NTNU
Norwegian University of
Science and Technology

Anti-Collision System for Multi-Rotor Drones Operating Close to Obstacles using Radar Data

Thomas Frimann Koren

Master of Science in Cybernetics and Robotics
Submission date: July 2017
Supervisor: Thor Inge Fossen, ITK
Co-supervisor: Tor Arne Johansen, ITK
Kristian Klausen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Thomas Koren
Department:	Engineering Cybernetics
Thesis title:	Anti-Collision System for Multi-Rotor Drones Operating Close to Obstacles using Radar Data

Thesis Description:

New developments in radar technology have enabled smaller devices that are feasible to mount on small Unmanned Aerial Vehicles (UAVs). These devices can provide relative measurements of nearby objects, providing means to do object detection and collision avoidance for multirotor type UAVs.

The goal of the project is to evaluate the data gathered from the radar system XeThru from Novelda, and implement collision-avoidance algorithms that use the provided data.

The following items should be considered:

1. Discuss the advantages and disadvantages to using radar technology for this purpose, relative to other techniques such as LIDARs, image recognition, ultrasonic rangefinders and 3D-camereras (e.g. Intel Real-Sense)
2. Design and construct a test-rig for the radar, which can provide measurement data with a reference. Consider mounting the radar on a rotating surface.
3. Investigate and discuss how to process the data for use in collision avoidance.
4. Investigate and test the use of the Null-Space Behavioral Control (NSB) methodology for collision avoidance for multirotors.
5. Develop and implement interface drivers for the radar to DUNE.
6. Design a radar hardware module to be mounted on the laboratory UAV.
7. Conclude findings in a report. Include Matlab/C-code as digital appendices together with a user guide.

Start date:	2017-01-23
Due date:	2017-07-03
Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Kristian Klausen, Dept. of Eng. Cybernetics, NTNU

Abstract

This thesis concerns the development of a radar-based collision avoidance system for multirotor vehicles. The first chapter is a small introduction to radar systems, to aid in understanding the rest of this thesis. Further on an embedded radar system is developed using the *X2M200* radar system from *Novelda*. Data from the embedded radar system is then tested on a object detection algorithm which looks for blobs (connected areas with several reflections). Cylinders are fitted to the detected blobs and used in a collision avoidance algorithm based on Null-space-based behavioral control from [Antonelli, Arrichiello, and Chiaverini 2008b]. The software is verified with a point mass simulator and also tested on a quadrotor simulator.

Unfortunately did not time allow for the entire system to be tested together, but each part of the system is tested separately and verified to work.

Sammendrag

(Norwegian translation of the Abstract)

Denne masteroppgaven omhandler utviklingen av et radarbasert kollisjonsunngåelsessystem for multirotorfartøyer. Det første kapittelet gir en kort introduksjon til radarsystemer som et hjelpemiddel i å forstå resten av oppgaven. Videre blir et innvevd radarsystem utviklet ved bruk av *X2M200* radaren fra *Novelda*. Data fra det innvevde radarsystemet blir så testet på en objektoppdagelsesalgoritme som ser etter klumper (sammenkoblede områder med flere refleksjoner). Sylindere blir tilpasset de oppdagede klumpene og brukt i en kollisjonsunngåelsesalgoritme basert på nullrombasert oppførselskontroll fra [Antonelli, Arrichiello, and Chiaverini 2008b]. Programvaren er verifisert med en punktmassesimulator og også testet på en quadrotorsimulator.

Uheldigvis var det ikke nok tid til å teste hele systemet sammen, men hver del av systemet har blitt testet hver for seg og verifisert at fungerer.

Acknowledgments

First I want to extend a thank you to my girlfriend Shubhaangi as well as my family, who has supported me and motivated me to keep on working.

I would also like to thank the Centre for Autonomous Marine Operations and Systems (AMOS) at NTNU for support with equipment and funding, this thesis would not have been possible without them.

The workshop at the Department of Engineering Cybernetics has been invaluable to me, aiding me with production of circuit boards and a test platform, and also deserve a big thanks.

At last I want to thank my supervisors, Thor Inge Fossen, Tor Arne Johansen, and Kristian Klausen for their guidance as well as being tolerant to my sometimes stupid questions.

Thomas Frimann Koren

Contents

List of Figures

List of Tables

1 | Introduction

1.1 Background and Motivation

This thesis lays the foundation for a radar-based collision avoidance system for multirotors. Such a system could be an integral part of a completely autonomous system. Autonomous multirotors could be used for inspection in hazardous environments, or in places which are hard to reach by other means. It can also be useful during search and rescue operations, helping rescue personnel locate a missing person faster and without themselves getting in danger.

The system could also be a part of a semi-autonomous multirotor, overriding the pilot input if the multirotor is on a collision course with an obstacle. This is particularly useful if the operation is performed beyond line of sight of the pilot. It is also imaginable that the system can be locked to a certain obstacle, in order to keep a set distance from e.g. a wall. A camera could then be pointed at the wall in order to create a high resolution panorama that could be used for inspection.

The results of this thesis will be used by the startup company SCOUT, which is funded by the Research Council of Norway. The startup aims to build a multirotor solution for industrial inspection. The inspection might be held in locations which are difficult or hazardous for humans, and such a multicopter solution would greatly reduce cost and efficiency.

1.2 Previous Work

With the advent of multirotor Unmanned Aerial Vehicles (UAVs) and increase in processing power of embedded devices, there has been done a lot of work on making multirotors autonomous, where collision avoidance plays a big role. In order to map obstacles both [Yoo, Won, and Tahk 2011] and [Nolan et al. 2013] utilize a camera and optical flow algorithms with a monocular camera. In [Yoo, Won, and Tahk 2011] the camera view is split into a left and right side and the multirotor turns towards the side with the least optical flow. [Nolan et al. 2013] is instead using the optical flow data to create a probabilistic 3D occupancy grid, which contain the likelihood of a grid being occupied. This is further built upon by [Nieuwenhuisen and Behnke 2015], where an optimal path to a point is updated as frequently as the occupancy grid is updated. The optimal path is calculated in three steps; first a coarse path through the grid is

calculated with the A* algorithm, then approximations of the missing dimensions (e.g. velocities) are calculated before the trajectory is smoothed out by optimizing w.r.t. input.

An alternative collision avoidance scheme is detailed in [Bareiss, Bourne, and Leang 2017]. Using a lidar the immediate area around the multirotor is mapped. The presented algorithm predicts the multirotor trajectory using the pilots current input over a time horizon. If the probability of a collision is higher than a certain threshold, a change in input is calculated by finding the smallest change in input that will reduce the chance of a collision through an optimization algorithm.

The work in this thesis is built on an entirely different framework of collision avoidance, Null-space-based behavioral control (NSB). NSB was originally introduced in [Antonelli, Arrichiello, and Chiaverini 2005] and divides behavior into several small tasks represented by a task function. The aim of each task is to keep the task function at the same value, which is achieved by each task removing the component from the input that will cause a task violation and adding a component to the input that increase fulfillment of a task. The approach was proven to be stable in [Antonelli, Arrichiello, and Chiaverini 2008a]. In addition to collision avoidance, NSB has also been used for formation control as in [Rosales, Leica, and Scaglia 2016]. An extension to NSB, which aimed to allow the value of the task function to be within a certain set, was presented in [Moe et al. 2016].

Radar systems have been used extensively for autonomous operation of cars, such as for the cars produced by Tesla [The Tesla Team 2016]. Tesla utilize fleet wide learning so that all cars cooperate on mapping the environment and over time reduce false positives. In [Engels et al. 2017] it was concluded that radar systems are a key component in autonomous cars.

As radar systems have become smaller over the past few years, coupled with an enormous increase in computing power, it is becoming feasible to create radar systems for multicopters as well. This was done in [Weixian et al. 2016], where a radar system mounted to a quadrotor was used to obtain high resolution images using Synthetic Aperture Radar techniques. The main breakthrough of this paper is removing the effects the minute movements of the quadrotor has on the data.

A radar system for multicopter collision avoidance was developed in [Moses et al. 2014], where the system could detect other aerial vehicles. The system could only measure distances along one axis and collision avoidance itself was only quantitatively discussed. More recently a radar-based collision avoidance system for multirotors has been announced, the Aerotenna µSharp [Aerotenna 2016], which consists of several antennas in order to survey a 360° perimeter.

1.3 Contribution and Scope of this Thesis

The goal for this thesis is to develop a radar-based collision avoidance system for multirotors, which can be used as a subsystem in an either semi or fully autonomous multirotor system. Primarily this can be thought of as two separate tasks, obstacle detection and obstacle avoidance, and is treated as such. The obstacle detection is

implemented separately from the obstacle avoidance and both can be exchanged without affecting the other. The embedded radar system developed in this thesis builds upon a prototype developed in [Koren 2016].

An embedded radar system, which is responsible for the obstacle detection, covers a 360° degree perimeter around the multicopter. The data from the embedded radar system is then searched for possible obstacles.

The collision avoidance system will detect when it is about to collide with an obstacle and adjust the input as to avoid a collision. The software is developed so that it can easily be extended for any kind of obstacle detection system.

During development, some assumptions were made in order to limit the scope of the thesis:

- The multirotor is level at all times, so that the pitch and roll does not have to be taken into account when processing the radar data.
- All detectable obstacles can be encapsulated by a vertical cylinder.

Unfortunately time did not allow for the different systems to be integrated properly with each other. At the end of this thesis we are left with an embedded radar system, a obstacle detection algorithm for data from the radar system, and a robust collision avoidance system.

1.4 Organization of this thesis

An introduction to radars, which is necessary for understanding the system, is given in Chapter 2. The radar equation, antenna design and pulse-doppler radar processing are among the topics that are covered. Additionally there is a short segment comparing radar to other possible obstacle detection sensors.

Chapter 3 concerns the development of the hardware and software for the embedded radar system. The design process behind the circuit board, as well as the embedded software is discussed. At last there is a section discussing how the system was tested.

In Chapter 4 we discuss two approaches to object detection for radars, one which is concluded to be not usable for the developed radar system. The other approach is a novel approach which is based around locating connected regions (blobs) in the data output from the radar. The blob detection algorithm is then tested against data from the embedded radar system and proven to be able to detect obstacles.

The next chapter, Chapter 5, details the development of the collision avoidance system. The system is then developed with the aim of being extendable should there be any need for additional behaviors in the future. The software itself is verified on a simple point mass simulator, and then tested to its limits on a quadrotor simulator. Additionally two subsets of the collision avoidance algorithm are implemented and tested against each other.

Chapter 6 provides a summary of the results, as well as a closing discussion.

A user manual for the system is provided as an appendix, and all software along with schematics and blueprints for the hardware are provided as digital appendices.

2 | Introduction to Radar Systems

An understanding of the purpose of radars can be done by looking at the meaning of the word; *RAdio Detection And Ranging* [Skolnik 2001, p. 2]. This describes the main purpose of a radar — to locate and obtain information about a target. A radar system achieves this goal by reflecting radio waves of a target, picking up the reflections and reading information off the returned wave. Through a variety of techniques, such as utilizing the Doppler shift to measure the radial velocity of a target, to modulating the radar signal for higher radial resolution, radar systems can be tailor-made for a specific application. Nonetheless the key principle of reflecting radio waves remain, and understanding the construction of a radar system, as well as a rudimentary understanding propagation and reflection of electromagnetic waves are vital for utilizing a radar system to its fullest.

Contents

2.1 Radar Systems

Since the beginning of radars in the early 20$^{\text{th}}$ century, through a rapid evolution and deployment of radar systems in the second world war, we have arrived at the age of modern embedded radar systems. However, common across all radar systems is that they can be divided into three parts, a transmitter, a receiver and an antenna system [Skolnik 2008, p.1.1]. This section will discuss these three parts in more detail, as well as cover some of the basics of how a radar sees a target. At the end there will be a short introduction to the particular radar system used in this thesis, the *Novelda X2M200*.

2.1.1 The Building Blocks of a Radar

As mentioned earlier, a radar can be divided into three, the receiver, the transmitter and the antenna system. The transmitter is responsible for generating the signal, the signal is then transmitted by the antenna system, before an eventual reflection is picked up by the antenna system and transmitted to the receiver. Customization of these three stages alter the characteristics of the radar so it can be used to obtain the desired information about a target.

Transmitter

The transmitter is responsible for generating waveform that will be transmitted by the radar, usually with a frequency in the range of 300 MHz to 300 GHz. The transmitter may vary in complexity, from generating a simple Continuous Wave (CW) signal or pulsing the signal on and off with a specific Pulse Repetition Frequency (PRF), to more complex modulation patterns. The signal is usually generated with a Local Oscillator (LO) that is shared between the transmitter and receiver. [Kinzie 2011, pp. 2–8]

Receiver

The receiver is responsible for detecting the reflected signal, which is often done by correlating the received signal with the LO reference signal. The receiver can usually measure the amplitude, phase and travel time of a reflection. Moreover, the received signal might be integrated over several periods to remove random background noise. The ability of the receiver to pick up a reflected signal is defined by the ability to differentiate the signal from the background noise, also referred to as the minimum Signal to Noise Ratio (SNR). [Kinzie 2011, pp. 11–12]

Antenna system

There are different kinds of antenna systems, but in general they can be categorized into three types according to [Kinzie 2011, p. 10], such as:

1. Monostatic — the same antenna is used for receiving and transmitting.

2. Bistatic — separate antennas for receiving and transmitting
3. Multistatic — several antennas used for transmitting and receiving.

On top of this, the antenna might be directional or omnidirectional. Whereas a directional antenna will be more effective in one direction, an omnidirectional will transmit and receive from all directions. The topic of antennas are covered to a greater extent in section 2.1.2.

2.1.2 The Radar Equation

As with most things, we are able to describe the behavior of a radar system through mathematics. Of special interest is the equation known as "The Radar Equation", which can be used to calculate the received signal strength of a radar target. There are different versions of the radar equation depending on the use case of the radar. A general version, found in [Skolnik 2008], is:

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma F^4}{(4\pi)^3 R^4} \tag{2.1}$$

where

- P_r and P_t is received and transmitted power respectively.
- G_r and G_t is the receiver and transmitter antenna gain respectively.
- λ is the transmitted wavelength.
- σ is the radar cross section, also called the scattering coefficient.
- F is the pattern propagation factor, i.e. a loss factor due to multipath loss, antenna pattern loss etc.
- R is the distance between the radar and the target (assuming that the transmitter and receiver are located at the same location).

All of these terms are mostly self-explanatory, except for the radar cross section σ and the antenna gain, which we will look a bit further upon.

Radar Cross Section

The radar cross section is a measure of how much the incident radar waves are reflected back by the target in the direction of the radar. It is given in square meters (m^2) and is the cross-sectional area of a sphere that would induce a reflection of the same strength. The radar cross section is close to impossible to calculate explicitly, and usually is found either through experiments or computer simulations. [Skolnik 2001, pp. 49–50] There are three main factors that contribute to the radar cross section:

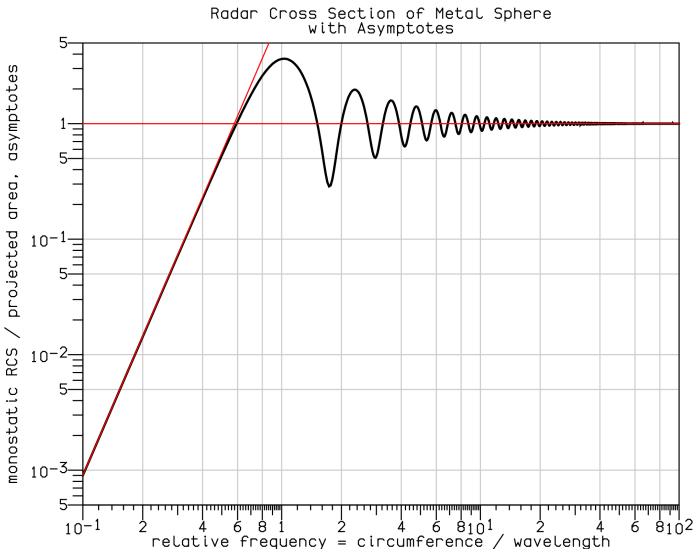


Figure 2.1: The normalized cross section of a sphere as a function of its circumference. The Rayleigh region is the area when the relative frequency is < 1. The Mie region, characterized by alternating constructing and destructive interference is found in the range $[1,\ 10]$. The remaining region is the optical region. The image is source from [Wikipedia - Radar cross section of metal sphere from Mie theory.svg].

Size In general, the larger an object is the larger is its radar cross section. For a sphere, the radar cross section increases linearly with the circumference, when the circumference is approximately 10 times as long as the wavelength. This area is called the optical region. For smaller spheres things start to get interesting, as seen in fig. 2.1. When circumference is between $[\lambda,\ 10\lambda]$, the cross section oscillates as a function of the circumference. This is caused by alternating constructing and destructive interference between the part of the radar wave that directly reflects off the sphere and the creeping radar wave that travels around the circumference of the sphere. The phenomena is called Mie Scattering, which also is responsible for the white glow around the sun. If the sphere is even smaller it will start resonating with the wave and return the energy in all directions, predominantly in and opposite to the direction of the radar wave. What we observe here is known as Rayleigh scattering, which is famously known for creating a blue sky. [Skolnik 2001, p. 51] The same principles are valid for more complex shapes than a sphere, but the interplay between constructive and destructive interference relate to how the wave propagates in the material

Shape The shape of an object contributes a lot to how much is reflected back in the direction of the radar (backscattered). As shown in fig. 2.2, a corner reflector or a perpendicular flat plate will have a very large cross section due to almost all the incoming radiation being backscattered. A $1\,\mathrm{m}^2$ plate might have a radar cross section of $> 100m^2$. [Skolnik 2001, p. 53]. For most objects, the cross section will vary a lot depending on the incident angle of the radar wave, which can clearly be seen in fig. 2.3. For some of the spikes it might be a flat area that suddenly became perpendicular to the incident wave, in other cases it might be a corner

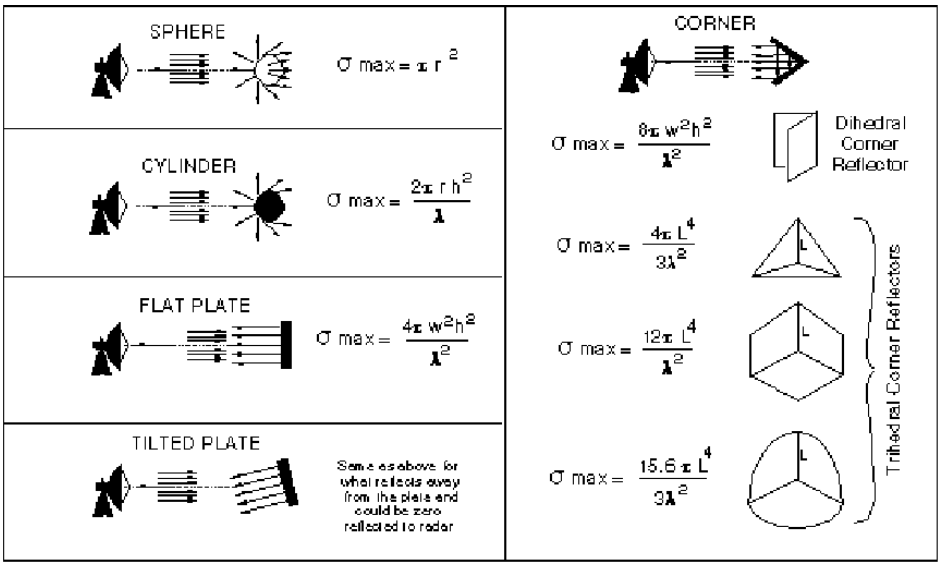


Figure 2.2: How some basic shapes reflect electromagnetic waves. We see that a corner is a very good reflector, the same goes for a flat plate perpendicular to the radar waves. The image is sourced from [Electronic Warfare and Radar Systems 2013, p.4-11.2]

reflector.

Material A general rule of thumb is that the more electrically conductive a material is, the more likely it is to be a good reflector. This is why targets made of metal have a larger cross section than a similar object made of a non-conductive material such as plastic.

Clever utilization of shape and material can allow a designer to choose the cross section of a target. In the case of a fiberglass sailboat, which is almost invisible to radars, it is common to fix radar reflectors to the mast to increase visibility. On the other hand, fighter planes such as the *F-117 Nighthawk* utilize dissipative paint and large flat plates with few joints in order to reduce the radar cross section to $0.025\,\mathrm{m}^2$. [Richardson 2003].

Antenna

The purpose of an antenna is to affect the radar waves so that they predominantly radiate in one direction. The reference antenna is the isotropic antenna [Stutzman and Thiele 2013, pp. 36–37], which spreads the energy uniformly in all directions, as well as receives equally well from all directions. The ideal antenna has a unity gain in all directions, as shown in fig. 2.4a. Such an antenna is impossible to create, so fortunately it is rather useless for most radar purposes. For a radar we want to be able to direct the beam towards a specific direction as to light up a target, as well as receive from that direction. There are several ways to increase the directivity of an antenna, one of the simplest and most known is to attach a parabolic reflector to an isotropic antenna. An example radiation pattern is shown in fig. 2.4b. Relative to the isotropic antenna we see that radar waves from the back are attenuated while strengthened in the front.

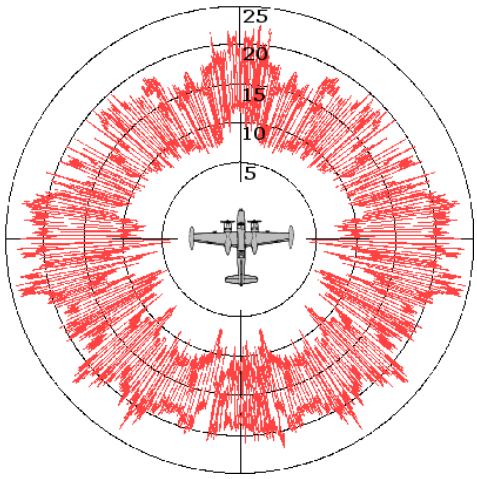


Figure 2.3: The radar cross section of a Douglas A-26 Invader, showing off how the radar cross section varies as a function of angle. It can also be seen that the radar cross section of a complex target is wildly varying. The image is fetched from [Wikipedia - Sigma_invader_RCS.png].

Different radar systems utilize different antennas. An automotive radar system might require a wide aperture compared to e.g. an airport or a weather radar. As such there exists a wide variety of antenna styles.

Noise

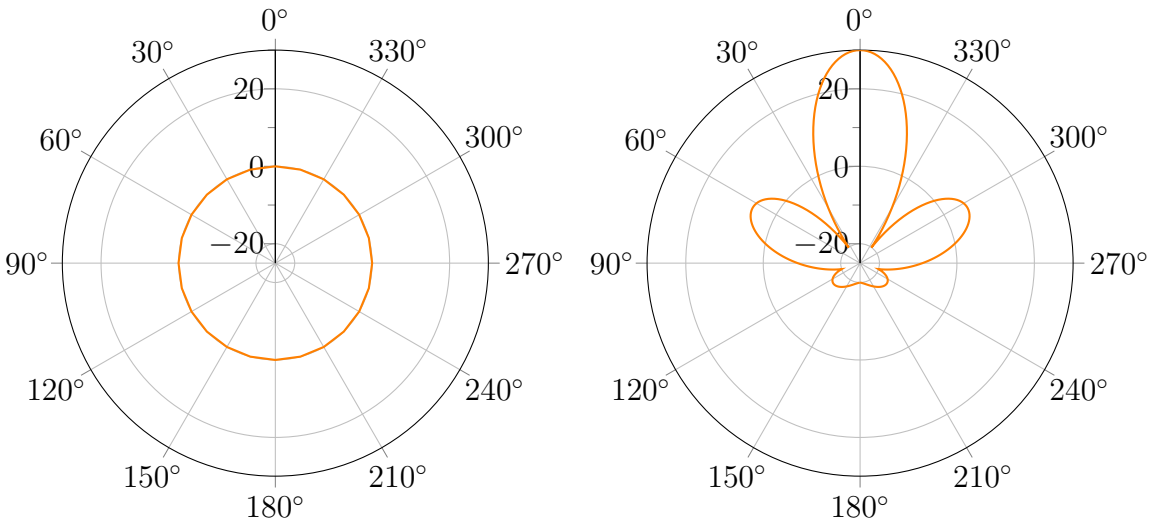
As with any system, the radar is affected by noise. The predominant sources of radar noise is thermal noise induced in the receiver as well as objects that produce wideband noise, such as the Sun, the Earth or cosmic background noise. The last form of noise is caused by the environment called clutter. Sources of clutter can be unwanted reflection from the ground, a flock of bird or chaff[1]. The ability of a radar system to differentiate a signal from background noise is given as the minimum SNR of the system.

Radar systems operating on the same frequency can also be a cause of noise. For pulsed systems, this is usually avoided by using techniques such as staggered PRF, which means that the pulse repetition rate changes slightly between each pulse. By integrating the results over several pulses, other radar systems will just appear as an increased noise floor.

2.2 Ultra-wideband Pulse-Doppler Radar

The radar hardware that is used in this thesis is the *XeThru X2M200* from *Novelda*. It utilizes the *Novelda X2*, a Ultra-Wide Band (UWB) Pulse-Doppler Radar System on a Chip (SoC) (more on the system in section 3.1.2). This makes it worthwhile to discuss the features and differences of a UWB Pulse-Doppler radar system. The information in this section is based on [Skolnik 2008, Chapter 4] and [Kinzie 2011, Section 2.2].

[1]Small strips of metal that is spread by e.g. fighter aircraft as a radar countermeasure.



(a) Isotropic antenna

(b) Parabolic antenna

Figure 2.4: (a) show the radiation pattern of an isotropic (omnidirectional) antenna in 2 dimensions.
(b) show an approximation to the radiation pattern of a parabolic antenna. Both plots are inspired by [Stutzman and Thiele 2013].

The most prominent feature of a Pulse-Doppler radar is that it is possible to differentiate moving targets from background clutter by using the Doppler effect. When a target has a radial velocity relative to the radar system, the carrier frequency of the reflected wave is shifted slightly and gives rise to the phenomena known as the Doppler frequency shift. Originally Doppler radars were CW systems, and measured the Doppler frequency shift of the reflected waves. The disadvantage of this approach was the inability to detect the range to a target. Later systems solved this problem by either modulating the radar wave, such as Frequency Modulated CW (FMCW) systems, or by pulsing the radar wave on and off, which gives us a Pulse-Doppler radar system.

2.2.1 Velocity Measurement

A target moving radially relative to the target imparts a small frequency shift on the reflected radar wave. With f_d as the Doppler shift, v_r as the radial velocity, f_t as the carried frequency and c as the speed of light we have the following formula:

$$f_d = \frac{2 f_t v_r}{c} \tag{2.2a}$$

$$= \frac{2 v_r}{\lambda} \tag{2.2b}$$

Now, what is misleading about the name Pulse-Doppler radar is that the Doppler frequency shift is not measured directly. As the target is moving radially, the number of

wavelengths the signal has to travel ($^{2R}/_{\lambda}$) changes, and causes the phase of subsequent reflections to change ever to slightly. The total phase change is given by the formula:

$$\phi = 2\pi \frac{2R}{\lambda} \tag{2.3}$$

Here ϕ is the phase change, R the distance the target and λ is the wavelength of the carrier signal. Differentiating eq. (2.3) with respect to time and inserting eq. (2.2) gives us:

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{4\pi}{\lambda}\frac{\mathrm{d}R}{\mathrm{d}t} \tag{2.4a}$$

$$= \frac{4\pi v_r}{\lambda} \tag{2.4b}$$

$$= 2\pi f_d \tag{2.4c}$$

This establishes a relation between phase change and Doppler frequency shift, showing how measuring the phase between subsequent pulses allow us to calculate the radial velocity. Another takeaway from eq. (2.3) is that it potentially can be used to get a better estimate of the range R.

2.2.2 Pulse Repetition Frequency and Range-gating

A Pulse-Doppler radar generates a radar wave pulse train with a specified interval between each pulse. The size of this interval is given with the PRF. The PRF decides what we call the Maximum Unambiguous Range (MUR) or the radar. If the PRF is 30 MHz the MUR is $^{c}/_{2\,\cdot\,15\,\mathrm{MHz}} \approx 10\,\mathrm{m}$, which is how far an object can be from the radar for the reflection to arrive back at the radar before another pulse is sent. In this situation, if there is a strong reflector 11 m away from the radar, the radar will not be able to distinguish this from a reflection returned from a target 1 m away.

Staggered PRF is a technique where the PRF is slightly changed in a pseudo-random pattern, which causes the ambiguous target to change possition between each pulse. When averaged over several pulses the ambiguous target will become a part of the noise floor. Another benefit staggered PRF was briefly touched upon earlier, which is that another pulsed radar system operating at the same carried frequency will not cause erroneous target detections as it will just disappear into the noise floor.

A technique utilized together with staggered PRF is range gating, in which the receiver is only turned on for a brief amount of time. This segments the space in which targets can exist, such that e.g. reflections only from the segment placed between 1 m to 2 m away from the radar will be registered by the radar.

2.2.3 Ultra Wide Band

For a radar system to be UWB it has to occupy a fractional bandwidth ≥ 0.2 according to [Aerospace and Electronic Systems IEEE Society 2007]. The fractional bandwidth is defined as follows

$$b_f = 2\frac{f_h - f_l}{f_h + f_l} \tag{2.5}$$

With f_h and f_l begin the upper and lower frequencies at which the signal is 10 dB lower than at the frequency with the most energy.

UWB Pulse-Doppler systems usually become UWB because the pulses are very short in the time domain, usually on the order of a few nanoseconds, which corresponds to a wide frequency-domain spectrum. The primary benefit of such radar systems is very fine range resolution, increased immunity to passive interference from rain and chaff, and increased target detection probability [Fontana 2004]. UWB is also necessary for doing Synthetic Aperture Radar (SAR) processing, as it will increase the resolution in range.

2.3 Radar vs. Other Technologies

While radar itself is an interesting technology, it is worth taking a qualitative evaluation of other technologies that can perform well at obstacle detection. In this section I will discuss some of the pros and cons of radar vs RGB-D cameras, lidars and sonars.

2.3.1 Radar

The main benefit of radar systems is that they do not depend on optical visibility of the target, and a radar system can be used without good lighting conditions. Radars can also have an impressive range, up to 3 km like the Echodyne MESA-DAA [Echodyne 2016] (at the cost of much lower range resolution at 3.5 m). Other radar systems have much lower maximum range, which also comes with much finer resolution, such as the *Novelda X2* SoC used in *XeThru X2M200*.

Radar systems are only able to measure the distance to objects and not their direction. It can however measure through an object, which no of the other mentioned technologies are able to. Unless the radar in some way is steerable, either mechanically, or as a phased array antenna, there is no way to gain information about where the obstacle lies. Another possibility is to use several radar systems or antennas and collect data from them, similar to what has been done in the Aerotenna µSharp [Aerotenna 2016]. Most available radars also usually have a quite wide beam, because they operate at a relatively low frequency. Operating at higher frequencies to obtain a smaller beam width would require the radar to output more power in order to reach the same distance, which could cause problems with electromagnetic spectrum regulations.

Radar systems are not very sensitive to any noise that is likely to appear in the working environment of a multirotor. A radar is however sensitive to clutter, and if there are several obstacles in close proximity it might be difficult to differentiate them. Some targets are also not very visible to radars, such as some electrical cabling or Polyvinyl Chloride (PVC) pipes, although this depends heavily on the center frequency of the radar. Small embedded radar systems are also rather cheap, such as the *X2M200* retailing at 149.00$.

2.3.2 Lidar

Lidar is essentially the same as a radar, except for using light instead of radio waves, which we can be clued in upon by looking at the expanded name LIght Detection and Ranging. Lidar systems use short laser pulses and measure the time until a a reflection returns. One big benefit of lidars compared to radars is the possibility of having a very narrow beam width, which gives the lidar a narrow field of view. By using a series of mirrors it is possible to steer the lidar beam as to measure a series of points, along either one or two dimensions, and create a depth map that can be used for collision avoidance.

Lidars have mostly been used within atmospheric research, but with the advent of autonomous vehicles they have become increasingly more common in obstacle detection. Historically lidars have been very expensive, as they require complex moving mirrors in order to cover a large area. However Micro-Electro-Mechanical System (MEMS) lidars are under development and is expected to be released sometime this year (2017), which is likely to reduce the costs quite a lot [Velodyne LiDAR 2017].

Lidar systems, such as the Velodyne Puck [*Velodyne Puck VLP-16*], provides a 30° degree Field of View (FOV) while still covering a 360° perimeter. It has a range of up to 100 meters, with an accuracy of ±3 cm in radial direction. This is an upper end automotive lidar and is priced as such. Cheaper options, such as *Slamtec RPLIDAR*, can only scan in the plane, in a distance up to 6 m away from the lidar. It has an accuracy of less than 1% of the distance.

Lidars are quite sensitive with regards to lighting conditions and will often perform poorly in broad daylight. The lidar target also needs to be reflective, as well as not occluded, in order to be picked up by the lidar. [*Lidar - Range-Resolved Optical Remote Sensing of the Atmosphere*]

2.3.3 Ultrasonic Sensors

Ultrasonic sensors operate on a similar principle as radar and lidar, except for using ultrasound[2]instead of electromagnetic waves. They can be provided with a narrow and constant beamwidth, which is needed in order to find the direction of an obstacle, as well as the distance with millimeter precision.

[2]Sounds with a frequency higher than the human ear is able to pick up

Unfortunately most ultrasonic sensors are one dimensional in that it is not possible to discern the position of an obstacle. While a solution akin to a phased array antenna would be possible, there does not seem to be any available. In order to know the approximate direction of a target it would be necessary to either mechanically rotate the sensor or to use an array of sensors pointing in different directions as was done in [Gageik, Benz, and Montenegro 2015].

An ultrasonic sensor has an advantage in the fact that it does not require special lighting conditions. It is however sensitive to what materials it can discover, as the material preferably has to be stiff as to reflect the sound waves. Ultrasonic sensors can also be acquired at a very affordable price.

2.3.4 RGB-D Cameras

Red Green Blue + Depth (RGB-D) cameras is a collective term for camera systems that use various techniques, both active and passive, in order to add depth information to each pixel in the picture. The added benefit of this approach is that image processing algorithms can be used together with the depth information for more accurate extraction of obstacles.

Passive

One example of a passive technology is *Zed* by *StereoLabs*. The *Zed* uses two cameras in order to calculate the depth of a pixel. It is capable of measuring the distance to a pixel in a range of 0.3 m to 20 m.

The benefit of a passive technology such as this is that it is quite resilient to strong external light sources, as well as working well over long distances. However it will not perform good on uniform surfaces, much in the same way as human eyes. [Deris et al. 2017]

Active

The active technologies usually consist of one camera and one projector. The projector projects a pattern, either in the visible or the in Infrared (IR) spectrum, that the camera picks up. By analyzing the distortions in the pattern it is possible to extract information on the distance between the object and the camera. Examples of cameras that utilize such a technology is the *Microsoft Kinect*, the *Intel RealSense*, and the Norwegian company *Zivid*. Common to all systems is a rather small working range, usually between 0.5 m to 2 m. This approach is however much more accurate than the passive approach, with a depth resolution of up to 0.1 mm for the *Zivid*.

The active systems can still perform well on uniform surfaces, but will not work as well in strong lighting as the projected pattern will be swamped out.

2.3.5 Comparison

The different technologies presented all differ greatly in their strengths and weaknesses. A camera and a lidar might not work in broad daylight as well as a radar and a sonar, while it is easier to find the direction of an obstacle with the former compared to the latter. In order to develop a more robust system it would be a good idea to consider using more than one kind of sensor, preferably two that complement each other well, such as a camera and a radar as was done for an automotive vehicle detection system in [Wang et al. 2016]. This kind of sensor fusion is outside the scope of this thesis, but it is definitely a topic worthy of further exploration. For example could a cheap laser distance sensor be mounted with the radar to work as a poor mans lidar and increase the directivity of the measurement.

3 | Embedded Radar System

In order to test a radar anti-collision system both a robust radar platform, as well as suitable support software, is necessary. This chapter discusses how the embedded radar system was built, what choices was made and why they were made, and describes each part of the system. In addition the software controlling the multirotor, software for communication with the multirotor and debugging software is described. A test platform is designed to verify the capabilities of the system. A user manual of the system is available in appendix A.

Contents

3.1 Hardware

The hardware that was developed for the radar system is to provide a good base for developing algorithms for object avoidance, and builds upon a prototype developed in [Koren 2016]. This section details its development and conclusion, as well as the different parts of the hardware, such as the multirotor and its Onboard Computer (OBC), the characteristics of the radar system and the Printed Circuit Board (PCB) on which the radar is mounted.

Before starting development a small set of specifications was created for the system.

1. Must be able to cover a 360° perimeter.
2. Must be able to differentiate objects on the 360° perimeter.
3. The system must be able to measure the distance to a target with good accuracy.

3.1.1 Multirotor

The radar system is supposed to be flown on a hexarotor as shown in fig. 3.1b. The hexarotor platform is a custom setup, using a Beaglebone Black as its OBC. The OBC runs a custom Linux distribution, GNU/Linux Uniform Environment Distribution (GLUED), developed by Laboratório de Sistemas e Tecnologia Subaquática (LSTS), as a minimalist distribution for embedded systems. GLUED is paired with DUNE: Unified Navigation Environment (DUNE), which is a runtime environment for the on-board software of unmanned systems. DUNE provides interfaces for green threads, message passing, math libraries, and a hardware abstraction layer, and is easily extendable in C++.

On the hardware side, the multicopter can provide two voltages, regulated 5 V and unregulated battery voltage. Connections made to the OBC are done with *Molex Micro-Fit* connectors, which can be seen in fig. 3.1a.

3.1.2 Radar

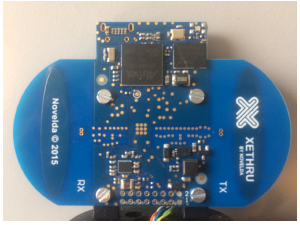
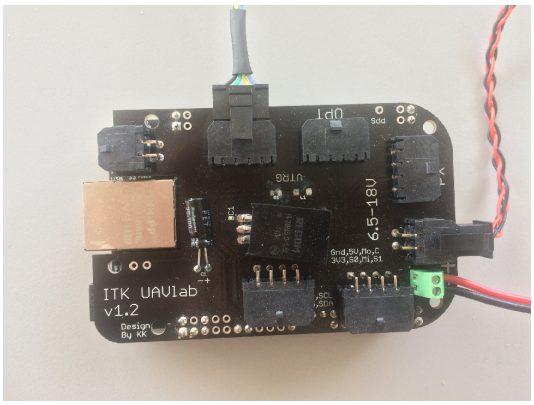


Figure 3.2: The *Novelda X2M200*

The *XeThru X2M200* [see Novelda AS 2016a] is a sleep and respiration monitor kit developed by *Novelda*. It is powered by the *X2* SoC also developed by *Novelda*. The kit utilizes a Pulse-Doppler radar to sense the minute movements of the chest when breathing. In addition to its use as a medical sensor, it is capable of outputting preprocessed radar measurements. This makes the *X2M200* viable for use in collision detection and avoidance.

Novelda recently launched the *X4M03*, explicitly for development purposes, with a much more powerful radar SoC. While it was released too late to be of use in this thesis,



(a) The OBC of the multicopter, a Beaglebone Black. On top of the Beaglebone is a cape with a 5 V voltage regulator and connectors for other systems

(b) The hexarotor that the embedded radar system was designed for.

Figure 3.1

the increased range and measurements frequency makes it better suited for further explorations into this topic.

Further on in this section we will go into detail on the characteristics of the system as found in [Novelda AS 2015] and [Novelda AS 2016b].

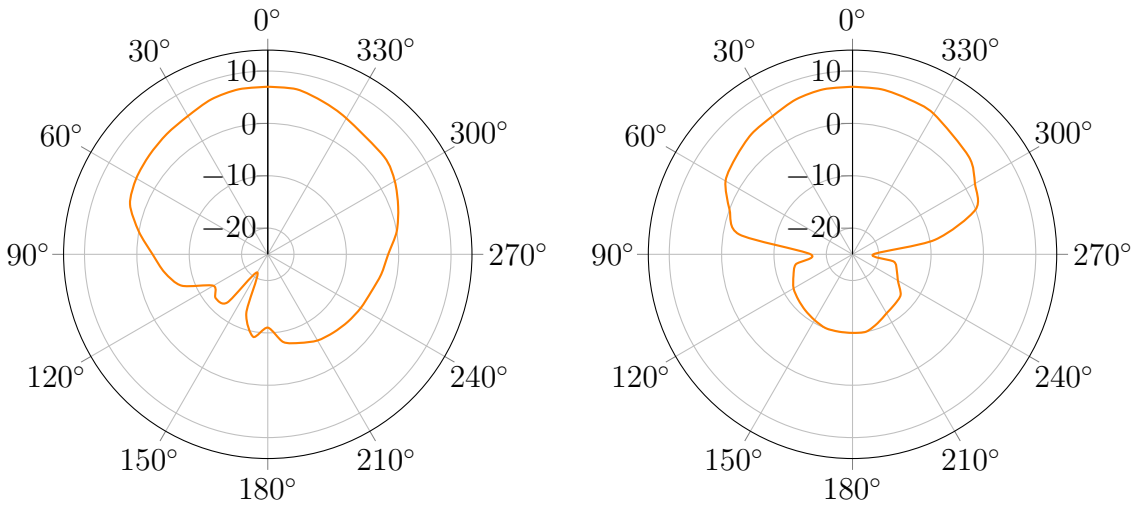
Update rate

The update rate of the *X2M200* averages at 20 Hz, but experimental verificattion of this shows some jitter in the update rate, at about ±1 Hz. The pulse repetition frequency of the radar itself is approx. 30 MHz, and the returned pulse is averaged over several samples in order to reduce noise and get more reliable measurements. All in all, these effects are what leaves us with an update rate of 20 Hz.

Distance

The system is able to return a radar frame covering a distance interval of 2 m. The interval must be positioned inside the interval [0.3 m, 4.1 m], and can be placed with a resolution of 1 cm. The measurement interval is then split into bins of approximately 4 cm[1], each associated with an amplitude and phase measurement. The *X2* utilizes range gating, with a range gate of 1 m. In order for the *X2M200* to return data from an interval of 2 m the active range gate is switched between successive measurements performed by the *X2*.

[1]The bin size of approximately 4 cm comes from the wavelength of the center frequency.



(a) Antenna radiation pattern in the horizontal plane.

(b) Antenna radiation pattern in the vertical plane.

Figure 3.3: The antenna radiation pattern for the transmit antenna of the *X2M200*. Zero degrees is orthogonal to the *X2M200* hardware. The data is fetched from [Novelda AS 2016a, pp. 18–19]

Frequency and Bandwidth

The *X2M200* operates at a center frequency of 7.3 GHz with a 10 dB bandwidth of 2.3 GHz. The output pulse lasts for approximately 6 periods of the output frequency, that is 8.2 ns. The mean PRF is set automagically by the *X2* depending on the distance, but the MUR is increased by staggered PRF. Another benefit of PRF in this context is that is allows us to use several *X2M200* in close proximity to each other, as other units are likely to appear mostly as an increased noise-floor and not cause false positives.

Antenna

As the *X2M200* is meant to be stationary and sense humans over a large area, the receive and transmit antennas have rather wide beams. This can be seen in fig. 3.3, which has a 10 dB beamwidth of 150° in the horizontal plane. For a system that relies on accurately pinpointing the position of an object this can pose a problem, as an object will be visible from many different orientations.

Communication

Communication with the *X2M200* is over Universal Asynchronous Receiver/Transmitter (UART). The message format, as well as the frame format, is explained in further detail in [Novelda AS 2016b]. The frame structure is shown in table 3.1. The first byte is a Start of Frame (SoF) byte, followed by the data, a checksum and an End of Frame (EoF) byte. The first byte of the data-field specifies the message-type, sometimes followed by a larger 4-byte word further specifying the message. Supported values are either byte,

0	1 N-3	N-2	N-1
<Start>	Data	<CRC>	<End>

Table 3.1: The structure of a frame received from the *X2M200*. The first row is the byte number of the bytes in the second row.

byte arrays or 32-bit integers and floating point values in little-endian byte order. An additional special byte, the escape byte, is also used by the protocol. The escape byte is inserted before any special byte, so that the system knows that the special byte is to be treated like a normal byte.

Checksum The checksum is a parity byte calculated by taking the Exclusive OR (XOR) of all successive bytes, starting with the start byte and ending with the last byte in the data-field (excluding escape bytes). This is a terribly weak approach to verifying data integrity, as it is only able to discover if a bit within each byte has been flipped an odd number of times. On several occasions data received from the *X2M200* has been corrupted, with the only way of verifying data integrity is to analyze the data in relation to previous data. This is especially a problem when using UART, as acknowledged by *Novelda* at [XeThru Forum 2017]. A better alternative would have a proper Cyclic Redundancy Check (CRC) checksum, either 8, 16 or 32 bit long, with a standard polynomial.

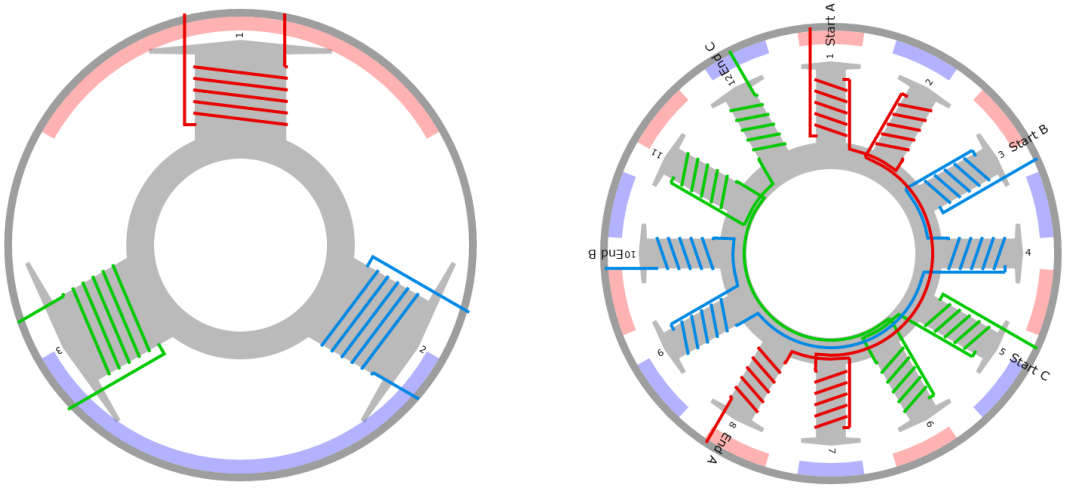
3.1.3 Motor

In order to cover a 360° a few options were initially considered. The most robust option, to use several radars pointing in different directions around the multirotor, was deemed too expensive for a prototype. Another strategy, inspired by the stereotypical view of radar systems, was to rotate the radar. The benefit of this approach is increased resolution in the plane of rotation, at the cost of increased mechanical complexity.

In [Koren 2016] it was concluded that using a Brushless Direct Current (BLDC) motor was the best choice for this application because of their slim profile, 360° operation, as well as support of a slip ring. A BLDC motor consists of a permanent magnet rotor and a stator with three separate coils. The three coils are connected in a manner as shown in fig. 3.4 such that applying a current through them creates a magnetic field. Controlling this magnetic field can then be used to control the position of the rotor.

In late 2016 several BLDC motor manufacturers started delivering their motors with an optional magnetic encoder, as well as support for slip rings. The magnetic encoder uses an array of Hall effect [2]sensors, that can sense the rotation of a toroidal magnet attached to the rotor of the motor. Two BLDC motors with magnetic encoders were tested, their specifications listed in table 3.2. Two motors were ordered so that we could test both motors to see which was suitable for our application, and in the end the *GBM2804H-100T* won because of size and weight.

[2]The Hall effect appears when a magnetic field is orthogonal to a current through a conductor, which produce a voltage difference orthogonal to both the magnetic field and the current.



(a) (b)

Figure 3.4: (a): A simple BLDC motor with 3 coils spaced 120 degrees apart, as well as 2 poles.
(b): A gimbal BLDC motor with 12 coils and 14 poles. This is the same configuration as the BLDC motor used for this project.
Both images are courtesy of [Homebuilt Electric Motors - Felix Niessen].

	GBM2804H-100T	GBM4108H-120T
Weight	41.5 g	120 g
Height	16.8 mm	25 mm
Diameter	36.8 mm	46 mm
Slots	12	12
Poles	14	14
Turns per Pole	100	120
Resistance	6.5 Ω	11.2 Ω
Voltage	7.4 V	11.1 V

Table 3.2: Comparison of the two BLDC motors that were bought.

Magnetic encoder problems Both of the motors in table 3.2 were delivered with an *AS5048A* magnetic encoder from AMS. The intent was to use the encoder to get a better measurement of the orientation of the motor. However after some testing it became obvious that the measurement drifts opposite to the direction of rotation, and the rate at which the measurement drifts seems to be (nonlinearly) related to the angular velocity of the motor. This makes it clear that the magnetic field produced in motor interferes with the operation of the *AS5048A*.

An alternative method for estimating the angle was devised, using a photoreflector that was supposed to calibrate the magnetic encoder. If the speed of the BLDC motor is constant, the time between successive rotations can be measured and the speed can be estimated quite precisely. This data can then be used to interpolate the pointing angle of the radar.

3.1.4 Microcontroller

The system utilizes a Microcontroller Unit (MCU) from the *STM32*-series of *ARM Cortex-M* based microcontrollers from *STMicroelectronics*. Ease of portability to another MCU in the same series, as well as computational power were two of the main factors behind this decision. *STMicroelectronics* provides a portable peripheral library, *STM32 HAL*, that fascilitates switching between different MCUs. *STMicroelectronics* also has a range of MCUs that makes it easy to choose a more powerful microcontroller should that prove necessary in the future.

During the development, the *Mini-M4* development kit from *MikroElektronika* was used. As it is delivered in a DIP40-package, it is easy to plug into a breadboard for prototyping, as well as attaching to a PCB. The *Mini-M4* sports a *STM32F415RG*, which can operate at 168 MHz, has a single precision floating point unit, as well as support for the *ARM* digital signal processing instruction set [*MINI-M4 Datasheet*]. This is useful should the need arise for processing the radar data on the MCU instead of retransmitting it.

3.1.5 Circuit board

A circuit board was necessary to combine all parts of the system. The circuit board works as a mount for the motor, as well as connecting all other parts of the system. The developed circuit board can be roughly divided into four parts; power supply, MCU, motor control and angle sensor. The project files and full schematics available as digital appendices.

Power Supply

There are two voltages available on the multirotor, unregulated battery voltage (12 V) as well as regulated 5 V. However the microcontroller as well as the *X2M200* need 3.3 V, and the motor could benefit from an adjustable voltage to control power consumption.

Therefore two power rails are provided on the PCB, one at 3.3 V and another adjustable one.

The two power rails are regulated with two Low DropOut (LDO) regulators . The adjustable voltage regulator was in turn connected to a potentiometer that could be used to set the motor voltage.

Microcontroller

As the *Mini-M4* comes in a DIP-40 package, the most convenient solution for attaching it to the circuit board is to use a DIP-40 socket. The *Mini-M4* also comes with a 3.3 V power supply that can power both the MCU and the *X2M200* from USB during prototyping.

LEDs The *Mini-M4* comes with two integrated Light Emitting Diodes (LEDs), but three additional LEDs were placed on the PCB in order to allow for quick visual inspection of system status. The significance of each LED can be found in appendix A.

Programming The *Mini-M4* can be programmed and debugged with an ST-LINK debugger using the Serial Wire Debug (SWD) protocol. When debugging it is important to keep in mind that the debugger and the MCU need to share common ground.

Motor Control

The gist of driving a BLDC motor it that the motor consist of a rotor with a permanent magnet and a stator with three separate coils. The three coils are connected in a manner as shown in fig. 3.4, and sending a different current through each coil produces a magnetic field that the rotor will align to. By driving the three coils with sinusoidal waveforms 120° degrees out of phase with each other, the motor rotates. The period of the sinusoid decides the speed. The current, and by extension the voltage, necessary to produce a magnetic field of sufficient strength depend on the characteristics of the motor, such as the strength of the permanent magnets and the resistance of the coils. This is why the motor is in need of a higher voltage than the readily available 3.3 V-rail as well as why the MCU-circuitry needs to be protected from the motor control-circuitry.

There are two feasible solutions to create a motor driver, create it from scratch with Metal-Oxide-Semiconductor Field-Effect-Transistor (MOSFET) drivers or use a commercially available driver such as the *L6234* Three-Phase Motor Driver from ST Microelectronics. Again, in an effort to shorten development time, the *L6234* was chosen. The main drawback of this strategy is that it uses more physical space than a MOSFET-solution, as the *L6234* need support circuitry for internal voltage regulators as specified in [STMicroelectronics 1998]. The *L6234* is also slightly overkill for this purpose as it can be used for voltages up to 52 V with an output current of 5 A. The *L6234* nonetheless was supplied with a small heat sink to dissipate heat.

Two protection measures exist between the MCU and the *L6234*. When the *L6234* is not connected to the high voltage rail, the input pins, EN1-3 and IN1-3, are essentially short circuits to ground. Applying a voltage to these pins in such an occasion can damage the *L6234*. As we have no way of knowing if the MCU will be powered on before the high voltage rail or not, 1 kΩ current limiting resistors are added to the inputs of the *L6234*. In addition there are 15V zener diodes on the MCU inputs, that aid in ElectroStatic Discharge (ESD) protection. In hindsight, these diodes can be skipped as a cost and space saving measure.

Because the BLDC motor is an inductive load, the voltage over the motor coils are a function of the rate of change of the current.

$$v(t) = L\frac{di(t)}{dt} \tag{3.1}$$

Essentially this can cause a very high voltage on the outputs of the *L6234* if the power is cut abruptly, which could cause damage to the *L6234*. To prevent this, a 40 V Schottky diode is added to each output.

Angle Measurement

The magnetic encoders that are delivered with the BLDC motors are connected to the 3.3 V rail on the PCB. The absolute angle of the motor can be read out from the sensor over an Serial Peripheral Bus (SPI) bus.

Additionally the measurements from the encoder were to be calibrated with a photoreflector. The photoreflector has a built in IR LED, which is reflected by a small rod on the radar mount. This causes the photoreflectors built-in phototransistor to open, which pulls the line going to the MCU to logic low. The output of the phototransistor might bounce[3], causing the MCU to register a falling edge more than once per rotation. To prevent this behavior, a capacitor and a resistor is connected in parallel with the phototransistor, effectively dampening the voltage such that there will be no bouncing.

Connectors

The PCB comes with three *Molex Micro-Fit* connectors so that it can easily connect with the Beaglebone Black. The pinout of these connectors can be found in appendix A.

3.1.6 Radar Mounting Bracket

In order to mount the *X2M200* to the motor a mounting bracket was created. The mounting bracket is made so the system has a low profile, with the radar rotating around its horizontal center. Holes for both M2 and M3 screws have been added, so that it can be mounted to both of the motors that were being considered. There is

[3]Bounce originally signifies contact bounce found in mechanical switches, where the springiness of the metals caused the contacts to bounce against each others, rapidly turning the switch on and off until the bounce stabilized. A similar phenomena appear in phototransistors.

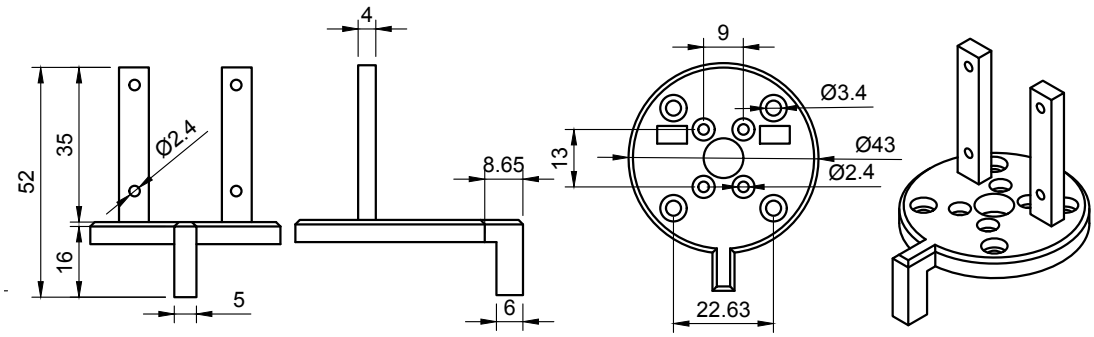


Figure 3.5: The mounting bracket for attaching the radar to the motor. All dimensions are in millimeters.

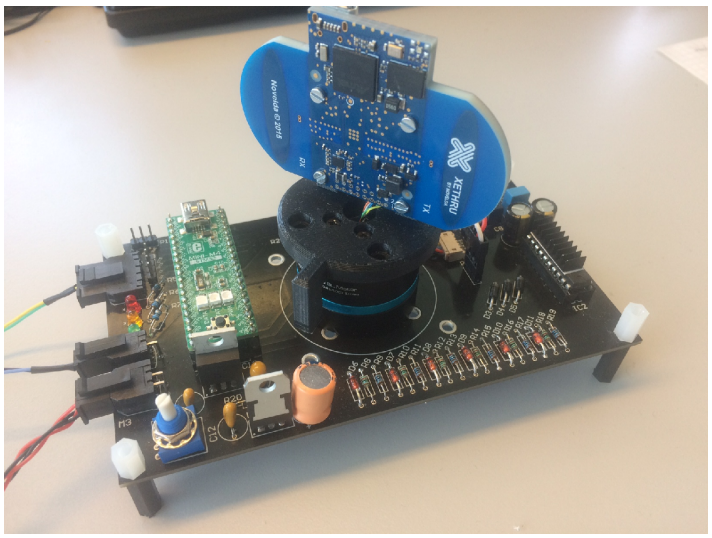


Figure 3.6: The final embedded radar system.

also a pass-through hole for the cables coming from the slip ring and a small rod for activating the photoreflector.

3.1.7 Final Hardware

The final hardware can be seen in fig. 3.6. The motor control circuitry is on the right hand side, while the power supply and the microcontroller is on the left hand side. The photoreflector is hidden by the motor.

3.2 Software

Three different types of software were developed that directly relate to the hardware; embedded software for the *MINI-M4*, communication software, and a visualizing software

for debugging and understanding. This section presents how they were built as well as their purpose.

3.2.1 Embedded Software

The embedded software is responsible for controlling the motor, measuring the pointing angle, as well as communicating with the *X2M200* and the *Beaglebone Black*. The main goal when developing the software, besides fulfilling the specifications, is that it is easy to develop and maintain. A prototype of the software relied heavily on the *STM HAL*. While this makes the code easy to port between different members of the *STM32*-family, it is neither easy to maintain, nor to add new features. Information flow within the software was also poor, as there are no native message passing interface in the C Standard Library nor in *STM HAL*.

The problems with the software developed in [Koren 2016] could best be redeemed with a total rewrite. *ChibiOS*, which is a small Real Time Operating System (RTOS) combined with peripheral libraries. An RTOS allow us to segment the application into smaller tasks, and provides us with an interface for task synchronization and communication. The peripheral libraries that come with *ChibiOS* has very good support for the *STM32*-family, which makes the code easily portable should a different chip be considered in a future revision.

Structure

The embedded software was designed to avoid any circular dependencies, so that it will be possible to replace the radar, motor or communication protocol without affecting the rest of the system. A dependency graph of the program can be seen in fig. 3.7. The different parts of the system can be described as follows:

- **analyze** – Responsible for collecting information from all sources and transmitting to an external destination. Originally it was also intended to analyze the data and decide areas of interest for which the motor would slow down, which is the reason behind the name.
- **rot_measurement** – Gathering measurements from the hall effect sensor at a predefined interval. Also interpolates between measurements. Was later given an option to only depend of the phototransistor.
- **xethru** – Responsible for controlling peripherals and memory when communicating with the *X2M200*.
- **hall** – Read out angle from the hall effect encoder.
- **motor** – Control the motor and associated peripherals. Supports variable speed.
- **xethru_protocol** – Driver developed by *Novelda* for parsing and creating frames for communicating with the *X2M200*.
- **extcom** – Implementation of the external communication protocol. The protocol is explained in further detail later.

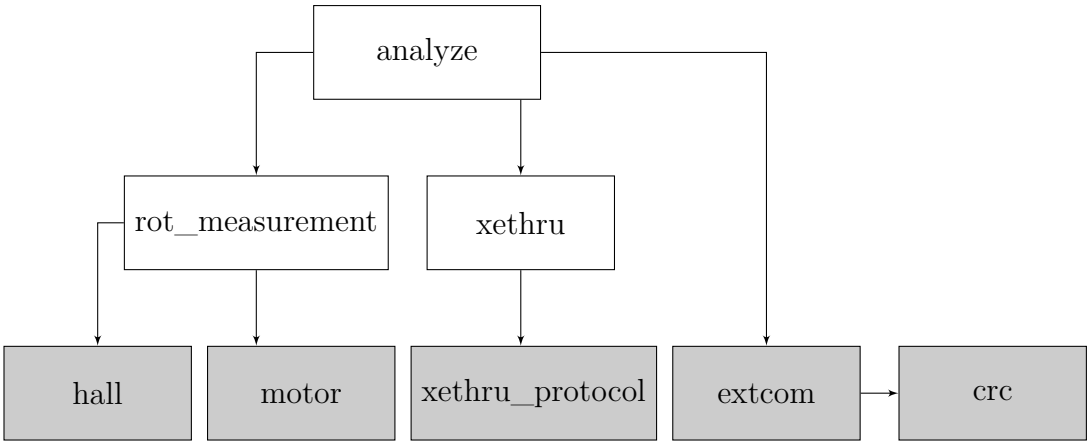


Figure 3.7: The general structure of the software developed for the *Mini-M3*. White blocks are tasks, while gray blocks are drivers.The names correspond to the name of the c source-files.

Name	Length [bytes]	Purpose
Start byte	1	Signifies start of frame. Value is `0x7D`.
Length	4	Remaining bytes of frame
Data	"Length"−5	Arbitrary data
CRC	4	Used to verify data integrity.
End byte	1	Signified end of frame. Value is `0x7F`.

Table 3.3: The general frame structure used for communicating with the MCU.

- **crc** – Driver for utilizing the crc-peripheral on the *STM32F415RG*.

Communication Protocol

The external communication protocol implemented is very simple, and tailored so that it will demand very little resources to implement. The structure of a frame is shown in table 3.3. The main benefits of this frame structure are the "start byte", the "length" field, as well as the "crc". The "start byte" allows a microcontroller to ignore all other data while looking for the "start byte". After receiving the "start byte" the MCU receives the 4-byte long length field, checks that the length is valid (typically less than 1024). Now the MCU can start a Direct Memory Access (DMA) transaction for the remaining bytes, and does not need to use processing power on transferring the frame. After receiving the specified number of bytes the MCU can check the "end byte" to verify that the data is likely to be a full frame, and calculate the CRC to verify data integrity. The specifics of the frame format, how to calculate the crc, as well as the contents of the "data"-field are explained in more detail in appendix A.

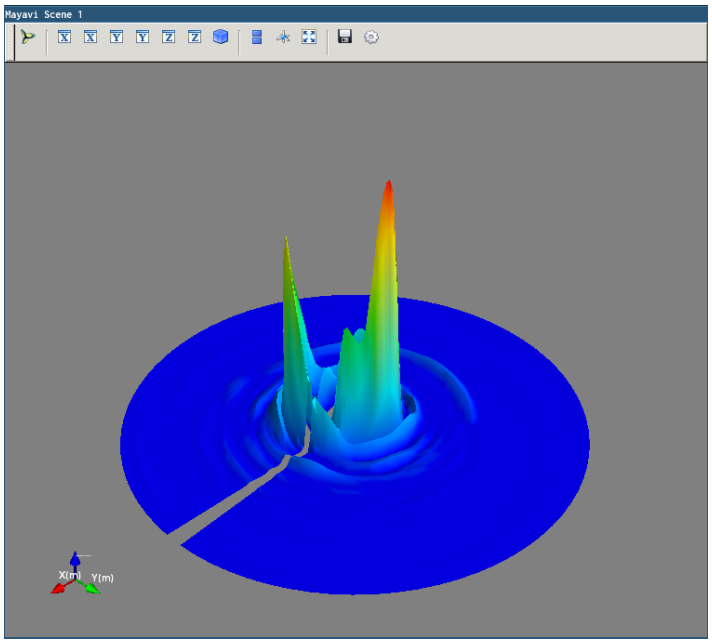


Figure 3.8: A view of the live 3D view software developed for viewing the radar data more intuitively. The peaks in the data is caused by, among others, a computer screen and a chair.

3.2.2 Communication Software

The communication protocol for communicating with the embedded radar system was implemented in two languages. First one python module was created. Because the python module was developed mainly for testing the hardware and prototyping it does not verify the crc of the received frame. Using the python module a logger that saves the radar data to the MATLAB .mat file format was created.

The second implementation is a C++ implementation for DUNE. This implementation verifies the crc, as well as transmitting the data to other DUNE tasks as an Intermodule Communication (IMC) message.

3.2.3 Visualization Software

Using the python communication module, an application for a live 3D view of the radar data was developed. The main use was to help with acquiring an intuitive understanding of the radar data, as well as being useful for debugging purposes.

The application was built using the MayaVi python module. A screenshot of the application running can be seen in fig. 3.8. Usage of the program is described in appendix A.

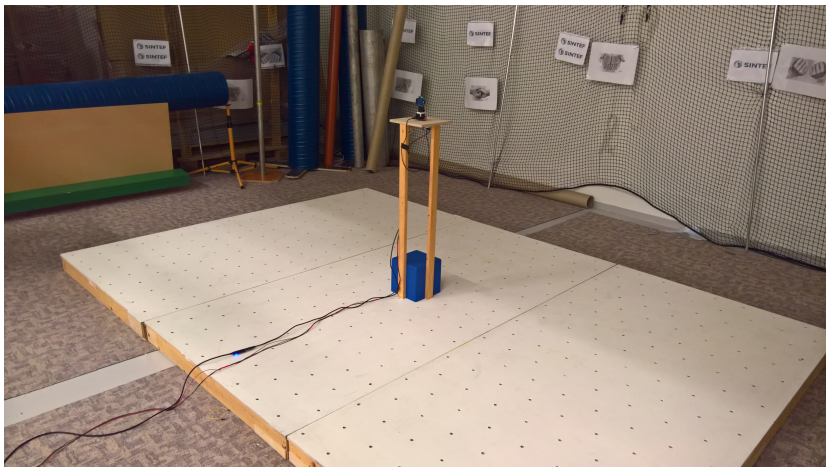


Figure 3.9: The test platform together with the pedestal on which the radar system was placed.

3.3 Test Rig

A test rig was also developed for the embedded radar system, and was used in [Koren 2016] to obtain data from a prototype of the final system. The test rig consists of two parts, a test platform and test targets.

3.3.1 Test Platform

A platform, on which to test the system, was made in cooperation with the workshop at the Department of Engineering Cybernetics at NTNU. The platform consists of a wooden base together with a pedestal on which to place the radar system, as shown in fig. 3.9. The final dimensions of the base were 3.66 m by 2.44 m. It has several evenly spaced holes in it, which work as a coordinate system for placing targets, making it easier to repeat a test. The pedestal positioned the radar at a height of 1.04 m over the surface of the base. This was done in order to reduce possible reflections from the ground, as well as increasing the visibility of the targets in the vertical direction.

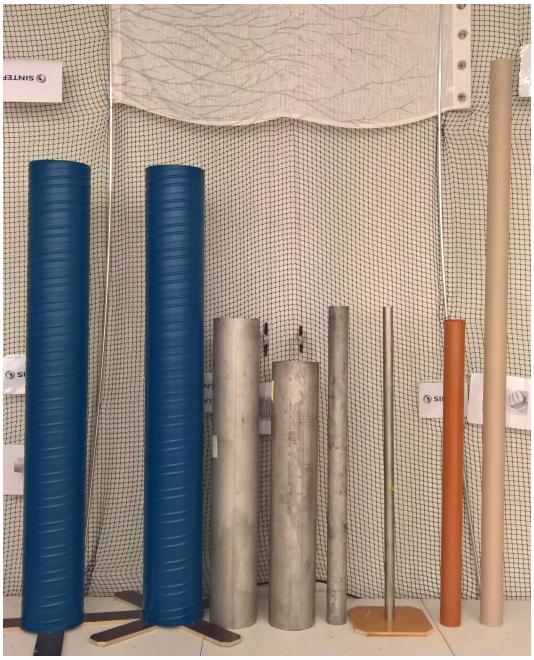
3.3.2 Test Targets

Targets were chosen in order to provide a diverse set of materials and shapes to test the system with.

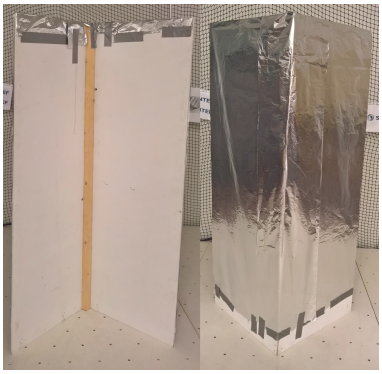
Cylinders Cylinders were chosen because of their simple shape, which has the same radar cross section from all angles [Skolnik 2001]. 6 different cylinders were selected as shown in table 3.4. Unfortunately it was difficult to obtain cylinders of the same height, but in most test situations the top and bottom of the cylinder will be outside the field of view for the radar. The selected cylinders are shown in fig. 3.10a

Name	Diameter[m]	Height[m]	Material
Cylinder Ø31	0.310m	2.400m	Aluminum (ventilation pipe)
Cylinder Ø25a	0.250m	1.625m	Stainless steel
Cylinder Ø25b	0.250m	1.390m	Stainless steel
Cylinder Ø10.5	0.105m	1.700m	Stainless steel
Cylinder Ø5.5	0.055m	1.700m	Stainless steel
PVC Pipe Ø11	0.110m	1.620m	Polyvinyl Chloride
Cardboard Ø11.5	0.115m	3.020m	Cardboard

Table 3.4: Table with the names and dimensions of the cylinders used in the test. A picture of the cylinders can be found in fig. 3.10a



(a)



(b)

(c)

Figure 3.10: (a) show the cylinder used for testing. From left to right in same order as table 3.4.
(b) show the aluminum-foil covered corner.
(c) show the concrete wall used for testing. The net and the vertical aluminum rods were assumed invisible to the radar when compared to the wall.

Corner The corner was made in order to both test the inside corner as a corner reflector, which has a big radar cross section [Skolnik 2001], as well as to test how the corner of a building would appear to the radar system. The corner was rotated with 45 degree increments. The corner was made in wood, which is almost transparent to a radar operating at 7.3 GHz , and then covered in aluminum foil as shown in fig. 3.10c. Unfortunately the aluminum foil layer was a bit crinkled, which decreases the radar cross section slightly. The corner was $1.70m$ high and measured $0.61m$ wide from the corner in both direction. The corner was considered to be 0 degrees relative to the radar when it worked as a corner reflector as shown on the left hand side of fig. 3.10c.

Wall A wall was used in order to check the visibility of concrete structures, as well as the visibility of a large flat area. Concrete itself is not very visible at 7.3 GHz [Yunqiang and Fathy 2005], but the rebar lattice inside the concrete wall is visible. As the wall is not movable, the pedestal was moved around in order to get closer to the wall.

Fun Two mundane objects were tested simply for the fun of it. One of the test objects was a human, myself, the other was a stack of 6 chairs. This can be helpful for testing out object detection, to see how it reacts to more complex targets.

3.3.3 Test Results

A test plan is available in appendix B. The results of testing all targets are available as a digital appendix as MATLAB .mat-files. The purpose of these tests it to gather an extensive set of data that can later be used for object recognition. The data set was extensively analyzed in [Koren 2016], and this report is also included as a digital appendix. [Koren 2016] concluded that the radar is able to detect the distance to an object very well, but because an obstacle is visible for large parts of a rotations it is difficult to pinpoint the direction of an obstacle from visual inspection. Additionally two objects close to one another might merge into one big reflection. At last the embedded radar system has a large noise floor close to the radar.

3.4 Improvements

During testing a few suggestions for future improvements came up. There was no time to do a revision of the hardware with these improvements, but they are written for future reference.

MOSFETs The first suggestion is to replace the *L6234* with MOSFETs. The main problem is that the *L6234* is meant to drive BLDC motors at high speed, while a MOSFET solution can possibly be more efficient as well reducing the slight stuttering of the motor. Additionally it will remove the need for the support circuitry for the *L6234*. I recommend the *TC4452* MOSFET from *Microchip*.

Surface Mount Components Now that the hardware has been tested it would be a good idea to try to minimize the footprint. One way to realize this would be to use surface mount components instead of through hole components. It would cause the soldering process to take some more time though.

Higher Amperage Regulators The voltage regulators that were chosen for the 3.3 V and the adjustable voltage rail tend to become close to overheating during operation. Both of them can deal with 1 A, which is close to the current drawn by the system. Replacing these with higher amperage regulators would reduce this problem, and increase the lifetime of the regulators.

Integrated Microcontroller It would be a good idea to integrate the microcontroller with the PCB instead of on a DIP-40 PCB. This would further reduce the footprint of the embedded radar system.

3.5 Conclusions

The embedded radar system is a prototype of a collision avoidance radar for a multirotor. A final system is likely to drop the rotating radar in favor of several static radars due to the increased complexity of rotating the radar, however the system as it stands is enough to test if a radar is sufficient for collision avoidance for a multirotor.

The system has been developed with support software that simplifies debugging, prototyping and actual development. The embedded software was also developed with the intent that parts of the hardware can be replaced.

4 | Object Detection

In this chapter we will discuss how to use the radar data in order to detect possible objects that the multirotor can collide with. Two algorithms are considered for this, SAR which sharpens the data, and a blob detection algorithm. The blob detection algorithm is tested on data gathered with the test platform from section 3.3.

Last semester, in [Koren 2016], a lot of tests a prototype of the system in chapter 3 was gathered. Some of this data is tested on the blob detection algorithm, and yields acceptable results.

Contents

4.1 Approaches to Object Detection

The data that is received from the radar makes every obstacle smeared out in both the direction of rotation and in range direction. In order to sharpen the information received from the radar, so that the data can be used directly, we consider SAR as a possible approach. We also consider a blob detection algorithm in which we can discern the approximate location of an object.

4.1.1 Synthetic Aperture Radar

SAR uses information gathered from moving relative to a radar target to create a higher resolution radar image, both in the radial (range) direction and in the direction of travel (cross-range). This could then give us precise locations, and maybe an estimate of size, of obstacles. Because SAR is a bit of a buzzword when it comes to radars, so it is worth discussing its use in a collision avoidance radar. Unfortunately SAR is not suitable for this purpose, and the reason for this is discussed in this section. The section is based upon information found in [Ch.17 Skolnik 2008], [Maître 2008], and [Moreira et al. 2013].

Why is Synthetic Aperture Radar Good?

With a conventional radar system it is necessary to have a very narrow beamwidth in order to get a good cross-range resolution[1]for a moving radar system. In order to create a narrow beamwidth, the aperture of the antenna has to increase, which also means an increase in physical size. For a parabolic antenna, this basically means that the larger the parabola, the more the energy can be focused. An approximation of the cross-range resolution δ_{cr} is given by:

$$\delta_{cr} \approx \frac{R\lambda}{D} \tag{4.1}$$

Here R is the range, λ is the wavelength and D is the aperture. Instead of using a large antenna to get a big physical aperture we can move a smaller antenna through space to achieve a synthetic aperture. The diameter of the synthetic aperture is given by $2L_{sa}$ where L_{sa} is the length of the traveled path. This means the longer the path of the antenna is, the higher resolution we have cross-range.

An explanation of why we have greater cross-range resolution with a synthetic aperture is that as the radar moves in a line parallel to the targets, each target in cross-range direction will have a different doppler frequency shift, as their speed in range direction is different. Each target point will therefore contribute to several measurements at a certain range in cross-range direction. We estimate how a point target is spread across

[1]Cross-range is the direction the radar system travels, it is usually perpendicular to the range direction, i.e. the pointing direction of the radar.

several measurements with a point spread function, whose spectrum we use later during SAR Processing.

SAR Processing

Data gathered from a SAR system before any processing is done is a 2 dimensional matrix with range in one direction and cross-range in another. Each cell of the matrix contains a complex value, the magnitude and phase of the returned data transformed to cartesian coordinates. In the ideal case the distance to the target surface is so far that we can assume a planar wave front, the antenna does not experience pitch or roll, and the antenna keeps a uniform distance to the target surface. For now we consider the ideal case, and our SAR pipeline is rather simple.

The data in our matrix has been smeared out in both dimensions. In range dimension it is smeared out by the fact that the chirp of our radar in the spatial domain is longer than each range bin. This causes reflections from nearby range bins to appear in each range bin. In addition the radar wave is returned from each target several times in cross-range direction, with a slightly different phase each time because of the doppler shift. In order to correct for this, the matrix is brought to the Fourier domain. The Fourier transformed matrix is convolved in range direction with the complex conjugate of the chirp spectrum to give range compressed data. Further, in order to correct for smearing along cross-range direction the transformed matrix is convolved with the complex conjugate of the frequency response expected from a point target.

Implementation Considerations

Unfortunately there are several obstacles in the way for implementing SAR for the system described in chapter 3. First and foremost, if the multirotor is standing still, rotating the system has no use because each target will have a constant velocity in range direction and we can not perform separation of targets in cross-range direction (direction of rotation). Secondly, if the quadrotor is moving and the system is rotating, measurements that are in the direction of travel are mostly useless for SAR purposes as there is no cross-range movement (synthetic aperture), which defeats the purpose of using it for collision avoidance. This could be somewhat fixed by offsetting the radar from the axis of rotation, similar to what has been done in [Ali, Bauer, and Vossiek 2014].

Removing rotation from the equation and just having one radar perpendicular to cross-range direction for imaging purposes, there are still problems that have to be solved. Because the range of the *X2M200* is so small, and the antenna has a large beam width, we can not assume planar waves and have to mitigate for the fact that a target can appear in several target bins during a pass. This is accounted for by an anti-range migration algorithm such as the *polar format algorithm*. Errors caused by the movement of the multirotor such as pitching and rolling of the antenna or not moving at a constant range of the target has to be accounted for.

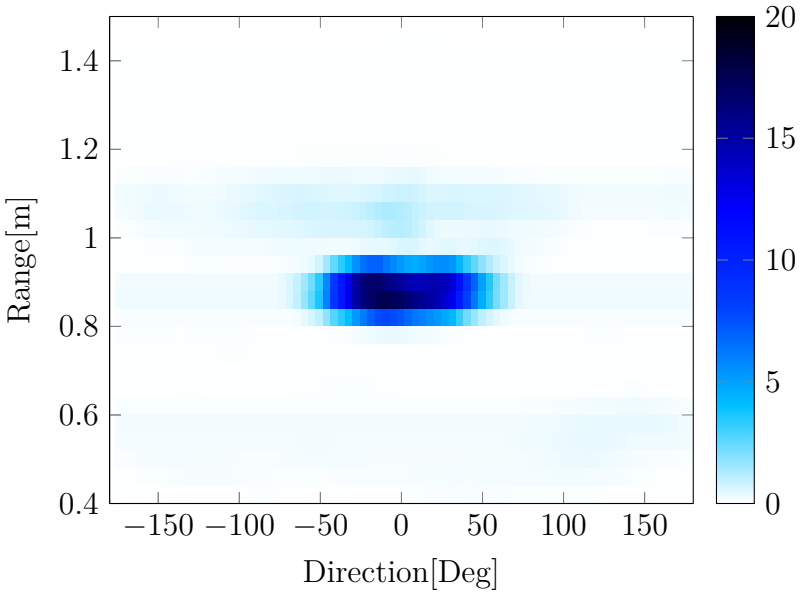


Figure 4.1: Interpretation of the magnitude of received reflections, shown as a matrix with pointing direction along x-axis and range along y axis.

4.1.2 Blob Detection

For collision avoidance we are only interested in the direction and distance to the target, as well as an approximation of its size. Using the data gathered from a rotating radar this is entirely feasible. Distance is an intrinsic part of radar data, a sense of direction is gathered by the radar rotating, while a sense of size can be gathered from the strength of the returned reflection and spread in the direction of rotation.

By looking at the matrix of received data, as shown in fig. 4.1, we see that an object appears as a blob of 4-connected measurements of higher value than their surroundings. Extracting the size of this blob, as well as its center gives us an estimate of the width and location of an obstacle.

Thresholding

It is important to filter out which points are to be considered as a part of a blob. The radar is prone to noise from clutter or multipath propagation, as well as the beam of the radar being very wide so that an element appears long after the radar is no longer pointing towards it. In order to ignore erroneous reflections a sufficient solution was to ignore all measurements with a magnitude under a certain threshold. As the reflections decrease by a power of 4 depending the range (see eq. (2.1)) the thresholding also needs to be dynamic. The following formula was used:

$$t(r) = k_t / r^4 \tag{4.2}$$

k_t is an adjustable gain, r is the range and $t(r)$ then gives the threshold for a specific range. All points that exceed this threshold can be considered for inclusion into a blob,

but adding an additional based on the peak value of the blob will ensure that what we include is more based upon the physical size rather than the radar cross section (i.e. amount of reflected energy). The additional threshold is applied to all points that are 4-connected to a local maxima:

$$t_{blob}(m_{peak}) = m_{peak}10^{\frac{k_{dB}}{20}} \tag{4.3}$$

m_{peak} is the magnitude of the maxima, k_{dB} a gain in decibel, and t_{blob} is the threshold.

A Greedy Approach

Another benefit of blob detection is that we can develop this as a greedy, memory conserving algorithm that is suitable for embedded devices. The data from the radar can be viewed as a matrix, with each column representing a specific pointing direction and each row representing a specific range. In this matrix a target will appear as a peak value spread out in both directions, as can be seen in fig. 4.1.

As we receive data column by column, we want to be able to process the data as it comes. This means we will conserve memory by not storing the data for a long period of time, as well actually detecting the target as soon as the radar has swiped over it.

There might be a lot of peaks within each blob due to noise or movement of the platform/target, so it is desirable to merge peaks that are likely to stem from the same object into the same blob. At the same time it is a good idea to ignore all peaks below a certain threshold value, as they are likely to be just noise.

Using this information we can develop the following algorithm:

1. Start with an array capable of holding information about each range bin, such as a reference to a blob and if the range bin was used in the current iteration.
2. A range bin is activated when the magnitude of a reflection for that bin exceeds the activation threshold, and a new blob is created for that bin.
3. Each new magnitude received for that bin is added to the blob if it exceeds the activation threshold as well as a local threshold based on the registered maximum magnitude for that blob.
4. After each column is processed, all blobs that are immediately adjacent in range-direction are merged into a bigger blob.
5. If no data was added to a blob after a column, then it is assumed that the blob has been found. All points that exceed the local blob threshold are included when calculating the distance and direction of the blob, as well as an estimation of the width of the blob. The ranges associated with the blob are cleared and marked as inactive.
6. Repeat from point 2 with a new column.

Figure 4.2: Test setup with one cylinder placed 0.92 m away from the radar at 0°.

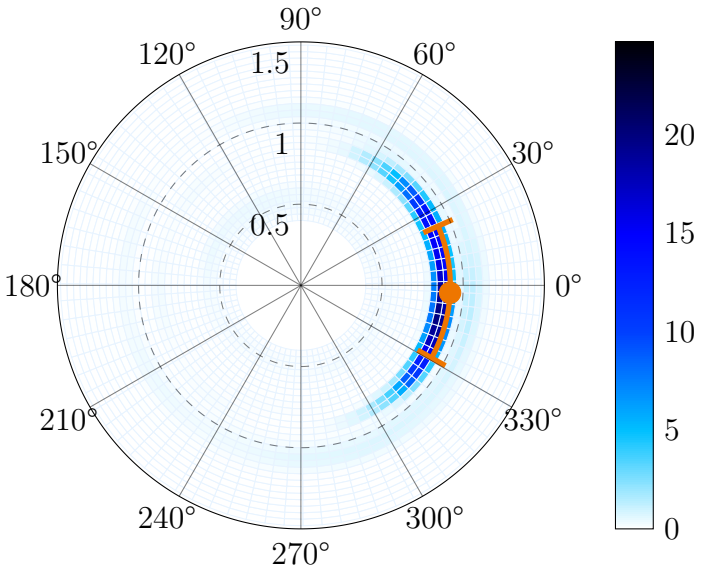
With the blob extracted from the algorithm we can either create a sector segment the multirotor has to avoid, or we can transform the direction and distance of the obstacle to the world frame and keep away from these coordinates. The algorithm as mentioned here is implemented and discussed a bit later in the thesis.

4.2 Testing Blob Detection

The algorithm described in section 4.1.2 was implemented in python in order to test its viability, and the software is documented in appendix A. The test data was gathered with the test rig from section 3.3, selecting from the data sets described in table B.1. Some error sources in the data that are worth noting is that the alignment between the radar 0° and the 0° of the setup was a bit off, in the range of 2° to 5°.

4.2.1 Test Results

The developed algorithm was tested on a few different scenarios aimed at testing how it detects single targets, multiple targets, large targets, and odd targets. For each test k_t (eq. (4.2)) and k_{dB} (eq. (4.3)) was tuned in order to maximize the accuracy of finding the blob center point. This was done because the width of the blob is useless without knowing the center.



Blob Color	Center r[m]	Center ϕ[deg]	Width [deg]
Orange	0.92 m	−2.83°	54.42°

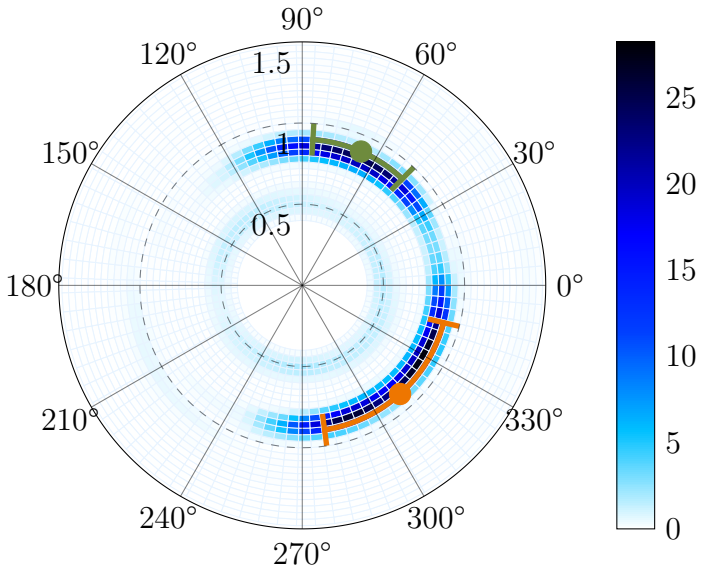
Figure 4.3: The result of the blob detection algorithm with a aluminum cylinder with a diameter of 0.31 m placed with the closes point at position $0.92\angle 0°$. This is the setup shown in fig. 4.2.

One Cylinder

In this test an aluminum cylinder, 0.31 m in diameter, was placed with a distance of 0.92 m between the radar and the surface of cylinder as shown in fig. 4.2. The cylinder was placed at 0°. Running python script, with $k_t = 2$ and $k_{dB} = -3$, we get the results shown in fig. 4.3. The center of the blob is quite close to the location of the point on the cylinder closest to the radar. However there is a slight offset in the angle which might be attributed either to a slight misalignment of the radar or that the dataset is not completely symmetric because of the radar antennas. The size of a sector that contains the physical cylinder can be shown to be 17°, while the calculated width is much larger. This can be reduced by setting k_{dB} to an even smaller value, but that affects the accuracy of the center position.

Two Cylinders

This test is quite similar to the other one. Two identical aluminum cylinders, 0.35 m in diameter, was placed 0.9 m away from the radar, at ±60° respectively. The script was run with $k_t = 3$ and $k_{dB} = -3$. The result is as shown in fig. 4.4, and here it did not work as well as in the last test. The green blob has been placed quite accurately, even though it is still too wide. As for the orange blob, both the center and the width is quite far off. This is reflected in the source data, which is shown shaded in blue, where the second cylinder also occupies a significantly larger sector. The cause behind this was not clear, but it might be a slight difference between the cylinders that caused this.



Blob Color	Center r[m]	Center ϕ[deg]	Width [deg]
Orange	0.90 m	−47.99°	68.52°
Green	0.90 m	66.30°	41.13°

Figure 4.4: The result of the blob detection algorithm with two aluminum cylinders with a diameter of 0.31 m placed at positions $0.9\angle \pm 60°$

More investigation is needed for this. The center positions are still usable if we avoid a large enough area around them.

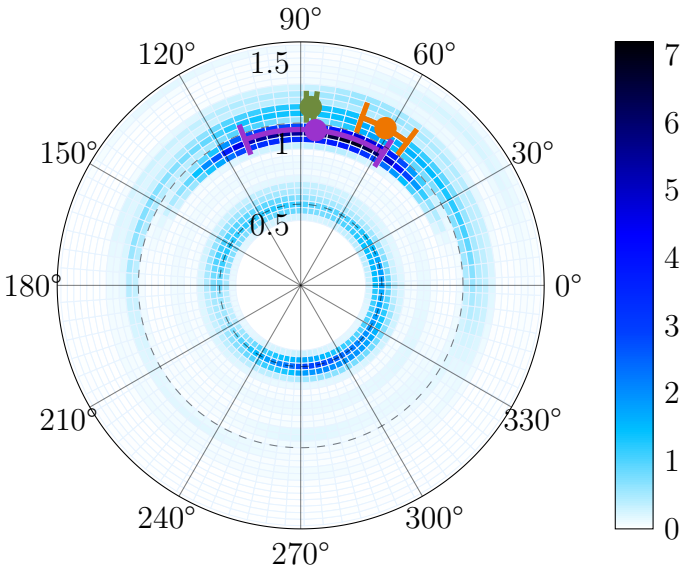
Concrete Wall

The radar system was placed 0.96 m away from a concrete wall, so that the closest point on the wall was at 90°. Concrete itself is not very visible to a radar, but the rebar and conduits inside the concrete is quite visible. This can also explain the erroneous detections in fig. 4.5, where the python script was run with $k_t = 3$ and $k_{dB} = -3$. The green and orange blob is likely rebar or conduits inside the wall, while the purple blob is the wall itself. The accuracy of the center coordinates of the purple blob is still quite good, with the inaccuracies likely to be caused by how the radar was positioned. Increasing the threshold gain k_t reduced the accuracy of the center of the purple blob, but removes the erroneous blobs. This is not really a valuable tradeoff as it is more useful with a good detection than to remove all false positives.

Also worth noting in fig. 4.5 is the amount of clutter close to the radar, which is likely caused by the test setup.

Human

A human is included simply because it is rather important for a multicopter to not hurt humans. The human (i.e. the author) placed himself at approximately 1 m and



Blob Color	Center r[m]	Center ϕ[deg]	Width [deg]
Orange	1.10 m	61.65°	18.27°
Green	1.10 m	86.76°	4.56°
Purple	0.96 m	84.48°	54.80°

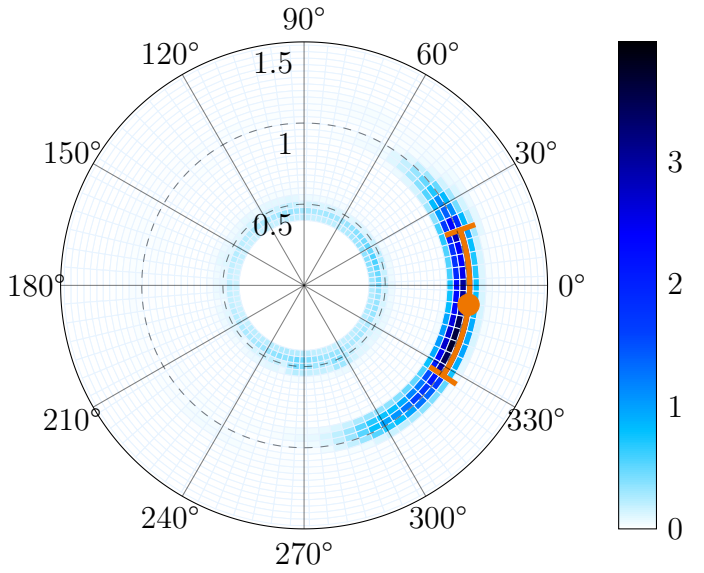
Figure 4.5: The result of the blob detection algorithm with a concrete wall placed at $0.96\angle 90°$.

0° and moved as little as possible. What is interesting about humans, compared to earlier targets, is that humans have a rather uneven surface which causes a much larger spread in range-direction as compared to earlier targets as seen in fig. 4.6. The python program was here run with $k_t = 1$ and $k_{dB} = -3$. As earlier, the center position of the blob is reasonably placed, and the width of the blob occupies a much larger section than the actual human.

4.3 Conclusion

When approaching the radar data we have available we see that SAR is not a good approach for detecting object to be used in collision avoidance. Instead an algorithm that is based on detecting blobs in images is considered. The algorithm, as it was presented in section 4.1.2, is capable of finding the center and closes point to an obstacle to a reasonably high degree. It is however unable to estimate the size of the object, which might also be a weakness with the radar system itself. As we are able to estimate the center of an obstacle we can just avoid that coordinate, as well as a safety zone around that coordinate. As we have no information on the height of an obstacle nor the shape, the simplest model that fit them all is an infinitely tall vertical cylinder. This forms the basis for the obstacles tested in the next chapter.

However the radar was not tested on a platform that was in motion, and it might be that the data from such a system would not work with the algorithm, except if the



Blob Color	Center r[m]	Center ϕ[deg]	Width [deg]
Orange	1.02 m	−6.71°	54.31°

Figure 4.6: The result of the blob detection algorithm with a human positioned at $1\angle 0°$.

radar system is able to update fast enough. The system also needs a update rate in order to update the position of the obstacles in order to avoid a collision. It might also be possible to chain data from several rotations, together with the movement of a multicopter, in order to estimate the size of an obstacle. As a concluding remark, the blob detection approach seems feasible for object detection on a rotating system that is not moving. More testing is needed to verify if it works for a moving system.

5 | Collision Avoidance

The topic of collision avoidance is getting more and more relevant to UAV research, as systems become more and more autonomous. It is also relevant for non-autonomous operations, as it can help the pilot avoid obstacles that is not visible to the pilot. At its roots, collision avoidance is basically the task of altering the input to a system in order to avoid it reaching a certain state. In this chapter we will look at one kind of collision avoidance, Null-Space-based Behavioral Control, how to implement it, and verifying the implementation on both a simple and a more complex simulator.

Contents

5.1 Null-Space-based Behavioral Control

Null-Space-based Behavioral (NSB) is a type of behavioral control, that is a combination of several simple behaviors that combined create a complex behavior. A behavior can be generalized as either keeping away from or moving towards a specific state. NSB was proposed in [Antonelli, Arrichiello, and Chiaverini 2005], and was a quite different approach compared to earlier methods. In overarching terms, NSB orders behaviors according to priority and allow higher priority behaviors to suppress parts of a lower priority behavior that collide with it.

5.1.1 Behavioral Control

When talking about behaviors in this context we think of how the quadrotor behaves in relation to its environment, such as keeping away from humans, avoiding obstacles, or aiming for a particular configuration. The behavior is expressed through a function of the system configuration, such as a cost or potential function. We call this function the task function. The output of the task function, which is aptly named the task value, is our measure for how well we conform to the desired behavior. When avoiding obstacles, the task function could be the 2-norm of the vector between origin of the body frame of the vehicle and the center point of the obstacle, i.e. the task value would be the distance between the vehicle and the obstacle. By controlling the task value, we are able to ensure that the vehicle conform to the expected behavior. Mathematically a task could be represented as follows:

$$\sigma = f(q) \tag{5.1}$$

with $\sigma \in \mathbb{R}^m$ representing the task variable we want to control, f is the task function, and $q \in \mathbb{R}^n$ is the system configuration. The dynamics of σ can be found through differentiation:

$$\dot{\sigma} = \frac{\partial f(q)}{\partial q}\frac{\partial q}{\partial t} \tag{5.2a}$$

$$= \boldsymbol{J}(q)v \tag{5.2b}$$

Here $J(q) \in \mathbb{R}^{m\times n}$ represents the jacobian of the task, and $v \in \mathbb{R}^n$ is the system velocity $\dot{q}$.

Now, by selecting a good v, it is possible to stabilize σ. This v can in turn be used as a reference velocity for our system. However, imagine that there are several such tasks, $\sigma_i,\ i \in [1, n]$, each with their own v_i. How will we ensure that two behaviors don't work against each other, or one behavior overrides another? Several such coordination schemes exist, such as a cooperative scheme where a supervisor assigns each v_i a weight, α_i, and then sum them all together to form a final v_d, or as written as a formula:

$$v_d = \sum_{i=1}^{n} \alpha_i v_i \tag{5.3}$$

Other schemes, such as a competitive subsumption architecture[Brooks 1986], give each task a priority, such that σ_1 has a higher priority than σ_2. The v_n of the lowest priority is cascaded up through the tasks of increasingly higher priority, either until it reaches a task that also wants to control v_d and therefore subsumes v_n in its own v_i. After passing through the last task, v_d is passed to the system as a reference. NSB is a subset of this competitive subsumption architecture.

5.1.2 How Null-Space-Based Behavioral Control Works

First the velocity v_i associated with each task σ_i has to be calculated. A simple solution to this problem is to use eq. (5.2) and invert it. This gives us the following:

$$v_i = \boldsymbol{J}^{\dagger}\dot{\sigma}_i \tag{5.4}$$

where $\boldsymbol{J}^{\dagger}$ is the Moore-Penrose Pseudoinverse

$$\boldsymbol{J}^{\dagger} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1} \tag{5.5}$$

A shortcoming with this practice is discrete integration of the reference velocity might cause a drift in the vehicle position. In order to solve this problem, [Antonelli, Arrichiello, and Chiaverini 2008b] used a closed loop inverse kinematics version of the formula:

$$v_i = \boldsymbol{J}^{\dagger}(\dot{\sigma}_i + \Lambda\tilde{\sigma}) \tag{5.6}$$

$\tilde{\sigma}$ is the task error $\tilde{\sigma} = \sigma_i - \sigma$, and Λ is a positive-definite matrix. Inserting eq. (5.6) into eq. (5.2) gives

$$\dot{\tilde{\sigma}} = -\Lambda\tilde{\sigma} \tag{5.7}$$

Which show that the error is asymptotically stable if Λ is positive definite, and that eq. (5.6) is a good input for eq. (5.2).

Now the clever thing about NSB is how the cascaded reference velocities are subsumed by the higher reference tasks. What we want to avoid is for the lower priority task to alter the completion of a higher priority task in any way. By projecting the lower priority reference velocity onto the null space of the jacobian of the the immediate

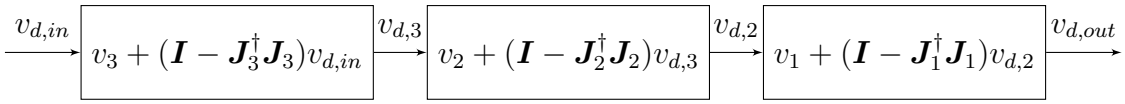


Figure 5.1: A representation of how eq. (5.9) would work if we have three tasks

higher-priority task, the conflicting components of the reference velocity is removed. I.e. only the velocity component that causes no change in task value remains. The null space of task σ_i can be defined as follows:

$$\text{null}_\text{i}(\sigma) = \boldsymbol{I} - \boldsymbol{J}_i^\dagger \boldsymbol{J}_i \tag{5.8}$$

With this we can create the following recursive formula for a system with n tasks.

$$v_{d,out} = v_{d,1} \tag{5.9a}$$
$$v_{d,i} = v_i + (\boldsymbol{I} - \boldsymbol{J}_i^\dagger \boldsymbol{J}_i) v_{d,i+1} \tag{5.9b}$$
$$v_{d,n+1} = v_{d,in} \tag{5.9c}$$

Here $v_{d,in}$ is the external velocity reference fed into NSB, $v_i, i \in [1, n]$ is the task desired velocity generated by eq. (5.6), and $v_d, i \in [1, n]$ is the output velocity generated by each task. The variable i is used to signify priority, with a lower i equal to higher priority. The equations system eq. (5.9) is graphically represented in fig. 5.1. Now in the case where J_i is of full rank, such that there is no null space, the lower priority references will be completely filtered out. However we can guarantee that we fulfill the highest priority task, as well as lower priority tasks that do not interfere with the highest priority task [Antonelli, Arrichiello, and Chiaverini 2008a].

Selecting task gain Λ

[Antonelli 2008] presents a method for selecting the individual task gains that guarantee stability for the different tasks. The conditions for stability are presented for a scenario with three task jacobians. Each jacobian either has to be independent from all higher priority jacobians stacked on top of each other, or it has to be the annihilator of the jacobian of the immediate lower priority task while the remaining jacobian is independent from both. In other words:

$$\boldsymbol{J}_{12} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} \tag{5.10a}$$
$$rank(\boldsymbol{J}_1^\dagger) + rank(\boldsymbol{J}_2^\dagger) = rank([\boldsymbol{J}_1^\dagger \ \boldsymbol{J}_2^\dagger]) \tag{5.10b}$$
$$rank(\boldsymbol{J}_{12}^\dagger) + rank(\boldsymbol{J}_3^\dagger) = rank([\boldsymbol{J}_{12}^\dagger \ \boldsymbol{J}_3^\dagger]) \tag{5.10c}$$

or

$$\boldsymbol{J}_n \boldsymbol{J}_{n+1}^{\dagger} = 0, \ j = 1, 2 \tag{5.11a}$$
$$rank(\boldsymbol{J}_n^{\dagger}) + rank(\boldsymbol{J}_{n+2}^{\dagger}) = rank(\boldsymbol{J}_n^{\dagger} \ \boldsymbol{J}_{n+2}^{\dagger}) \tag{5.11b}$$
$$rank(\boldsymbol{J}_{n+1}^{\dagger}) + rank(\boldsymbol{J}_{n+2}^{\dagger}) = rank(\boldsymbol{J}_{n+1}^{\dagger} \ \boldsymbol{J}_{n+2}^{\dagger}) \tag{5.11c}$$
$$\tag{5.11d}$$

The gain Λ_n can then be chosen as $\Lambda_n = \lambda_n \boldsymbol{I}$, with $\boldsymbol{I}$ as the identity matrix of dimension equal to σ. λ_n is chosen as

$$\lambda_1 > 0 \tag{5.12a}$$
$$\lambda_2 > max(0, \ \frac{\bar{\lambda}_{21} - \underline{\lambda}_{11}}{\underline{\lambda}_{22}} \lambda_1) \tag{5.12b}$$
$$\lambda_3 > max(0, \ \frac{\bar{\lambda}_{31} - \underline{\lambda}_{11}}{\underline{\lambda}_{33}} \lambda_1, \ \frac{\bar{\lambda}_{32} - \underline{\lambda}_{22}}{\underline{\lambda}_{33}} \lambda_2) \tag{5.12c}$$

The values $\bar{\lambda}_{nn}$ and $\underline{\lambda}_{nn}$ are the maximum and minimum eigenvalues of the matrix $\boldsymbol{P}_{nn}$ that can be calculated with the following equations:

$$\begin{aligned} \boldsymbol{P}_{11} &= \boldsymbol{I}, & \boldsymbol{P}_{22} &= \boldsymbol{J}_2 null(\boldsymbol{J}_1) \boldsymbol{J}_2^{\dagger}, \\ \boldsymbol{P}_{21} &= \boldsymbol{J}_2 \boldsymbol{J}_1^{\dagger}, & \boldsymbol{P}_{32} &= \boldsymbol{J}_3 null(\boldsymbol{J}_1) \boldsymbol{J}_2^{\dagger}, \\ \boldsymbol{P}_{31} &= \boldsymbol{J}_3 \boldsymbol{J}_1^{\dagger}, & \boldsymbol{P}_{33} &= \boldsymbol{J}_3 null(\boldsymbol{J}_{12}) \boldsymbol{J}_3^{\dagger} \end{aligned} \tag{5.13}$$

By using these gains with a maximum of three tasks we ensure that the task value error $\tilde{\sigma}$ will go to 0 if conditions in eq. (5.10) and eq. (5.11) are satisfied.

5.1.3 Task Activation

When using NSB it is not always desired that a task should be active. One such example could be a fencing behavior, where the system is to stay within a certain distance of a specific configuration. Typically this distance would be represented as σ_d. If the task was to be active at all times it would cause the system to go towards this boundary, while ignoring input in and opposite to the direction of the boundary. This is obviously not desired behavior, and some care need to be put into when to activate tasks. The topic was briefly touched upon in [Antonelli, Arrichiello, and Chiaverini 2008b], suggesting to activate a task when the task value is within a certain threshold and the direction of the velocity vector coincides with the direction of the obstacle.

Velocity-vector Approach

Activating when the velocity vector $v, d, n+1$ is intersecting with a boundary drawn by the desired task value was tested in [Klausen 2013]. Here the author considers the following task, avoiding a circle:

$$\sigma = \|\boldsymbol{q} - \boldsymbol{c}\| \tag{5.14}$$

The coordinates of the center of the circle is represented by c. In order to activate the task, he finds the length of the distance, σ_v, which is the vector between the center c and the projection of c onto the velocity vector. The angle θ_v between the velocity vector input to the task, and the vector from q to c is also considered. The task is activated if

$$\sigma_v < \sigma_d \tag{5.15}$$

$$\|\theta\| \in \left[0, \frac{pi}{2}\right) \tag{5.16}$$

While this is a good and valid approach, another approach is tested out in this thesis. Any point along the velocity vector can be calculated by the following formula:

$$p = q + tv_{d,n+1} \tag{5.17}$$

This can be inserted into the equation for the task function eq. (5.1) to find how the task function develops along the ray, i.e.:

$$\sigma_d = f(q + tv_{d,n+1}) \tag{5.18}$$

Solving this equation for t and concluding with $t > 0$ means that the velocity vector is intersecting the boundary formed by σ_d. The drawback of this implementation is that it has to be solved specifically for each kind of task function.

Here we solve it for the task function eq. (5.14). First note that the task function can be rewritten as:

$$\sigma = (q - c) \cdot (q - c) \tag{5.19}$$

Now inserting eq. (5.17), replacing q with $q + tv_{d,n+1}$, and solving for t gives the following second degree polynomial:

$$t^2(v_{d,n+1} \cdot v_{d,n+1}) + t(2v_{d,n+1} \cdot (q-c)) + (q-c) \cdot (q-c) - \sigma^2 = 0 \tag{5.20}$$

By using the quadratic formula[Brahmagupta 0628] to solve for t, we get to know if the velocity vector intersects the boundary circle. If both solutions, t_1 and t_2, are positive and real, the velocity vector is in a direction that will intersect with the boundary circle. The obvious drawback of this method is that it is not general and needs to be derived specifically for each kind of task function. However a similar approach of finding the intersection between the velocity vector and the boundary might work for other task functions.

[Antonelli, Arrichiello, and Chiaverini 2008b] also talks of a maximum activation distance, which basically defines the interval for which we bother to consider activating the task. In the use case where an operator wants to operate close to a task limit at low speeds we do not want the task to activate unless there is a chance of collision. However in the scenario that the operator is not aware of the surroundings and is about to ram into the task limit at maximum velocity, we want to activate the task as soon as possible to slow down the vehicle. This is possible to achieve using a dynamic activation distance, such as one based on the velocity in the direction of increased task value.

By integrating $\ddot{\sigma}$, acceleration in direction of increased task value, twice with respect to time we get the two following equations. Note that this assumes that σ is scalar.

$$\dot{\sigma} = \ddot{\sigma}t + \dot{\sigma}_0 \tag{5.21}$$

$$\sigma = \frac{1}{2}\ddot{\sigma}t^2 + \dot{\sigma}_0 t \tag{5.22}$$

By setting $\dot{\sigma} = 0$, solving eq. (5.21) with regards to t and inserting in eq. (5.22). Assuming the maximum acceleration $\ddot{\sigma}_{max}$ has a different sign than $\dot{\sigma}$ we get the following equation for the distance takes to break the system:

$$\sigma_{stop} = \frac{1}{2}\frac{\dot{\sigma}^2}{\ddot{\sigma}_{max}} \tag{5.23}$$

This gives the final maximum activation distance

$$\sigma_a = \sigma_d + \sigma_{stop} \tag{5.24}$$

This activation distance can then be calculated on every new measurement in order to decide whether or not to check for a possible collision.

In fig. 5.2 we can see how such a system could work. The velocity vector intersects with the circle and the craft is inside the activation distance. NSB will then remove the component of the vector that is in radial direction so that only the tangential component remains.

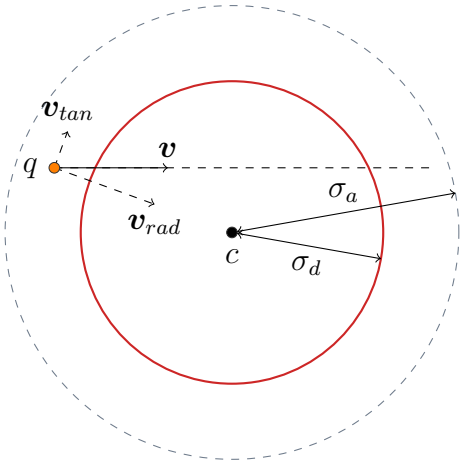


Figure 5.2: This figure shows a craft with velocity vector v, decomposed into tangential and radial components. The desired task value σ_d is shown with the red ring, while the maximum activation distance σ_a is is shown with the dashed line. The center of the circle is shown with c.

Set-based NSB

As an alternative to the activation scheme presented earlier [Moe et al. 2016] introduces Set-based NSB tasks, where the goal of steering the task value to a certain desired task value has been replaced with a goal of keeping the task value within a predefined interval, $\sigma \in D$. When the task value is inside D, there is no need for the task to affect the behavior of the system. This makes activation/deactivation of the task to be an integral part of the task itself.

The interval in which the task value is valid, $D = [\sigma_{min}, \sigma_{max}]$ forms the basis for the extended tangent cone[1], a key property used in the activation/deactivation of the set-based task. The extended tangent cone to the set D is defined as

$$T_{\mathbb{R},D}(\sigma) = \begin{cases} [0, \infty), & \sigma \leq \sigma_{min} \\ \mathbb{R}, & \sigma \in (\sigma_{min}, \sigma_{max}) \\ (-\infty, 0], & \sigma \geq \sigma_{max} \end{cases} \tag{5.25}$$

If the derivative of the task value is inside the extended tangent cone, $\dot{\sigma} \in T_{\mathbb{R},D}$, then we can see that $\sigma \in D \forall t \geq t_0$ if $\sigma(t_0) \in D$ as well as $\sigma \to D$ when $t \to \infty$ if $\sigma(t_0) \notin D$. When $\dot{\sigma} \in T_{\mathbb{R},D}$ there is no need for the task to be active, and the desired velocity does not need to be altered. However, in the case when $\dot{\sigma} \notin T_{\mathbb{R},D}$ the task is active and the velocity propagates through the system as in eq. (5.9), with v_i defined as

[1]The non-extended tangent cone is identical, except for only being defined for $\sigma \in D$.

$$v_i = \boldsymbol{J}^{\dagger}(\dot{\sigma}_i + \Lambda\tilde{\sigma}) \tag{5.26a}$$

$$\tilde{\sigma} = \begin{cases} \sigma_{min} - \sigma, & \sigma \leq \sigma_{min} \\ \sigma_{max} - \sigma, & \sigma \geq \sigma_{max} \end{cases} \tag{5.26b}$$

A stability analysis of the Set-based NSB approach was perfomed in [Moe et al. 2015], which concluded that all set based tasks would be satisfied if they had higher priority than equality-based NSB tasks if the conditions in section 5.1.2 are fulfilled.

5.2 Implementation

An implementation of NSB was written in C++ for the DUNE runtime environment. The possibility of both equality-based and set-based NSB tasks was considered, as well as possibility to further extend the software should it be necessary.

5.2.1 Specification and Planning

A small list of goals that the implementation of the Set and Equality-based NSB tasks had to conform to was made. The motivation behind the goals is to simplify maintenance, as well as usage, of the software for a future maintainer.

1. Must have both set-based and equality-based tasks.
2. Must be possible to implement custom tasks.
3. Must be able to manage priorities.

The following text discusses how NSB was implemented in order to fullfill these requirements.

5.2.2 Hierarchy

In order to ease flexibility of the program it was decided to separate task management from task functionality. Task management is responsible for organizing tasks in order based on priority, as well as cascading the desired velocity through the tasks in order. Task functionality is more based around implementation of each task function, calculation of the jacobian, as well as calculating the velocity input.

Task Functionality

In order to implement task functionality a class hierarchy as shown in fig. 5.3 was drawn up. The overarching idea is that main functionality, which has to be calculated no matter what kind of task, is implemented at the top level, in *NSBTask*, as well as

pure virtual function prototypes (in cursive) that has to be implemented by a subclass. Additionally, all functions on all levels are virtual and can be overridden by a subclass should it be necessary for a task to deviate a bit from the norm.

In fig. 5.3 we see an abbreviated version of how this would work. Null-space projection and calculation of the task value and its derivative among others are implemented in the base class *NSBClass*. The base class presents a coherent interface towards the task management functionality, as well as presenting pure virtual functions that are implemented by the subclasses. One such example is the *cascadeInput* function, which has to be implemented differently for a set-based task vs. an equality based task. The subclasses may also have their own virtual functions, such as the *shouldBeActive*-function of *EqTask* which is used for implementing the Velocity-vector based approach to task activation as described in section 5.1.3.

Task Management

Task management is implemented in the *NSB* class. The main idea is that each *NSBTask* object is associated with an ID and a priority. The ID is used to identify the task later, when changing priority or deleting the task. The function *processInput* provides a simple interface for cascading the input through the tasks in order based on priority.

This class could be extended with more management opportunities, such as dynamic enabling and disabling of certain tasks based on other external data.

5.2.3 Limitations

There are some limitations in the software that has been developed, I will try to list them, as well as how to mitigate them.

Scalar Task Values Task values can only be scalar. While this limitation is inherent to set based tasks, there are circumstances where a vector task value could be desired for an equality-based task. The best way to mitigate this is to create several equality-based tasks, one for each element in the vector task value, and insert them with adjacent priorities.

Completely Arbitrary Priorities When adding a task to an *NSB*-object it is not possible to assign a task a higher priority than the number of tasks already in managed by the object. One scenario where this might become a problem is if the object is managing zero tasks and three tasks are to be added. If the tasks are added in order with priorities 2, 1, and 0 they will end up with final sorting order 0, 2, and 1. This is because the task with priority 2 gets reset to 0 when it is added, and then the task with priority 1 gets the bottom priority. When the task with actual priority 0 is added it gets the top priority. This is a feature of how the *NSB*-object is implemented. Should it become a problem, one solution is to reset the priority of all tasks with priority lower than the last added priority.

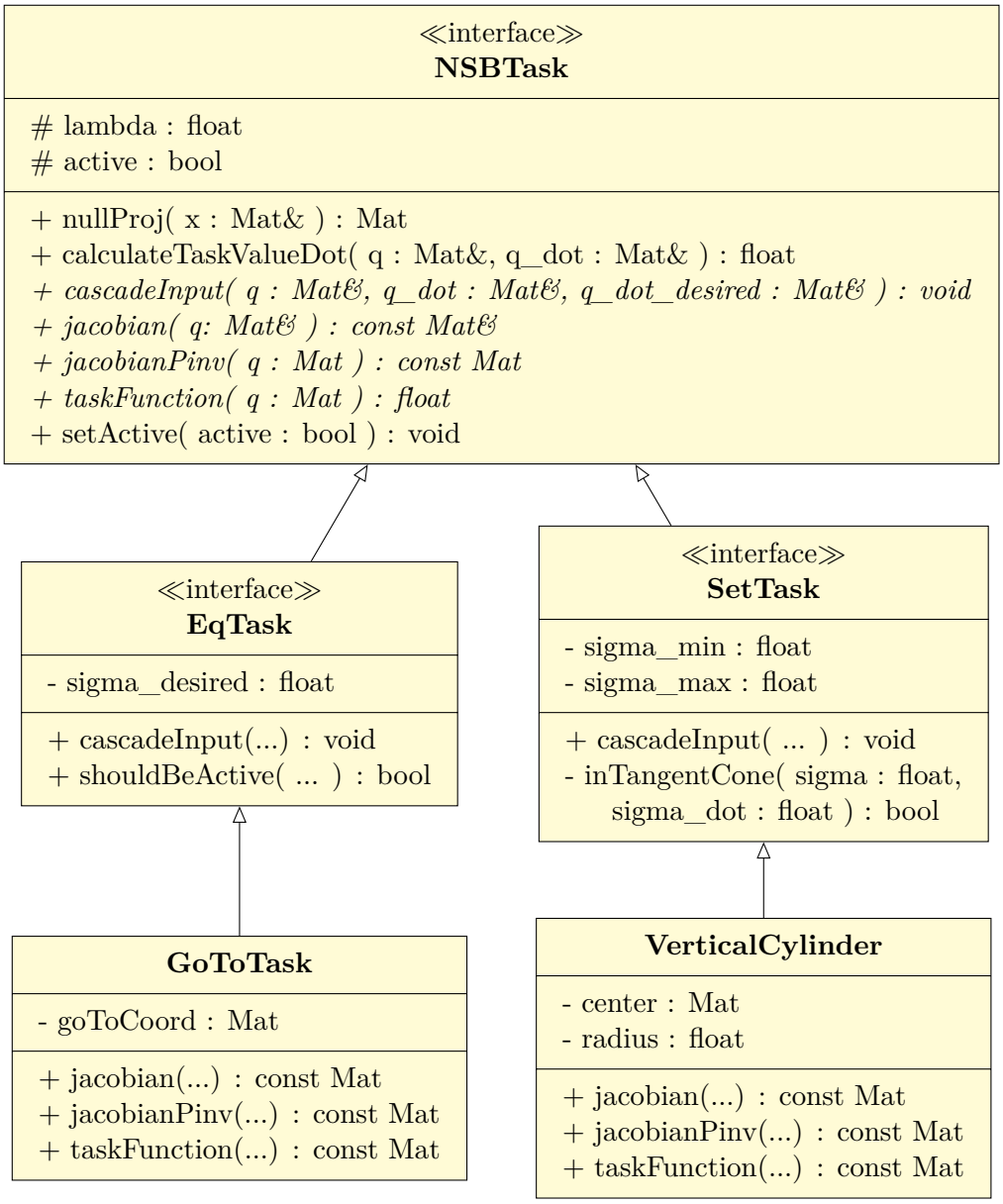


Figure 5.3: Class diagram over the NSB-software. NSBTask is an interface which is partly implemented by EqTask and SetTask, which perform operations specific to equality-based tasks and set-based tasks. Two kinds of tasks are shown here, a GoToTask which will try to pull the quadrotor towards a specific coordinate, as well as VerticalCylinder, which is an obstacle represented as an infinitely high cylinder. The task function and the jacobians are implemented in these classes. The diagram does not contain all the functions in the actual implementation of the class in order to better get the structure across.

NSB
- task_map : map<string, NSBTask*> - task_priority : list<NSBTask*>
+ newTask(t : NSBTask, id : String, prio : Int) : void + changePriority(id : String, prio : Int) : void + getPriority(id : String) : Int + removeTask(id : String) : void + getTask(id : String) : NSBTask + processInput(q : Mat&, q_dot : Mat&, q_dot_desired : Mat&) : Mat

Figure 5.4: Showing the structure of the class that organizes NSBTask-objects according to their priority. Each task is associated with an ID, given as a string. A good ID could be a description (for a persistent task) or a Universally Unique Identifier (UUID) for short-lived tasks. The function processInput is used to cascade an input through all tasks in order of priority and returns an input that adheres to the defined behaviors.

5.2.4 Tasks

Three different kinds of behaviors were implemented as a part of this thesis; obstacle avoidance, ground clearance, and a go-to position behavior. The tasks where chosen as they were each able to prove different points. A cylinder task was chosen as a generic obstacle, as every obstacle can be covered with a cylinder that is large enough. The go-to position behavior was chosen as it is useful for testing the obstacle avoidance. Ground clearance was also included because it is a one-dimensional task, and it shows of some interesting dynamics.

Go to position This is an equality-based task which aims to move the multirotor to a fixed position. The task has the following task function:

$$\sigma(q) = \|q - a\|_2 \tag{5.27a}$$
$$\sigma_d = 0; \tag{5.27b}$$

Where q are the coordinates of the quadrotor, a are the coordinates of the desired position and σ_d is the desired task value. The jacobian of this task function is

$$J(q) = \frac{(q-a)^T}{\|q-a\|_2} \tag{5.28}$$

Worth noting about this task is that an additional magnitude saturation step was added on the task-specific desired velocity v_i. This was added because the GoTo-coordinate might be placed far away from the multirotor, which could cause the desired velocity to

be very large. The alternative, which is performing a saturation on the final output of the NSB algorithm might unfairly punish velocity components from other tasks which have a smaller desired velocity.

Vertical Cylinder Obstacle Two different vertical cylinder tasks was implemented. One which is a set-based NSB task, another equality-based one which is built upon the activation algorithm from section 5.1.3. They similar in all aspects except for the conditions at which they activate.

Now the cylinder is modelled as an infinitely tall vertical cylinder. The task function, as well as the jacobian is identical to the goto-task, eq. (5.27) and eq. (5.28), except for only using the two first elements of q, ignoring the down-component. The desired task value, σ_d/σ_{min}, is set to the radius of the cylinder plus the minimum safe distance between the origo of the body frame and the cylinder wall.

Ground clearance A simple set based task, with $sigma_{min} = -\infty$ and σ_{max} equal to the ground clearance (assuming that a d-value of 0 is the ground). The task value is simply the d-value of the position of the multirotor, while the jacobian is a unit vector pointing in positive d-direction.

5.3 Simulations

The software was verified by testing different scenarios on a simple point mass simulator in DUNE. The simulator runs at 100 Hz and it reports its own position in the NED frame once for every timestep. The simulator accepts a velocity as an input, which is immediately set as the velocity of the point.

When designing the tests the following goals were set

1. Verify that a behavior will be followed.
2. Verify that the priority of the tasks matter.
3. Check for edge cases in which behavior is not optimal.

Testing the equality-based vs. the set-based cylinder task was not prioritized here, and is instead a topic that is brought up later in the chapter.

5.3.1 Test 1: Following expected behavior

This test consists of two different scenarios. The setup of the first scenario is detailed in table 5.1. The scenario consists of the simulated craft starting 10 m above ground at the origin. A vertical cylinder is placed a bit away from the origin and the minimum distance to the center of the cylinder is set to 4 m. A velocity vector pointing in a

Variable	Value
Start position [NED]	$[0, 0, -10]^T$
Velocity input [NED]	$[1, 1, 0]^T$

(a) Various initial values

Task	λ	Coordinates	σ_{min}	σ_{max}
Vertical Cylinder	1	$[5, 4.9, x]^T$	4 m	∞

(b) Set-based tasks

Table 5.1: The setup variables for the first scenario of test 1. The start position is the initial position of the simulator in NED-coordinates. The velocity input is the initial input that is cascaded through the system. The obstacle is placed so that the simulated craft will have to pass around it, keeping a distance of 4 m from the center of the obstacle.

direction that will cause a colition with the obstacle is fed into the NSB algorithm as a starting velocity[2].

The final result of this scenario is shown in fig. 5.5, here we can see the craft behaving similar to an object sliding off a sphere. As the velocity component to the tangent of the constraint, i.e. in the null space of our jacobian, is very small, the craft slides around the obstacle very slowly at initial impact with the constraint.

In the second scenario, the velocity input was replaced with a GoTo-task, which aimed at reaching the coordinate $[10, 10, 10]^T$. The setup for this task is shown in table 5.2. The behavior, which is shown in fig. 5.6, is quite similar to what we saw in the previous scenario, fig. 5.5, however we can see the interaction between the GoTo-task as well as the Cylinder-task. Because the Cylinder-task has the highest priority, it overrides the velocity component desired by the GoTo-task that would result in violation of the Cylinder-task (and a potential collision with the associated obstacle). The vehicle glides smoothly around the obstacle and ends up at the specified coordinate.

5.3.2 Test 2: Task priority

As in the previous test, this test also consist of two different scenarios. The setup is detailed in table 5.3. In both scenarios the simulated craft starts 10 meter above ground at origo, and has a GoTo-behavior at coordinate $[10, 0, -2]$ as well as a ground clearance behavior that tries to keep the task 4 meters above ground (-4 m).

In the first scenario, whose result is shown in fig. 5.7a, the ground clearance behavior has a higher priority than the GoTo-behavior. This causes the craft to "slide" along the Ground Clearance-task while the GoTo-task tries to minimize the distance between the

[2]If the velocity vector pointed directly at the center of the obstacle we would end up with a fringe case that does not work on such a simple simulator. When the craft eventually reaches the obstacle, the null-space projection of the velocity vector will be 0 and the craft would be stuck. Because of measurements inaccuracies this would solve itself on an actual quadrotor.

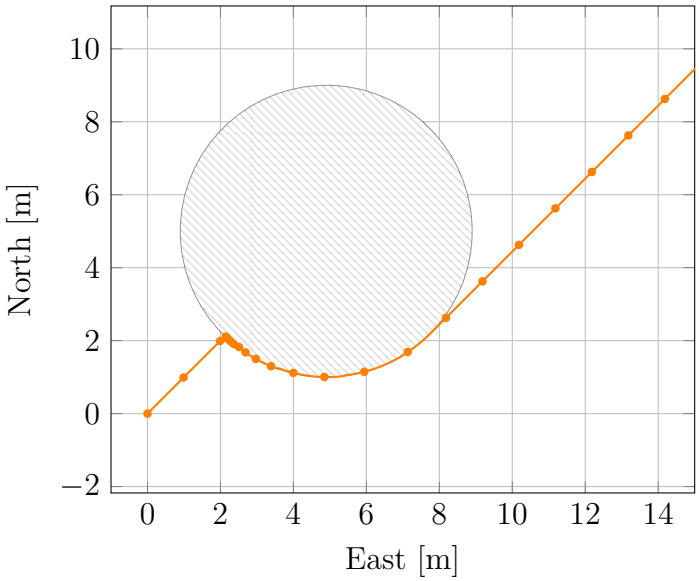


Figure 5.5: This is a simulation of the setup in table 5.1. The hatched circle represents the cylinder, while the orange line is the trajectory of the simulated craft. The trajectory has a mark every second, to give an idea of the speed of the craft.

Variable	Value
Start position [NED]	$[0, 0, -10]^T$
Velocity input [NED]	$[0, 0, 0]^T$

(a) Various initial values

Task	λ	Coordinates	σ_{min}	σ_{max}	Priority
Vertical Cylinder	1	$[5, 4.9, x]^T$	4 m	∞	0

(b) Set-based tasks

Task	λ	Coordinates	$\sigma_{desired}$	Priority
GoTo	1	$[10, 10, -10]^T$	0	1

(c) Equality-based tasks

Table 5.2: The setup variables for the second scenario of test 1. The initial velocity from table 5.1 is replaced with a GoTo-behavior which causes the craft to aim for a point on collision course with the vertical cylinder. .

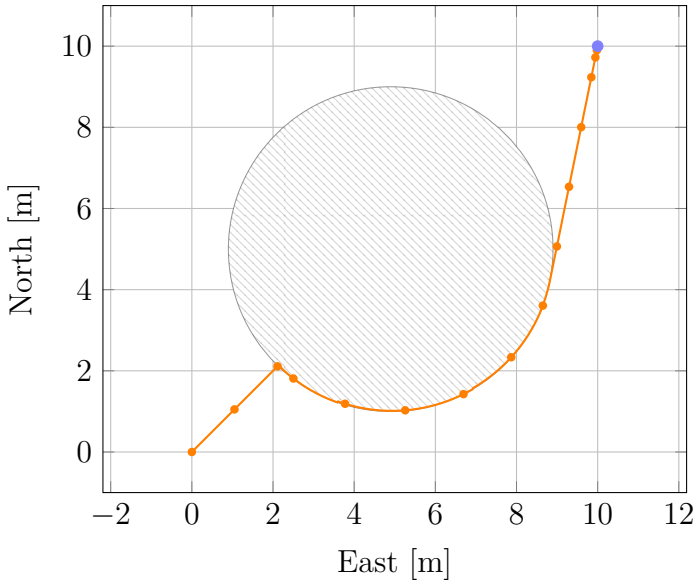


Figure 5.6: This is a simulation of the setup in table 5.2. The hatched circle is the cylinder, while the orange line is the trajectory of the craft. The trajectory has a mark for every second. The light blue dot is the coordinate of the goto task.

craft and the GoTo-location. Without the ground clearance constraint the trajectory would have been a straight line.

In the second scenario the priority of the two behaviors have been switched around, and the result is clearly shown in fig. 5.7b. We see that switching priorities causes the craft to proceed to the GoTo-coordinate, and that the priority system works. We can also see an artifact caused by the conditions of eq. (5.10) not being fulfilled. In order to remove the violation of the ground clearance task, the task has a desired velocity upwards. This component cannot be removed by the GoTo-task as it is not in the null-space of the task, and it causes the craft to slow down. The λ of the GoTo-task therefore has to be larger than the λ of the Ground Clearance-task in order to reduce the effect of this somewhat.

5.3.3 Test 3: Unintended behavior

Testing several difference scenarios, only one ended up with what could be defined as unintended behavior. It shows a quirk in the Set-based NSB approach. The setup is identical to table 5.3, with the ground clearance behavior as highest priority. The only difference is that the starting position of the craft is at $[0, 0, 0]^T$ instead of 10 meters above the origin. The end result, as shown in fig. 5.8, is clearly not optimal as the craft has a long period standing almost still at a height of 2 meters. This is because the craft still has a velocity upwards when approaching 2 meters, which cause it to be inside the extended tangent cone, and the ground clearance task does not become active until the craft is at a standstill. This gives a case for just using the non-extended tangent cone to activate a task, or to activate the task anyway if the violation is above a certain threshold.

Variable	Value
Start position [NED]	$[0, 0, -10]^T$
Velocity input [NED]	$[0, 0, 0]^T$

(a) Various initial values

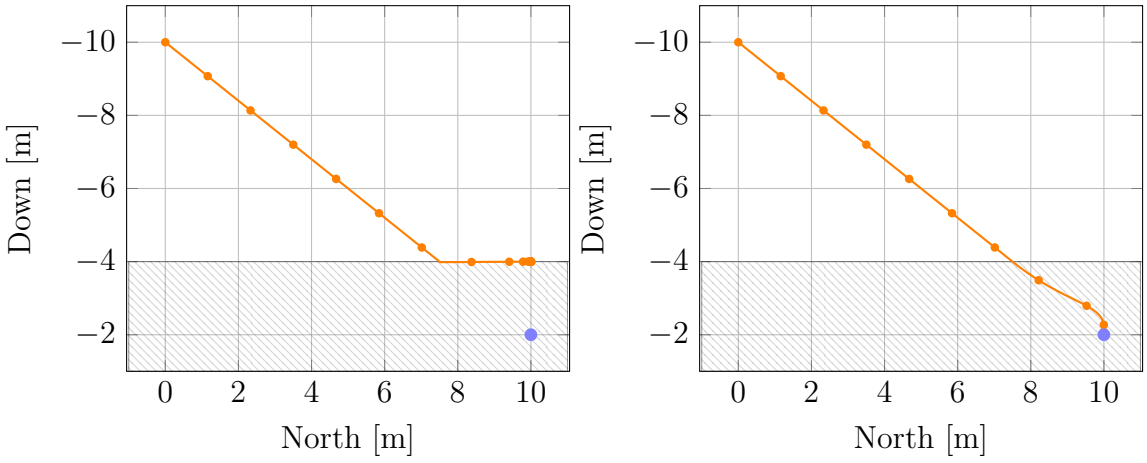
Task	λ	Coordinates	σ_{min}	σ_{max}	Priority
Ground clearance	1/1	N/A	$-\infty$	$-4\,\mathrm{m}$	0/1

(b) Set-based tasks

Task	λ	Coordinates	$\sigma_{desired}$	Priority
GoTo	1/1.2	$[10, 0, -2]^T$	0	1/0

(c) Equality-based tasks

Table 5.3: The setup for the scenarios considered in test 2. The craft starts 10 meters above ground with a destination point below the ground clearance constraint. In the cases where there is a difference between the two scenarios, the values are separated by a slash with scenario 1 on the left. .



(a) Ground avoidance has highest priority.

(b) GoTo has highest priority

Figure 5.7: These simulations of test two show north vs down as the simulated craft does not move in the east-direction. The orange line is the trajectory of the craft, with a mark for every second. The hatched area is the area forbidden by the ground avoidance task, while the blue dot is the coordinate of the GoTo-task.

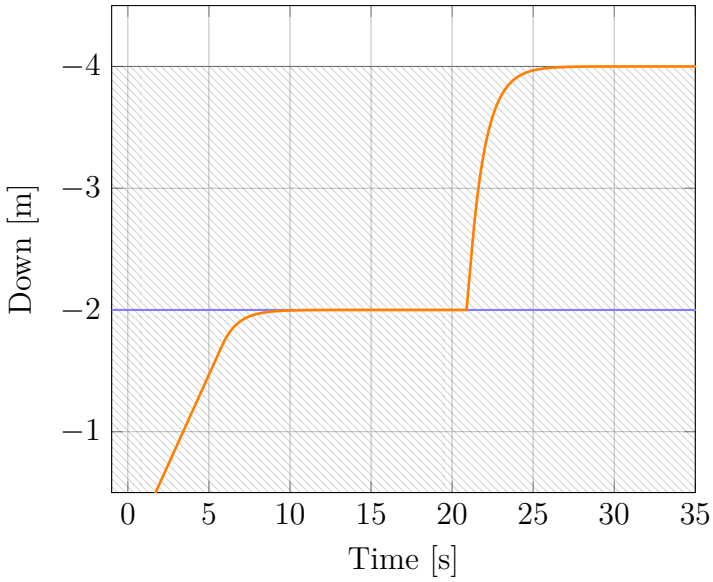


Figure 5.8: This plot shows the down coordinate as a function of time. The orange line is the trajectory of the craft, the light blue line is the height of the GoTo-task, while the hatched area is the area forbidden by the ground avoidance task.

5.4 Software in the Loop Testing

This section is about further testing of the implemented NSB algorithm. The point simulator from earlier is replaced with a simulated quadrotor, and a reference model is added in order to get a acceleration output from the velocity reference from the NSB algorithm.

5.4.1 Reference model

The multirotor controller accepts an acceleration vector in the North-East-Down (NED) frame as a reference, but the output from the NSB algorithm is in this case a velocity reference. The reference acceleration could be produced with a P or PI-controller, but then the acceleration reference would be neither smooth nor could it be guaranteed to be within the bounds of the quadrotor.

Another option is to use a reference model that consists of a low pass filter of sufficient order, in a manner inspired by [Klausen, Fossen, and Johansen 2015, p.8]. For our reference model we consider a third order low-pass filter

$$\dddot{v} + (2\zeta + 1)\omega_0 \ddot{v} + (2\zeta + 1)\omega_0^2 \dot{v} + \omega_0^3 v = \omega_0^3 v_d \tag{5.29}$$

This low-pass filter, constructed by one cascading a generic first order and a generic second order low-pass filter, has ξ as the damping coefficient and ω_0 as the cutoff frequency. $\dddot{v}, \ddot{v}, \dot{v}$, and $v \in \mathbb{R}^3$ are the reference jounce, jerk, acceleration and velocity respectively. v_d is the desired velocity.

Variable	Value
ω_0	4
ζ	0.9
$\dot{v}_{max}$	2
$\ddot{v}_{max}$	5

Table 5.4: The parameters used in the reference model eq. (5.29) and eq. (5.32).

Thankfully we are in complete control of this reference model, so we can decide that our input to the system is the jounce, $\dddot{v}$. In order to guarantee that the reference model converges, i.e. $v \to v_d$, we decide on the following cascaded controller as our input

$$\dddot{v} = u \tag{5.30a}$$
$$\tau_1 = k_1(v_d - v) \tag{5.30b}$$
$$\tau_2 = k_2(\tau_1 - \dot{v}) \tag{5.30c}$$
$$u = k_3(\tau_2 - \ddot{v}) \tag{5.30d}$$
$$\tag{5.30e}$$

By inspection we can find the values for the gains $k_1, 2, 3$ that guarantee that the right hand side of eq. (5.29) equals the left hand side are:

$$k_3 = (2\zeta + 1)\omega_0 \tag{5.31a}$$
$$k_2 = \frac{(2\zeta + 1)\omega_0^2}{k_3} \tag{5.31b}$$
$$k_1 = \frac{\omega_0^3}{k_3 k_2} \tag{5.31c}$$

The method so far solves the problem of generating a smooth acceleration reference for the multirotor, but we have no guarantee that constraints are not violated. We take advantage of our cascaded controller to introduce saturation at each step:

$$\tau_1 = sat(k_1(v_d - v),\ \dot{v}_{max}) \tag{5.32a}$$
$$\tau_2 = sat(k_2(\tau_1 - \dot{v}),\ \ddot{v}_{max}) \tag{5.32b}$$
$$u = k_3(\tau_2 - \ddot{v}) \tag{5.32c}$$

The reference model was simulated with the parameters from table 5.4, both with and without saturation. As can be seen in fig. 5.9 the trajectory is clearly smoothed, and with the saturation both $\dot{v}$ and $\ddot{v}$ stay inside the imposed limits.

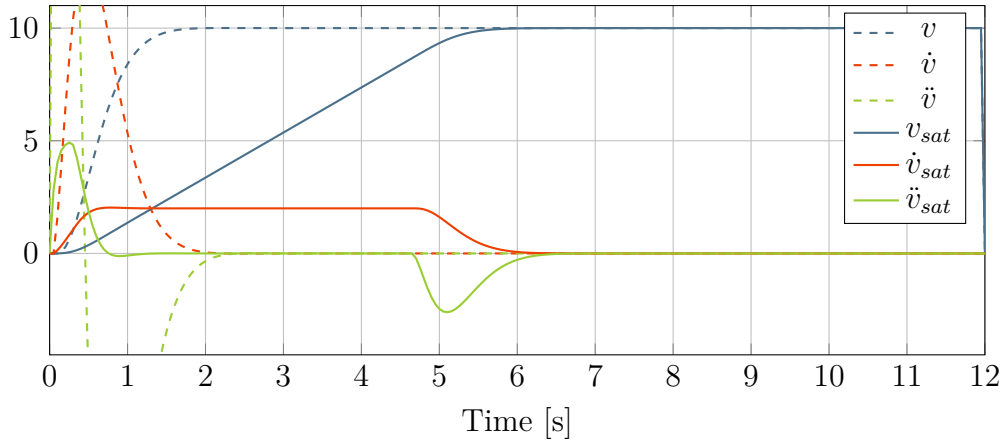


Figure 5.9: Reference model with and without saturation, simulated with parameters from table 5.4.

In order to be usable for a multirotor, one reference model is run for each cardinal direction. In addition the acceleration from the reference model is transformed into a reference acceleration a_d using the following equation.

$$a_d = a_{ref} - k_p(v_{ref} - v_{current}) \tag{5.33}$$

Here k_p is the gain for a controller that removes steady state error if the multirotor is unable to keep up with the acceleration reference. $v_{current}$ is the current velocity of the multirotor.

5.4.2 Control Hierarchy

At the bottom of the hierarchy is a simulated quadrotor delivered by ArduPilot as a part of its software suite. This software appears to DUNE as a quadrotor running ArduPilot, providing new position estimates at a frequency of 100 Hz. The simulated environment also simulates wind in NNE-direction at a speed of $1\,\mathrm{m\,s^{-1}}$. The simulator receives a desired acceleration in XYZ-direction in the body frame from DUNE.

The simulator outputs the position of the quadrotor, which is sent to an Real Time Kinematic Global Positioning System (RTKGPS) simulator, whose only job is to transform the position data into something usable by the control system. The NSB controller receives both a velocity reference input from an external source, as well as the state of the system. From the NSB controller we get a final velocity reference v_{d_out} which is sent to the reference model that outputs a reference acceleration for the simulator. The final control hierarchy can be seen in fig. 5.10.

5.4.3 Test Results

Three different scenarios were tested, all concerning avoiding the infinitely tall vertical cylinder. The two first scenarios test the difference between an equality-based and a

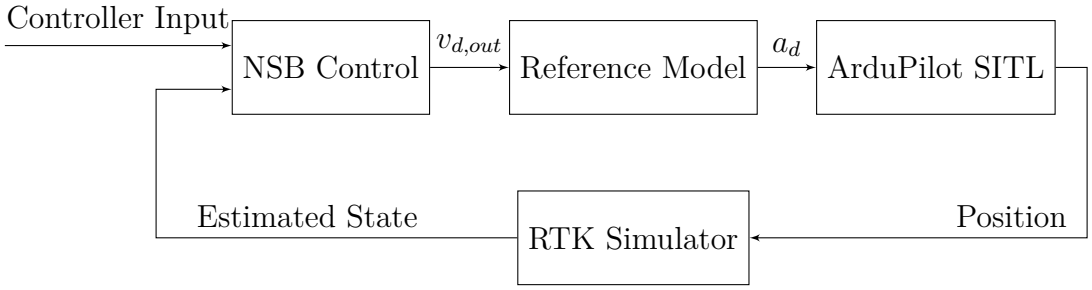


Figure 5.10: The control hierarchy as used in the simulation.

Variable	Value
Start position [NED]	$[0, 0, -50]^T$
Velocity input [NED]	$[0, 0, 0]^T$

(a) Various initial values

Task	λ	Coordinates	σ_{min}	σ_{max}	Priority
Vertical Cylinder	1	$[5, 4.9, x]^T$	4 m	∞	0

(b) Set-based tasks

Task	λ	Coordinates	$\sigma_{desired}$	Priority	$\|v_{i,max}\|_2$
GoTo	1	$[10, 10, -50]^T$	0	1	1

(c) Equality-based tasks

Table 5.5: The setup variables for the SITL test of a set-based cylinder placed between the multirotor and the GoTo-coordinate.

set-based cylinder task, while the last scenario tries to avoid an equality-based cylinder at high speed. The reference model is set to the parameters from table 5.4.

Set-based Cylinder Task

A GoTo-task as well as a Vertical Cylinder-task, similar to what was simulated earlier with the point mass simulator, was also simulated on the multirotor simulator. This scenario was run with the parameters in table 5.5, and the results are shown in fig. 5.11. The parameters were tuned to reduce the oscillatory behavior as much as possible, but no good solution was found. It is caused by the set-based task only being active when the system is in violation of the task (see eq. (5.25)). Because the system has inertia, in both the reference model and the Software in the Loop (SITL)-system, it gets thrown out by the task, but as the task is inactive it is allowed to accelerate towards the task boundary again. In order to dampen the oscillatory behavior the craft can move very slowly, but it is not a good solution. Instead it could possibly be solved by introducing a form of thresholding on when to deactivate the task, similar to what was done for the equality-based cylinder. This was not done in favor of implementing an equality-based cylinder task, which has proven to be working well under earlier circumstances such as [Klausen 2013].

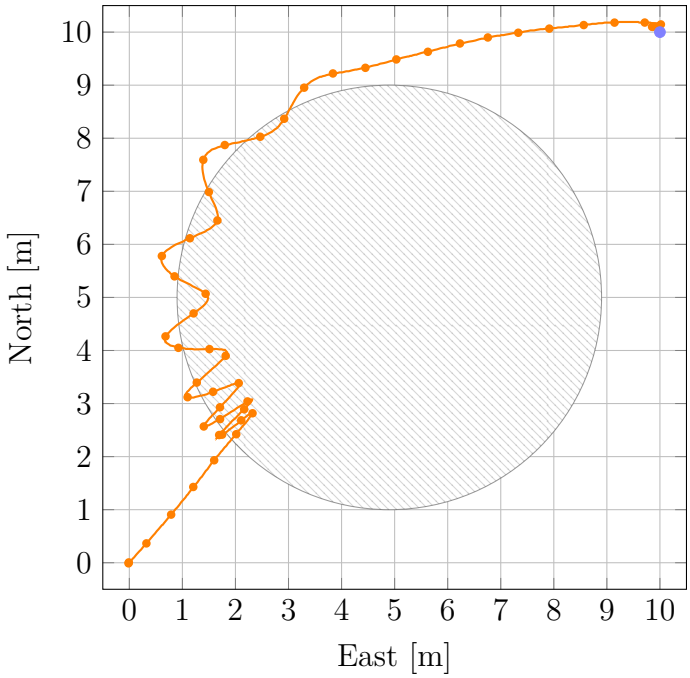


Figure 5.11: SITL test of the system with a set-based cylinder task. The test was ran with parameters from table 5.5. The hatched area is the location of the obstacle and the blue dot is the GoTo-coordinate. Along the trajectory a marker has been placed every second as an indication of speed.

Equality-based Cylinder Task

The scenario here is based upon the previous scenarios, except that the set-based cylinder task has been replaced with an equality-based one. The parameters had to be changed somewhat in order to work better with an equality-based task and can be found in table 5.6.

As can be discerned from fig. 5.12, the results are much better than what we saw in fig. 5.11 with a set-based task. The most noteworthy aspect is the increasing separation between the cylinder and the multirotor. This seems to be caused a steady-state offset caused by the small λ of the cylinder task and the simulated wind in the simulator in combination with the reference model being a bit slow. Increasing λ in order to reduce the separation caused the initial impact with the cylinder to be more oscillatory. In the end it was concluded that the separation is not likely to cause a big problem, as the wind situation will be very different with an actual obstacle and that the desired distance σ_d could be increased should the wind cause a collision with an obstacle.

High-speed Obstacle Avoidance

This scenario uses an equality-based task as an obstacle placed far away from the quadcopter, with a GoTo-coordinate placed even further away. The aim is to accelerate the multicopter to a high velocity on collision course with a cylinder. The parameters for the test can be found in table 5.7.

Variable	Value
Start position [NED]	$[0, 0, -50]^T$
Velocity input [NED]	$[0, 0, 0]^T$

(a) Various initial values

Task	λ	Coordinates	$\sigma_{desired}$	Priority	$\|v_{i,max}\|_2$	$\ddot{\sigma}_{max}$
Vertical Cylinder	1	$[5, 4.9, 0]$	4	0	∞	0.5
GoTo	1	$[10, 10, -50]^T$	0	1	1	∞

(b) Equality-based tasks

Table 5.6: The setup variables for the SITL test of a equality-based cylinder placed between the multirotor and the GoTo-coordinate.

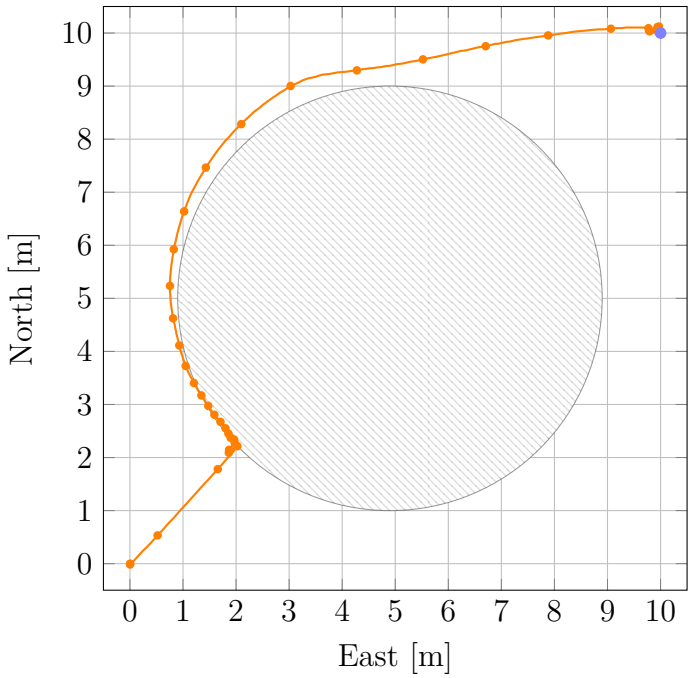


Figure 5.12: SITL test of the system with an equality-based cylinder task. The test was ran with parameters from table 5.6. The hatched area is the location of the obstacle and the blue dot is the GoTo-coordinate. Along the trajectory a marker has been placed every second as an indication of speed.

Variable	Value
Start position [NED]	$[0, 0, -50]^T$
Velocity input [NED]	$[0, 0, 0]^T$

(a) Various initial values

Task	λ	Coordinates	$\sigma_{desired}$	Priority	$\|v_{i,max}\|_2$	$\ddot{\sigma}_{max}$
Vertical Cylinder	1	$[50, 49.9, 0]$	4	0	∞	0.5
GoTo	0.5	$[100, 100, -50]^T$	0	1	∞	∞

(b) Equality-based tasks

Table 5.7: The setup variables for the SITL test of trying to collide with an equality-based cylinder placed between the multirotor and the GoTo-coordinate at high speed.

The end result, as can be seen in fig. 5.13, is interesting. Here the task activates very early in order to break the multirotor, which is caused by the dynamic activation distance discussed in section 5.1.3. The multicopter breaks a bit too much, as the task deactivates for a tiny bit, as seen around the coordinate $[35, 40]$. This might be removed by lowering $\ddot{\sigma}_{max}$, but no attempt was made at this as the result as it stands works as a proof of concept for the system. The part of the GoTo task's v_d which is not removed by the cylinder task is enough to move the multirotor away from the cylinder before impact.

An effect which could barely be seen in fig. 5.12, but is much more visible here, is the sudden acceleration away from the cylinder as the task deactivates. While the cause behind this has not been discovered, this could possibly be reduced by having the task deactivate in a more gentle way than a direct cutoff. Investigation into this is left for future works.

5.5 Conclusions

As we have seen so far, NSB is suitable for collision avoidance purposes. Both set-based NSB as described in [Moe et al. 2016] and new approach to equality-based NSB was implemented. The algorithm was implemented in C++ for DUNE, and was shown to work with a simple point mass simulator. During testing on a quadrotor simulator set-based NSB proved to be unreliable for an inertial system due to its activation conditions. However the equality-based NSB tasks worked both under high and low velocities because of the dynamic activation range. This shows that NSB is a feasible approach to collision avoidance for multirotors.

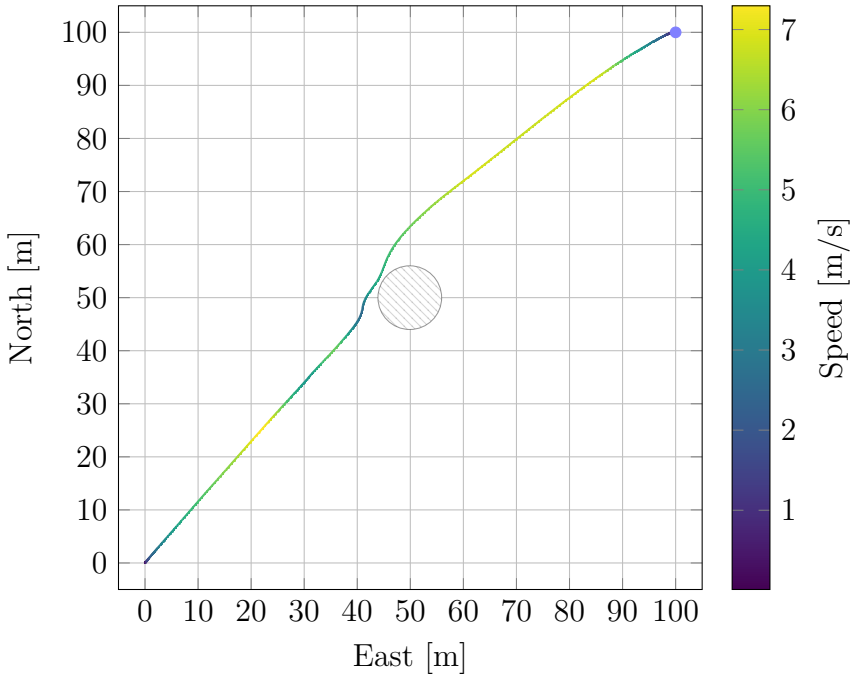


Figure 5.13: SITL test of the system, trying to collide with an equality-based cylinder task at high speed. The cylinder is shown as the hatched area. The speed of the quadrotor is indicated by the color of the trajectory. The GoTo-coordinate is a blue marker placed at $[100, 100]$. The simulations use the parameters from table 5.7.

6 | Conclusion and Closing Discussions

The goal for this thesis was to develop a radar-based collision avoidance system for multirotors. This goal was separated into building a radar system, detecting obstacles in the radar system and an obstacle avoidance system. Each part of the system was tested and verified to work on its own.

First an embedded radar system was developed. A small form factor commercially available radar system was mounted on a rotating base. The data from the radar was then coupled with a measurement of the pointing direction of the radar before being transmitted as serial data. Additional software libraries were developed to parse the serial data, as well as creating a live visualization of data. Finally a test rig for the system was developed, consisting of a base and several targets, and a data set that can be used for testing obstacle detection was created. Due to the added complexities of using a rotating base, it is recommended for future systems to instead use several radar systems to cover a 360° perimeter.

Further on it is shown that Synthetic Aperture Radar is not suitable for anti-collision purposes. Instead a novel approach to object detection was suggested, based on finding connected blobs in the data. This approach is shown to work sufficiently well for a stationary system using data gathered on the test rig. It is concluded that most obstacles can be represented as vertical cylinders.

Last we develop an implementation of Null-Space-based behavioral control (NSB) which is to be used for collision avoidance. Two different subsets of this algorithm are explored, set-based and equality-based NSB. The equality-based NSB also contains additional rules for when to activate a task. The developed software framework is made to be extendable to several different kinds of obstacles, and is verified to work through testing on a point mass simulator. The software is then tested on a quadrotor simulator, where it is shown that set-based NSB is not suitable for inertial systems without additional modifications. Equality-based NSB is shown to work well under both high and low approach velocities.

The three different systems were unfortunately only tested independently and not as a full collision avoidance system, so no conclusion can be drawn on how the system works as a whole. However we can conclude that each system works good on its own. It is imaginable that data from the radar can be fused with either visual data or lidar data in order to create a better map of obstacles than can be used by the collision avoidance software framework.

6.1 Future Work

Substituting the rotating base with several radar systems, and developing software for controlling them, is necessary to reduce the complexities of the system. If this happens it is necessary to develop a new algorithm for object detection.

It could also be a good idea to try to integrate all the components and verify if they work together. Especially test how the blob detection algorithm works with a moving base. The blob detection algorithm also needs to be implemented in DUNE for this to happen.

As for NSB it would be interesting to implement cubic obstacles, as it would allow for navigating in a probabilistic 3D grid map, where the probability is the chance of there being an obstacle in the grid element.

A | User Manual

This appendix is both a user manual as well as a documentation of what is delivered as a digital appendices. The purpose of this documentation is to make it easier for future work to pick up where I left.

Contents

Pin No.	Function	Direction
1	Battery Voltage	In
2	Battery Voltage	In
3	Ground	N/A
4	Ground	N/A

Table A.1: Pinout for the 2x2 MicroFit connector used for providing battery voltage

Pin No.	Function	Direction
1	Battery Voltage	In
2	Ground	N/A

Table A.2: Pinout for the 2x1 MicroFit connector used for providing regulated 5 V.

A.1 Hardware

This section explains the part of the hardware that was not covered in chapter 3.

A.1.1 Connectors

There are four connectors on the embedded radar system; two power connectors, one UART connector and one debug connector. The pin numbers are viewed from the front, counting clockwise from the top left corner. The pinout of the power connectors can be found in table A.1 and table A.2.

The pinout of the UART connector can be found in table A.3 and the debug connector in table A.4.

A.1.2 LEDs

Status LEDs are included on the embedded radar platform. They are to be interpreted as follows:

- **Green on MINI-M4** System has power

Pin No.	Function	Direction
1	N/C	N/A
2	UART TX	Out
3	UART RX	In
4	N/C	N/A

Table A.3: Pinout for the 1x4 MicroFit connector used for serial (UART) data.

Pin No.	Function	Direction
1	N_RST	In
2	SWD	In/Out
3	SWC	In

Table A.4: Pinout for the 2.54 mm pin header used for debugging the system

Name	Length [bytes]	Purpose
Start byte	1	Signifies start of frame. Value is `0x7D`.
Length	4	Remaining bytes of frame
Data	"Length"−5	Arbitrary data
CRC	4	Used to verify data integrity.
End byte	1	Signified end of frame. Value is `0x7F`.

Table A.5: The general frame structure used for communicating with the MCU.

- **Red on MINI-M4** Blinks as a set rate as long as the system is transmitting data. During boot is stays lit.
- **Orange on MINI-M4** Stays lit during boot.
- **Green** Is lit when the phototransistor is active.
- **Orange** Stays lit while transferring radar data out.
- **Red** Is blinked when a radar frame has been received from the *X2M200*

A.1.3 Communication

The UART connection with the embedded radar system is at 115200 bits per second, configured as 8N1, i.e. 8 databits, no parity and one stopbit.

Frame Format

The table from table 3.3 is reproduced in table A.5. The contents of the data field and the how to calculate the CRC is explained in the next sections.

Measurement Packet Format

Currently there is only one packet format supported by the data field of table A.5. All fields are a multiple of 4 bytes long and sent in little endian format. The format is specified in table A.6.

The array containing the start distance of bin i can be calculated:

$$\text{Distance}[i] = \text{Range offset} + \text{Bin length} \times i \tag{A.1}$$

Name	Length	Data type	Purpose
Bins	4	uint32_t	Number of bins
Bin length	4	float	Range length of each bin
Sampling frequency	4	uint32_t	Sampling frequency of the *X2*
Carrier frequency	4	float	Carrier frequency of the *X2*
Range offset	4	float	Range offset before first bin
Angle	4	float	Angle of system when the *Mini-M4* received the measurement
Angle Sync	4	uint32_t	Not 0 if the phototransistor has been active since last measurement. Used for debugging.
System time	4	uint32_t	System time in $\frac{1}{10000}$ of a second since startup.
Amplitude	4 * Bins	float[ ]	Amplitude measurements, one for each bin starting at closest range.
Phase	4 * Bins	float[ ]	Phase measurements, one for each bin starting at the closest range.

Table A.6: The measurement packet format that is contained in the frames received from the embedded radar system.

CRC

The CRC used to verify the frame data is specified by IEEE 8802-3, and is the same 32-bit CRC used for Ethernet. It uses the polynomial `0x04C11DB7`.

The following algorithm in C can be used to calculate the CRC. The CRC is only calculated on the contents of the data-field in the frame.

```
uint32_t crc32( const uint32_t *data, const uint32_t len ) {
  static const uint32_t polynomial = 0x04C11DB7;
  uint32_t crc = 0xFFFFFFFF;
  uint32_t i,j;
  for ( i = 0; i < len; ++i ) {
    crc ^= data[i];
    for ( j = 0; j < 32; ++j ) {
      crc <<= 1;
      if ( crc & 0x80000000 ) {
        crc ^= polynomial;
      }
    }
  }
  return crc;
}
```

A.1.4 Programming

The embedded radar system can be programmed with an SWD debugger that supports *STM32* using the debug header detailed in table A.4. It is important to make sure that the debugger and the embedded radar system share the same ground before commencing programming.

A.1.5 PCB Files

The PCB was developed with *Altium Designer 17*. The source files are available in the folder `Hardware/PCB/Altium` of the digital appendix. Manufacturing files are available in the folder `Hardware/PCB/Gerber` and a PDF-documentation is available at `Hardware/PCB/PCB.pdf`.

A.1.6 Radar Mounting Bracket

A .step-file of the radar mounting bracket is available at:

`Hardware/Bracket/bracket.step`.

A PDF blueprint is available at `Hardware/Bracked/blueprint.pdf`.

A.1.7 Developed Software

The embedded software is included as a digital appendix. The only requirement in order to compile is a standard UNIX environment and a recent distribution of the GNU ARM Embedded Toolchain. The embedded software is located in the digital appendix at:

`Software/Embedded`

A.2 DUNE Software

The full source code for a DUNE distribution is included as a digital appendix. There are three DUNE tasks that were developed for this thesis; a serial communication task, an NSB implementation, and a reference model for generating a desired force from a desired velocity. A tutorial on how to setup DUNE is available on `www.github.com/lsts/dune/wiki`. Some sample configuration files are available in the digital appendix at:

`Software/dune/user/etc/dev/`

Parameter	Default Value	Purpose
Serial Port - Device	`/dev/ttyUSB0`	Serial port descriptor
Serial Port - Baud Rate	`460800`	Serial port bit rate

Table A.7: Serial Communication Task parameters

A.2.1 Serial Communication Task

The serial communication task is located in the digital appendix at:

`Software/dune/user/src/Sensors/XeThruRadar/Task.cpp`

It accepts the following options that are written in table A.7. Each new measurement is broadcast as an IMC message of type `XeThruRadar`.

A.2.2 NSB Control Task

The NSB Control Task is located in the digital appendix at:

`Software/dune/user/src/Control/UAV/CollAvoid/Task.cpp`

It accepts the parameters that are specified in table A.8. The task inherits from `BasicUAVPilot` and activates when path control loop is required and it requires the speed control loop to start up. It accepts two kinds of IMC messages, `PWM` from an external radio controller, and `EstimatedState` from the craft. The final velocity is sent as an `DesiredLinearState` message.

A.2.3 Reference Model Task

The Reference Model Task can be found in the digical appendix at:

`Software/dune/user/src/Control/UAV/Velocity/Task.cpp`

It accepts the parameters specified in table A.9. This task also inherits from `BasicUAVPilot` and activates when the speed control loop is required. It accepts `DesiredLinearState` and `EstimatedState` IMC-messages. The desired force is sent as a `DesiredControl` message.

A.3 Python Software

All the Python software was written for Python version 3.6.

A.3.1 Serial Communication Module

There is a serial communication module located at:

Parameter	Default Value	Purpose
Latitude	63.45515289	Latitude of the NED frame
Longitude	10.91983723	Longitude of the NED frame
Height	0	Height of the NED frame
External Controller	False	If using external radio controller joystick as velocity reference.
Max Speed - XY	3	Used to transform controller input to speed.
Max Speed - Z	5	Used to transform the controller input to speed.
Max Acceleration	1	Used for dynamic activation range of Equality cylinder task
Input Speed	0,0,0	Input speed in NED frame to replace controller input.
Cylinder Position	10,0	Cylinder coordinates in the NED frame.
Cylinder Radius	2	Radius of the cylinder
Cylinder Clearance	0	Minimum clearance to the cylinder
Cylinder Active	False	If the cylinder task is active or not
Cylinder Priority	0	Priority of the task
Cylinder Lambda	1	Task gain λ
Cylinder Type	Equality	Equality or Set based cylinder.
Ground Clearance	0.5	Ground clearance for ground avoidance task.
Ground Active	False	If the task is active or not
Ground Priority	1	Priority of ground clearance task.
Ground Lambda	1	Task gain λ
GoTo Position	20,1,-5	Coordinate in NED frame to go to.
GoTo Active	False	If the task is active or not
GoTo Priority	2	Priority of the task
GoTo Lambda	1	Task gain λ
GoTo Saturation	0	Saturation on the desired velocity of the task.

Table A.8: NSB Control task parameters.

Parameter	Default Value	Purpose
Ref - Max Acc	6	Acceleration saturation
Ref - Max Jerk	8	Jerk saturation
Ref - Natural Frequency	1	Natural frequency ω_0 of the reference model.
Ref - Relative Damping	0.9	Relative damping ζ of the reference model.
K_p	1	Gain in equation eq. (5.32)
Multirotor - Mass	1	Mass of the multirotor, used to transform acceleration to force.
Linear State Producer	None	Task which to receive `DesiredLinearState` messages from.

Table A.9: Reference Model Task parameters

`Software/python/serial/connection.py`

And it contains the class Connection. The class is instantiated with only a serial port descriptor and the baud rate. Each new frame of data is received with the `get_data`-method, which returns one packet as a dictionary. This serial communication module does not verify the CRC of the received frames.

A.3.2 Visualization Software

The 3D visualization of live radar data is located at:

`Software/python/visualizer/visualize.py`.

Help for the program can be printed by running it with the `-h` switch.

A.3.3 Data logger

A data logger for logging data from the radar system is available at:

`Software/python/logger/store_data.py`

Help for the program can be printed by running it with the `-h` switch.

A.3.4 Blob detection

Blob detection software developed in chapter 4. As an input it requires .mat-files generated by either MATLAB or the data logger, and it outputs the blobs as .csv tabulated file. The program can be found at:

`Software/python/BlobDetect/store_blobs.py`

Help for the program can be printed by running it with the `-h` switch.

B | Test plan

A rigorous test plan for the embedded radar system was made, containing all different situations to be tested as well as a naming system for measurements and resulting data files as shown in table B.1. Each measurements was named in the pattern "series-no", such that the measurement at row 2 in table B.1 was named "A-1". Each measurement consisted of 20 rotations of the embedded radar system, and each measurement was repeated 3 times. This was done to check the repeatability of the measurements, and give a better data set for object recognition. Measurements without any targets were also done in order to check the noise floor and check if there was any clutter shading the radar. These checks were performed at a regular interval.

A few error sources that appeared under testing is a slight offset in the orientation of the radar, such that all targets appear with a small offset in the test data. There are also some slight inaccuracies when measuring the distance to the target, as the distance had to be measured in air and might have been measured at an angle.

The test data is available as a digital appendix in the folder `Test` together with MATLAB scripts for plotting, exporting and sanitizing the data. An analysis of the data can be found in [Koren 2016], which is available in the digital appendix at the root directory with the file name `project_report.pdf`

Type	Series	Test	Object 1		Object 2	
			Type	Pos[$cm\angle°$]	Type	Pos[$cm\angle°$]
Noise floor	0	0				
Material/Size	A	1	Cylndr Ø31	$92\angle 0$		
		2	Cylndr Ø25a	$92\angle 0$		
		3	Cylndr Ø25b	$92\angle 0$		
		4	Cylndr Ø10.5	$92\angle 0$		
		5	Cylndr Ø5.5	$92\angle 0$		
		6	PVC Ø11	$92\angle 0$		
		7	Cardbrd Ø11.5	$92\angle 0$		
Angular differentiation	B	1	Cylndr Ø31	$90\angle 90$	Cylndr Ø31	$90\angle -90$
		2	Cylndr Ø31	$90\angle 60$	Cylndr Ø31	$90\angle -60$
		3	Cylndr Ø31	$90\angle 30$	Cylndr Ø31	$90\angle -30$
		4	Cylndr Ø31	$90\angle 15$	Cylndr Ø31	$90\angle -15$
Distance	C	1	Cylndr Ø25a	$151\angle 0$		
		2	Cylndr Ø25a	$156\angle 0$		
		3	Cylndr Ø25a	$125\angle 0$		
		4	Cylndr Ø25a	$96\angle 0$		
		5	Cylndr Ø25a	$66\angle 0$		
		6	Cylndr Ø25a	$36\angle 0$		
Shape Corner	D	1	Corner 0°	$108\angle 0$		
		2	Corner 45°	$108\angle 0$		
		3	Corner 90°	$108\angle 0$		
		4	Corner 135°	$108\angle 0$		
		5	Corner 180°	$108\angle 0$		
		6	Corner 225°	$108\angle 0$		
		7	Corner 270°	$108\angle 0$		
		8	Corner 315°	$108\angle 0$		
Shape Wall	E	1	Wall	$226\angle 90$		
		2	Wall	$200\angle 90$		
		3	Wall	$174\angle 90$		
		4	Wall	$148\angle 90$		
		5	Wall	$122\angle 90$		
		6	Wall	$96\angle 90$		
		7	Wall	$70\angle 90$		
		8	Wall	$44\angle 90$		
Shading	F	1	Cylndr Ø31	$130\angle 0$	Cylndr Ø31	$55\angle 0$
		2	Cylndr Ø31	$130\angle 0$	PVC Ø11	$55\angle 0$
		3	Cylndr Ø31	$130\angle 0$	Cylndr Ø5.5	$55\angle 0$
		4	Cylndr Ø31	$130\angle 0$	Cylndr Ø5.5	$55\angle -10$
Fun Human	G	1	Thomas	$90\angle 0$		
Fun Chairs	H	1	Stack of chairs	$100\angle 0$		

Table B.1: The range of tests performed on the radar system in order to evaluate its capabilities.

Acronyms

BLDC Brushless Direct Current. 21–25, 32

CRC Cyclic Redundancy Check. 21, 28, 75, 76, 80

CW Continuous Wave. 6, 11

DMA Direct Memory Access. 28

DUNE DUNE: Unified Navigation Environment. 18, 29, 57, 64, 68, 72, 77

EoF End of Frame. 20

ESD ElectroStatic Discharge. 25

FMCW Frequency Modulated CW. 11

FOV Field of View. 14

GLUED GNU/Linux Uniform Environment Distribution. 18

IMC Intermodule Communication. 29, 78

IR Infrared. 15, 25

LDO Low DropOut. 24

LO Local Oscillator. 6

LSTS Laboratório de Sistemas e Tecnologia Subaquática. 18

MCU Microcontroller Unit. 23–25, 28, 75

MEMS Micro-Electro-Mechanical System. 14

MOSFET Metal-Oxide-Semiconductor Field-Effect-Transistor. 24, 32

MUR Maximum Unambiguous Range. 12, 20

NED North-East-Down. 62

NSB Null-Space-based Behavioral. 46–49, 51–53, 57, 60, 62, 64, 68, 72, 77

OBC Onboard Computer. 18, 19

PCB Printed Circuit Board. 18, 23–25, 33

PRF Pulse Repetition Frequency. 6, 10, 12, 20

PVC Polyvinyl Chloride. 14

RGB-D Red Green Blue + Depth. 15

RTKGPS Real Time Kinematic Global Positioning System. 64

RTOS Real Time Operating System. 27

SAR Synthetic Aperture Radar. 13, 35–37, 43

SITL Software in the Loop. 65–69

SNR Signal to Noise Ratio. 6, 10

SoC System on a Chip. 10, 13

SoF Start of Frame. 20

SPI Serial Peripheral Bus. 25

SWD Serial Wire Debug. 24, 77

UART Universal Asynchronous Receiver/Transmitter. 20, 21, 74, 75

UAV Unmanned Aerial Vehicle. 1, 45

UUID Universally Unique Identifier. 56

UWB Ultra-Wide Band. 10, 13

XOR Exclusive OR. 21

References

Aerospace and Electronic Systems IEEE Society (2007). *IEEE Standard for Ultrawideband Radar Definitions IEEE Aerospace and Electronic Systems Society*. May.

Aerotenna (2016). *Aerotenna μSharp*. URL: http://aerotenna.com/sensors/#usharp (visited on 07/01/2017).

Ali, Faiza, Georg Bauer, and Martin Vossiek (2014). "A Rotating Synthetic Aperture Radar Imaging Concept for Robot Navigation". In: *IEEE Transactions on Microwave Theory and Techniques* 62.7, pp. 1545–1553. DOI: 10.1109/TMTT.2014.2323013.

Antonelli, Gianluca (2008). "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems". In: *Proceedings - IEEE International Conference on Robotics and Automation* 25.5, pp. 1993–1998. ISSN: 10504729. DOI: 10.1109/ROBOT.2008.4543499.

Antonelli, Gianluca, Filippo Arrichiello, and Stefano Chiaverini (2005). "The null-space-based behavioral control for mobile robots". In: *2005 IEEE International Symposium on Computation Intelligence in Robotics and Automation*, pp. 15–20. DOI: 10.1109/CIRA.2005.1554248.

– (2008a). "Stability analysis for the Null-Space-based Behavioral control for multi-robot systems". In: *Proceedings of the IEEE Conference on Decision and Control* 2, pp. 2463–2468. ISSN: 01912216. DOI: 10.1109/CDC.2008.4738697.

– (2008b). "The null-space-based behavioral control for autonomous robotic systems". In: *Intelligent Service Robotics* 1.1, pp. 27–39. ISSN: 18612776. DOI: 10.1007/s11370-007-0002-3.

Bareiss, Daman, Joseph R Bourne, and Kam K Leang (2017). "On-board model-based automatic collision avoidance : application in remotely-piloted unmanned aerial vehicles". In: *Autonomous Robots*. ISSN: 1573-7527. DOI: 10.1007/s10514-017-9614-4.

Brahmagupta (0628). *Brahmasphutasiddhanta*. Bhinmal.

Brooks, Rodney A. (1986). "A Robust Layered Control System For A Mobile Robot". In: *IEEE Journal on Robotics and Automation* 2.1, pp. 14–23. ISSN: 08824967. DOI: 10.1109/JRA.1986.1087032. arXiv: 1010.0034.

Deris, A. et al. (2017). "DEPTH CAMERAS ON UAVs: A FIRST APPROACH". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W3.March, pp. 231–236. ISSN: 2194-9034. DOI: 10.5194/isprs-archives-XLII-2-W3-231-2017.

Echodyne (2016). *Echodyne Products*. URL: http://echodyne.com/products/ (visited on 07/01/2017).

Engels, Florian et al. (2017). "Advances in Automotive Radar". In: *IEEE Signal Processing Magazine* March. DOI: 10.1109/MSP.2016.2637700.

Fontana, Robert J. (2004). "Recent system applications of short-pulse ultra-wideband (UWB) technology". In: *IEEE Transactions on Microwave Theory and Techniques*. Vol. 52. 9 I, pp. 2087–2104. ISBN: 0018-9480. DOI: 10.1109/TMTT.2004.834186.

Gageik, Nils, Paul Benz, and Sergio Montenegro (2015). "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors". In: *IEEE Access* 3, pp. 599–609. ISSN: 21693536. DOI: 10.1109/ACCESS.2015.2432455.

Kinzie, Nicola Jean (2011). "Ultra Wideband Pulse Doppler Radar for Short-Range Targets". PhD Thesis. University of Colorado.

Klausen, Kristian (2013). "Cooperative Behavioural Control for Omni-Wheeled Robots". Master Thesis. NTNU.

Klausen, Kristian, Thor I. Fossen, and Tor Arne Johansen (2015). "Nonlinear control of a multirotor UAV with suspended load". In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 176–184. DOI: 10.1109/ICUAS.2015.7152289. URL: http://ieeexplore.ieee.org/document/7152289/.

Koren, Thomas Frimann (2016). *Radar-based Multirotor Collision Avoidance System*. Tech. rep. Trondheim: NTNU, p. 71.

Maître, Henri (2008). *Processing of synthetic aperture radar images*. Vol. 129. London: Wiley, p. 382. ISBN: 9781848210240. DOI: 10.1002/9780470611111.

MikroElektronika. *MINI-M4 Datasheet*. URL: http://www.mikroe.com/downloads/get/1946/mini-m4-stm32-manual-v100.pdf.

Moe, Signe et al. (2015). "Stability Analysis for Set-based Control within the Singularity-robust Multiple Task-priority Inverse Kinematics Framework". In: *2015 IEEE 54th Annual Conference on Decision and Control* 54. DOI: 10.1109/CDC.2015.7402104.

Moe, Signe et al. (2016). "Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results". In: *Frontiers in Robotics and AI* 3.April, pp. 1–18. ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00016.

Moreira, Alberto et al. (2013). "A Tutorial on Synthetic Aperture Radar". In: *IEEE Geoscience and Remote Sensing Magazine* march, pp. 1–43. ISSN: 2168-6831. DOI: 10.1109/MGRS.2013.2248301.

Moses, Allistair A. et al. (2014). "UAV-borne X-band radar for collision avoidance". In: *Robotica* 32.1, pp. 97–114. ISSN: 0263-5747. DOI: 10.1017/S0263574713000659.

Nieuwenhuisen, Matthias and Sven Behnke (2015). "3D Planning and Trajectory Optimization for Real-time Generation of Smooth MAV Trajectories". In: *2015 European Conference on Mobile Robots*. DOI: 10.1109/ECMR.2015.7324217.

Nolan, Andrew et al. (2013). "Obstacle mapping module for quadrotors on outdoor Search and Rescue operations". In: *International Micro Air Vehicle Conference and Flight Competition* September.

Novelda AS (2015). *X2 Datasheet*. URL: https://www.xethru.com/community/resources/x2-datasheet.24/.

– (2016a). *X2M200 Datasheet*. Kviteseid. URL: https://www.xethru.com/community/resources/x2m200-respiration-module-preliminary-datasheet.9/.

– (2016b). *XeThru Serial Protocol*. Kviteseid. URL: https://www.xethru.com/community/resources/xethru-serial-protocol.14/.

Richardson, Doug (2003). *Stealth Warplanes*. Zenith Press, p. 192. ISBN: 978-0760310519.

Rosales, Claudio, Paulo Leica, and Mario Sarcinelli-filho Gustavo Scaglia (2016). "3D Formation Control of Autonomous Vehicles Based on Null-Space". In: *Journal of Intelligent & Robotic Systems*, pp. 453–467. ISSN: 0921-0296. DOI: 10.1007/s10846-015-0329-5.

Skolnik, Merill I. (2001). *Introduction to Radar Systems*. Third ed. New York: McGraw-Hill. ISBN: 978-0072881380.

Skolnik, Merril I. (2008). *Radar Handbook*. Third ed. New York: McGraw-Hill, p. 1318. ISBN: 9780071485470.

STMicroelectronics (1998). *AN1088 L6234 Three-phase Motor Driver*. URL: www.st.com/resource/en/application{_}note/cd00004062.pdf.

Stutzman, Warren L. and Gary A. Thiele (2013). *Antenna Theory and Design*. 3rd. Wiley, p. 843. ISBN: 9780470576649.

The Tesla Team (2016). *Upgrading Autopilot: Seeing the World in Radar*. URL: https://www.tesla.com/blog/upgrading-autopilot-seeing-world-radar (visited on 07/02/2017).

Velodyne LiDAR. *Velodyne Puck VLP-16*. URL: http://velodynelidar.com/vlp-16.html (visited on 07/01/2017).

– (2017). *Velodyne LiDAR Announces New "Velarray" LiDAR Sensor*. San Jose. URL: http://velodynelidar.com/docs/news/Velodyne%20LiDAR%20Announces%20New%20Velarray%20LiDAR%20Sensor%20_%20Business%20Wire.pdf.

Wang, Xiao et al. (2016). "On-Road Vehicle Detection and Tracking Using MMW Radar and Monovision Fusion". In: *IEEE Transactions on Intelligent Transportation Systems* 17.7, pp. 2075–2084. ISSN: 15249050. DOI: 10.1109/TITS.2016.2533542.

Weitkamp, Claus. *Lidar - Range-Resolved Optical Remote Sensing of the Atmosphere*. Geesthacht: Springer Science. ISBN: 0-387-40075-3.

Weixian, Liu et al. (2016). "Premier results of the multi-rotor based FMCW synthetic aperture radar system". In: *2016 IEEE Radar Conference, RadarConf 2016*, pp. 3–6. DOI: 10.1109/RADAR.2016.7485188.

XeThru Forum (2017). *Baseband data over UART*. URL: https://www.xethru.com/community/threads/baseband-data-over-uart.183/ (visited on 07/01/2017).

Yoo, Dong-wan, Dae-yeon Won, and Min-jea Tahk (2011). "Optical Flow Based Collision Avoidance of Multi-Rotor UAVs in Urban Environments". In: 12.3, pp. 252–259. DOI: 10.5139/IJASS.2011.12.3.252.

Yunqiang, Yang and Aly E. Fathy (2005). "See-through-wall imaging using Ultra Wideband short-pulse radar system". In: *IEEE Antennas and Propagation Society, AP-S International Symposium (Digest)* 3 B, pp. 334–337. ISSN: 15223965. DOI: 10.1109/APS.2005.1552508.

Image Sources

Department, Avionics (2013). *Electronic Warfare and Radar Systems*. Fourth edition. Point Mugu, CA: Naval Air Warfare Center Weapons Division.

Felix Niessen (2013). *Winding Scheme Calculator*. [Online; generated December 20, 2016]. URL: `http://www.bavaria-direct.co.za/scheme/calculator/`.

Wikipedia, the free enclyclopedia (2009). *Radar Cross Section of Metal Sphere from Mie Theory*. [Online; accessed December 16, 2016]. URL: `https://en.wikipedia.org/wiki/File:Radar_cross_section_of_metal_sphere_from_Mie_theory.svg`.

Wikipedia, the free encyclopedia (2007). *Sigma Invader RCS*. [Online; accessed December 16, 2016]. URL: `https://en.wikipedia.org/wiki/File:Sigma_invader_RCS.png`.