. *Enhanced navigation solution for low-cost navigation units*. 2017.

[42] Yi Cao. *Learning the Unscented Kalman Filter*. Dec. 12, 2010. URL: https://www.mathworks.com/matlabcentral/fileexchange/18217.

[43] Egbert Torenbeek. Appendix B - International Standard Atmosphere. In: *Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes*. May 27, 2013.

[44] *Matlab*. May 7, 2017. URL: https://www.mathworks.com/products/matlab.html.

[45] *Simulink*. May 7, 2017. URL: https://www.mathworks.com/products/simulink.html.

[46] Abdel-Razzak Merheb. *PD Control Quadrotor - Simulink*. Apr. 10, 2014. URL: https://www.mathworks.com/matlabcentral/fileexchange/41149-pd-control-quadrotor-simulink.

[47] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[48] Gaël Guennebaud, Benoît Jacob, et al. *STM32CubeF4*. http://www.st.com/en/embedded-software/stm32cubef4.html. 2017.

[49] *APM Planner 2*. 2016. URL: http://ardupilot.org/planner2/.