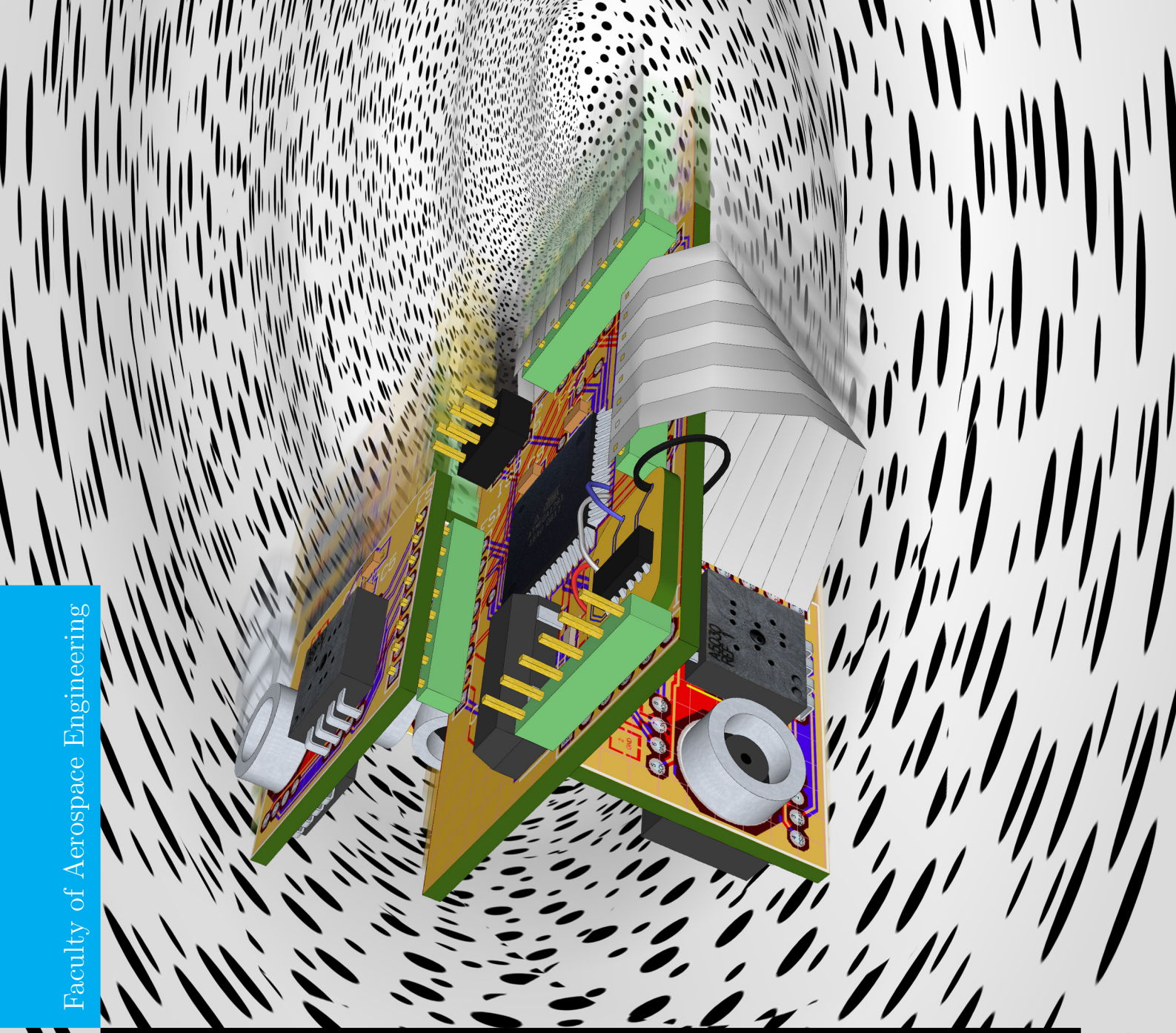


Faculty of Aerospace Engineering

Optical Flow Based State Estimation for an Indoor Micro Aerial Vehicle

M. J. Verveld

August 17, 2009

TU Delft Delft University of Technology

Challenge the future

Optical Flow Based State Estimation for an Indoor Micro Aerial Vehicle

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

M. J. Verveld

August 17, 2009

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Optical Flow Based State Estimation for an Indoor Micro Aerial Vehicle"** by **M. J. Verveld** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 17, 2009

Readers:

prof.dr.ir. J. A. Mulder

dr. Q. P. Chu

ir. C. de Wagter

ir. J. H. Breeman

Acknowledgements

The idea for this thesis work has developed during my internship at Georgia Tech where I experienced real world UAV engineering and the challenges which arise when translating ideas into hardware. Indoor autonomous UAV flight is the next big challenge in this field and a presentation by dr. R. W. Beard of Brigham Young University showed the use of optical flow (OF) sensors to estimate distances to obstacles in order to avoid them. After my return to Delft I decided to use those OF sensors for estimating the state of an indoor micro aerial vehicle (MAV).
I would like to thank prof.dr.ir. J. A. Mulder and dr. Q. P. Chu for accepting this idea as my MSc. thesis assignment.
I also want to thank dr. Q. P. Chu for his advise along the way, insightful discussions and suggestions for subjects which I had not yet considered. This has really deepened my insight and knowledge.
I want to thank Christophe de Wagter for his design of the sensor board and help with building and testing it. He has pointed out possible pitfalls and was there with advice when the idea for the thesis took shape. His extensive self-taught knowledge of electronics has inspired me and I have learnt a lot from him.
A big thank you to Wiebe, Armin, Menno Wierema and René Lagarde. They were always in for discussions and have shared their experience, which has helped me out a lot.
To complete the list, I would like to thank the Simona Graduate community: Gijsbert, Joep, Jan, Barend, Patrick, Sören, Ties, Matin, Marijke, Maarten, Daan, Arie, Laurens, Paul, Pieter, Rens, Jorg, Sjoerd, Menno, Mark van der Steen, Mark van den Hoven, Kakin and Mauro for their support and enlightening discussions. Especially the "off-topic" ones have made the graduation time very enjoyable.

Finally, I would like to dedicate this work to my parents for always supporting me during all these years in Delft.

Braunschweig, Deutsches Zentrum für Luft- und Raumfahrt M. J. Verveld
August 17, 2009

Nomenclature

Greek Symbols

α	field of view (fov) angle
Γ	$= \frac{\partial f}{\partial u}$, input mapping matrix
γ	Observation sigma point vector
$\delta\alpha$	Smallest measurable angular displacement
$\delta\Omega$	Smallest measurable OF increment
$\vec{\delta}$	Displacement vector from center of gravity (CoG)
ε	Controller error
Θ	Elevation angle, in spherical coordinates
θ	Pitch angle
$\underline{\mu}(\mathcal{O})$	Distance to rank deficiency of $\mathcal{O}$
σ_i	Singular values of a matrix
σ	Standard deviation, only appears without numeral subscript
Φ	Discrete state transition matrix
φ	Roll angle
χ	Sigma point vector of the unscented transform
Ψ	Azimuth angle, in spherical coordinates
ψ	Yaw angle
ω_n	Natural frequency
ω_s	Sample frequency
Ω_{X_i}	Optical flow component of sensor i in direction X
$\vec{\Omega}$	Optical flow vector field (three dimensional)
$\vec{\omega}$	Rotational rate vector
$\vec{\Omega}^*$	Optical flow vector field (two dimensional projection)

Latin Symbols

A	LTI system matrix
a	Acceleration (scalar)
A_x	Specific force component along the body X-axis
A_y	Specific force component along the body Y-axis
A_z	Specific force component along the body Z-axis
B	LTI system matrix
B	Image size
b	Lens image distance
$\mathcal{C}_O$	Observability condition number
C	LTI system matrix
C	Damper coefficient
D	LTI system matrix
d	Distance
d	circle of confusion (CoC) diameter
DCM	Direction Cosine Matrix
dt_s	Sampling timestep
$\vec{f}(\vec{x}, \vec{u})$	Dynamics equations
F	Jacobian of the dynamics
f	Lens focal length
f_s	Specific force
g	Gravitational acceleration (scalar)
$\vec{h}(\vec{x})$	Observation equations
H	Jacobian of the observations
I_x	Moment of inertia about the body X-axis
I_y	Moment of inertia about the body Y-axis
I_z	Moment of inertia about the body Z-axis
J_{xz}	$= \int xz\, dm$, product of inertia
K	Observer gain matrix, or Kalman gain matrix in the stochastic system case
K	Spring coefficient
L	The input space, generally $\subseteq \mathbb{R}^m$
L	Lift force along the body Z-axis
$L_f(g)$	$= \frac{\partial g}{\partial \vec{x}} \vec{f}$, Lie derivative of g with respect to $\vec{f}$
$\mathcal{M}_x$	Component of the moment about the body X-axis
$\mathcal{M}_y$	Component of the moment about the body Y-axis
$\mathcal{M}_z$	Component of the moment about the body Z-axis
M	The state space, generally $\subseteq \mathbb{R}^n$
m	Mass
$\mathbf{n}$	Viewing direction vector of unit length
n	Dimension of the state vector
$\mathcal{O}$	Observability matrix

P	Error covariance matrix
p	Component of $\vec{\omega}$ along the body X-axis
Q	Covariance matrix of $w(k)$
q	Component of $\vec{\omega}$ along the body Y-axis
$\mathbb{R}$	The set of all real numbers
R	Covariance matrix of $v(k)$
r	Component of $\vec{\omega}$ along the body Z-axis
S	Cross covariance matrix of $v(k)$ and $w(k)$
T	Thrust force along the body X-axis
t	Time
$\vec{u}$	System input vector with dimension m
u	Component of $\mathbf{V}$ along the body X-axis
$\mathbf{V}$	Velocity vector
v	Component of $\mathbf{V}$ along the body Y-axis
v	Lens subject distance
$v(k)$	Discrete measurement noise sequence
W	$= mg$, Weight
w	Component of $\mathbf{V}$ along the body Z-axis
$w(k)$	Discrete process noise sequence
$\bar{x}$	Mean state vector
$\hat{x}$	Estimated state vector
$\vec{x}$	System state vector with dimension n
X	Component of the total aerodynamic force along the body X-axis
$\vec{y}$	System output vector with dimension p (deterministic)
Y	Component of the total aerodynamic force along the body Y-axis
$\mathbb{Z}$	The set of all integers
$\vec{z}$	Observations vector (stochastic)
Z	Component of the total aerodynamic force along the body Z-axis

Subscripts

$_{body}$	Vector quantity is expressed in the body frame of reference
$_{e \rightarrow b}$	Denotes transformation from e for earth to b for body reference frame. May also include s for sensor frame
$_{earth}$	Vector quantity is expressed in the earth frame of reference
$_{sensor}$	Vector quantity is expressed in the sensor frame of reference

Superscripts

^{1}	Used to denote that the quantity is only the first term of a complete definition
^{T}	Transpose of a matrix

Acronyms

ADC analog-to-digital converter

ASTI Aerospace Software and Technologies Institute

BWB blended wing body

CNC computer numerical control

CoC circle of confusion

CoG center of gravity

cpi counts per inch

CS Chip Select

DC direct current

DCM direction cosine matrix

DLR dorsal light response

dof depth of field

DSP digital signal processor

EKF Extended Kalman filter

EMAV European Micro Aerial Vehicle Conference and Flight Competition

EMD elementary motion detector

EML Embedded Matlab

EoM equations of motion

fps frames per second

fov field of view

GPS Global Positioning System

HKF Hybrid Kalman Filter

IARC International Aerial Robotics Competition

I^{2}C Inter-Integrated Circuit Bus

IC Integrated Circuit

IEKF Iterated Extended Kalman filter

IMU inertial measurement unit

I/O Input/Output

ISP In-System Programming

LTI linear time invariant

MAV micro aerial vehicle

µC microcontroller

MEMS Micro-Electro-Mechanical Systems

MISO Master in Slave out

MMSE minimum mean squared error

MOSI Master out Slave in

MSD mean squared difference

MSE mean squared error

OF optical flow

OMR Opto-Motor Response

PWM Pulse Width Modulation

RISC Reduced Instruction Set Computer

RK4 4^{th} order Runge-Kutta

SCK Serial Clock

SLAM Simultaneous Localization And Mapping

SPI Serial Peripheral Interface Bus

UART Universal Asynchronous Receiver/Transmitter

UAV unmanned aerial vehicle

UKF Unscented Kalman filter

UT Unscented Transform

Contents

List of Figures

Chapter 1

Introduction

This introductory chapter discusses the motivation for the work of this thesis, it presents an approach based on literature research and it defines the goal of the thesis.

1-1 Background

Right from the beginning, aeronautical engineering has strived towards faster, larger or higher-reaching aircraft. Only recently has another research path been ventured upon. During the past decade, there has been an increasing research effort to develop aircraft that are as small as possible. At first, there was a military need for aircraft without a pilot onboard. These unmanned aerial vehicles (UAVs) have since proven themselves as highly successful in a growing range of missions. An increasing number of civilian applications are emerging as well. The common elements found in unmanned aerial vehicle (UAV) applications are Dull, Dangerous and Dirty (DDD). Although a typical military acronym, it is applicable to the civilian market as well.

As technological advances in electronics have allowed critical onboard systems to become ever smaller and lighter, a branch of UAVs has emerged which is designed for applications where small size is a driving requirement. The term adopted for these systems is micro aerial vehicle (MAV). One particular mission environment requiring small size, is the interior of buildings.

This thesis work is motivated by the specific challenges that this mission environment poses. Possible application areas include surveillance missions in hazardous situations such as fires or pollution accidents and intelligence gathering. In such situations sending in humans may not be possible due to hazards or risk of detection. A flying vehicle which can navigate the scene would give firefighters or special forces a valuable tool to assess the situation and to look for valuable information or the location of victims.

To encourage research and development, a number of design competitions and conferences are being held worldwide on a regular basis. Two important ones are the European Micro Aerial Vehicle Conference and Flight Competition (EMAV) and the International Aerial Robotics

Competition (IARC). Both include an indoor flight competition with an emphasis on precision navigation and autonomy. As an example of what is required for a practical application, the challenge posed by the IARC combines outdoor and indoor parts in a fully autonomous mission:

> *"The 5th [and current] Mission requires a fully autonomous aerial subvehicle - launched from a "mother ship" - to penetrate a building and negotiate the more complex interior space containing hallways, small rooms, obstacles, and dead ends in order to search for a designated target without the aid of global-positioning navigational aids, and relay pictures back to a monitoring station some distance from the building. — The 5th Mission will continue to adhere to the Competition's 18-year practice of posing tasks that cannot be completed with current technology and skills. As with previous missions, nothing within the World military or industrial arsenal of robots will be able to complete the proposed mission at the time the guidelines are released."*, (IARC, 2009).

(a) MIT's quadrotor (b) Georgia Tech's coaxial rotor (c) Embry-Riddle's monoblade

Figure 1-1: Examples of IARC 2009 aircraft designs

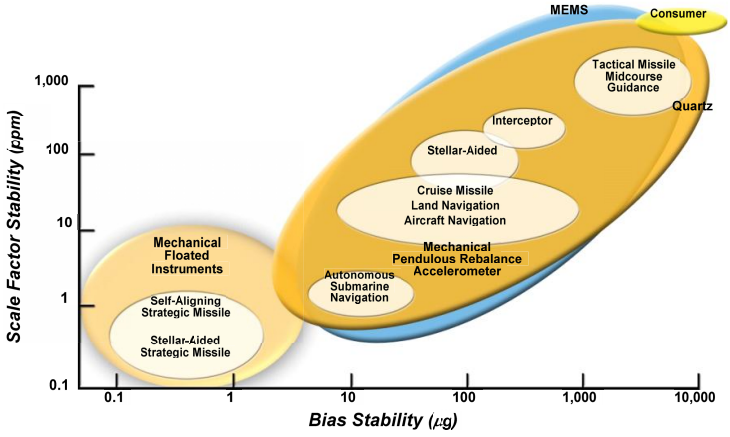
Most entries into this competition have approached the indoor aircraft problem with a quadrotor design, such as MIT's in Figure 1-1 a, while Georgia Tech has a coaxial rotorcraft, Figure 1-1 b, and Embry-Riddle Aeronautical University's solution is a more exotic monoblade design, Figure 1-1 c.

Another category which is not represented in the IARC, is the flapping wing or ornithopter design. An example of this approach is the Delfly from ASTI in Delft, Figure 1-2. This team is participating in the EMAV events.
Looking at these competitions and the designs which the participants have produced gives an idea of the direction in which future real-world applications may go. It is still an experimental and fast developing field and research and experience will have to show what the best solutions are and which applications MAVs will ultimately be most useful for.

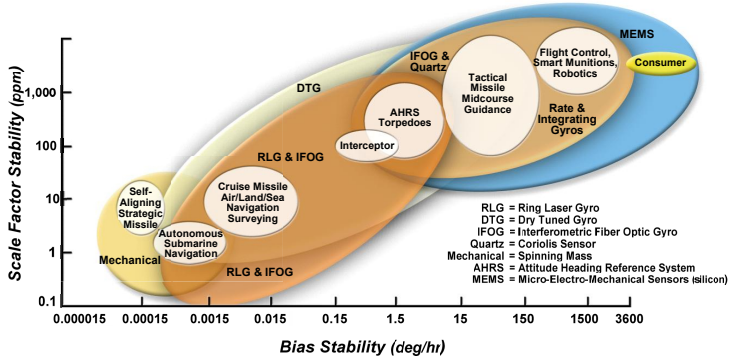
1-2 Problem Statement

The examples in the previous section show that there exist several aircraft design approaches which are capable of flying in the indoor environment. The next design problem to meet

Figure 1-2: ASTI's DelFly I



(a) Accelerometers



(b) Gyroscopes

Figure 1-3: Inertial sensor technologies and their applications, from (Schmidt, 2009)

mission requirements is to make a given platform autonomous. This means that it must be capable of stabilising itself in all degrees of freedom and on top of that have a way of navigating and exploring a building with unknown geometry.
The specific problem posed by building interiors with respect to aircraft control is that the

vehicle is enclosed in all directions. This limits not only the freedom of movement, but also the reception of positioning signals such as the Global Positioning System (GPS). The earth's magnetic field is distorted in many buildings as well. This limits the usefulness of magnetometers which are frequently used to control attitude. These factors mean that a conventional avionics suite consisting of inertial measurement unit (IMU), GPS and a 3-axis magnetometer will not work here.
The only component which is unaffected by the environment is the IMU. The accelerometers and gyroscopes making up the IMU of an MAV are generally Micro-Electro-Mechanical Systems (MEMS). Due to the size reduction and mass-producibility of these silicon-based sensors, they are the most viable if not the only option fitting within the extremely tight weight and size budget of MAVs. (Barbour, 2009) notes that the downside of MEMS is that, as size decreases, sensitivity decreases and noise increases. Also, the temperature dependency in the Young's modulus of silicon is ~ 100 ppm/°C, which significantly reduces scale factor stability. At present, the best attainable accuracy for all-MEMS IMUs is limited by the gyro performance, which is at around $10 - 30$ °/h bias stability, while accelerometer performance is at about 10 μg bias stability, see Figure 1-3. Note however, that this is the high-end performance limit and most commercially available MEMS devices are several orders of magnitude less accurate. This means performance is strongly dependent on budget and university research generally has to cope with far lower accuracies.
Integrating the signals from these sensors to velocities or even position and attitude angles directly would yield a quickly diverging solution, therefore MEMS IMUs will require either GPS to calibrate the solution using e.g. a Kalman filter, or "the integration of [other] augmentation sensors in GPS-denied environments", (Barbour, 2009).

1-3 A biology-inspired approach

One type of flying object that is commonly seen to successfully navigate buildings, is of course the insect. These small organisms have been optimized for flying in complex, dense environments through millions of years of evolution. Section 1-3-1 shows that the main instrument exploited by insects to achieve this, is optical flow (OF). Section 1-3-2 then looks at some OF based control strategies and a comparison of nature with technology.

1-3-1 Sense Perception in Insect Flight

Flying insects are capable of avoiding obstacles and navigating through unpredictable environments without any need for direct distance sensing such as sonar or laser rangefinders, (Aubépart & Franceschini, 2007). They depend on OF sensing processes for these capabilities. Attitude stabilisation is achieved through the halteres, which act as gyroscopic sensors, (Zufferey & Floreano, 2005), and the dorsal light response (DLR), the tendency to align with the up-down gradient of light intensity which is present in most daylight environments, (Neumann & Bulthoff, 2001).

OF is the angular speed at which any contrasting objects move past the eye. For several decades, coherent retinal image-shifts induced during locomotion have been considered a rich source of information about the momentary self-motion, (Krapp & Hengstenberg, 1996). The

local optical flow $\vec{\Omega}$ in viewing direction $\mathbf{n}$ (unit length) experienced by an observer moving at velocity $\mathbf{V}$ and rotational rate $\vec{\omega}$ in a stationary environment is given by Eq. (1-1) where d is the distance to the object seen in direction $\mathbf{n}$, (Neumann & Bulthoff, 2001). $\vec{\Omega}$ can be evaluated across the entire field of view (fov) with spherical coordinates Ψ for azimuth and Θ for elevation, resulting in a three dimensional vector field, the OF-field. Note that $\vec{\Omega}$ is orthogonal to the viewing direction $\mathbf{n}$. The observed OF field is a projection of $\vec{\Omega}$ onto a sensor surface.

$$\vec{\Omega}\left(\mathbf{n}\left(\Psi,\Theta\right)\right) = \frac{\mathbf{V} - \left(\mathbf{V}^T\mathbf{n}\right)\mathbf{n}}{d\left(\mathbf{n}\right)} + \vec{\omega} \times \mathbf{n} \tag{1-1}$$

(Aubépart & Franceschini, 2007) state that in flying insects, local OF is sensed by an elementary motion detector (EMD) neuron. The EMD takes input from at least two photoreceptors. It is sensitive to a particular OF direction. There exist two classes of models for these: *intensity-based* schemes (correlation techniques and gradient methods) and *token-matching* schemes. The electronic optical flow sensors (the ADNS-5030 (Avago, 2008)) used in this work are of the second kind.
The huge fov of the insect's compound eyes is mapped to a layer of EMDs. However, different kinds of self-motion may induce the same excitation in an EMD, because locally the corresponding optical flow fields may have the same orientation and magnitude. To extract information for flight control, a set of integrating neurons is wired to the EMDs. Each integrating neuron is sensitive to the input of a subset of EMDs corresponding to an OF pattern associated with a particular self-motion component. The firing of such an integrating neuron may then trigger for example a saccadic turn to avoid an obstacle or affect the beat frequency of the wings to adjust the perceived ground speed. (Franz, Chahl & Krapp, 2004) have investigated the capabilities of this circuitry by building an artificial egomotion estimator which emulates the insect neural structure. They report accurate and robust estimation of rotational rates and "reasonable" quality translation estimates. Note however, that they use prior knowledge od the distance distribution of the environment, which is something not available to flying insects.
Research performed by (Krapp & Hengstenberg, 1996) on the blowfly shows that its brain contains cells (the tangential neurons) receiving input from many local EMDs on their extended dendritic networks. Disabling a portion of those cells revealed that they are involved in course control and gaze stabilisation. 10 neurons, called VS neurons, are a subgroup of the tangential neurons. (Krapp & Hengstenberg, 1996) show that the VS neurons capture rotational self-motion from the OF-field, Figure 1-4. (Ibbotson, 1991) has identified a similar group of interneurons sensitive to translatory optical flow fields in the horizontal plain. One such neuron, the Hx, is sensitive to OF induced by horizontal translational self-motion, Figure 1-5.
The global structure of the response fields strongly suggests that distinct neurons in the fly's visual system act as sensory filters which extract specific components of self-motion from the momentary optical flow. Notice that the VS neurons have a low motion sensitivity in the ventral part ($\Theta < 0°$) of their receptive field. This is the direction where the strongest translation induced OF can be expected during flight at finite height. And indeed, Hx is especially sensitive in the ventral direction. This reduces the crosstalk from translation signals into rotation channels. The VS neurons have a high sensitivity in the dorsal part of their receptive field, which is around and above the horizon in symmetric upright flight. This exploits the large

distances to the horizon and clouds to preferentially sense rotations, because translational optical flow components decrease with distance, whereas rotational components do not. This can be deduced from Eq. (1-1).

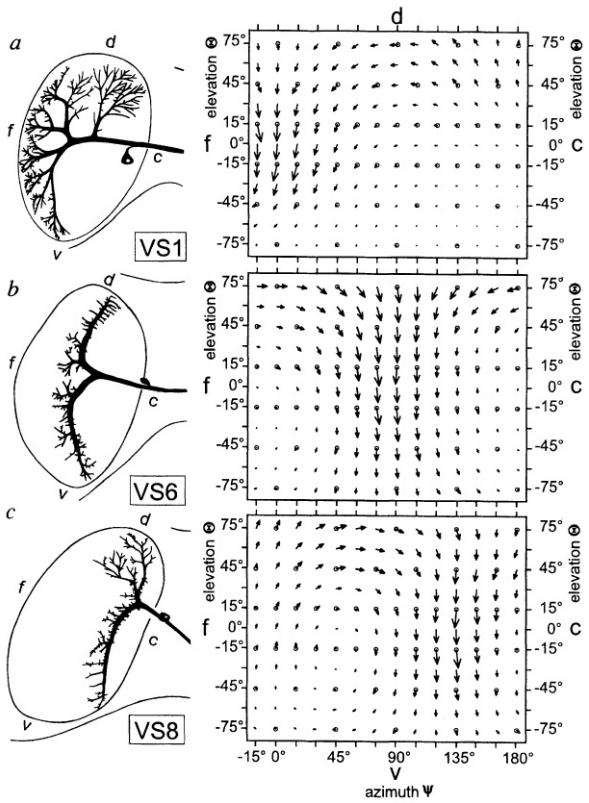


Figure 1-4: Anatomy and response of the neurons VS1 in subfigure a, VS6 in subfigure b and VS8 in subfigure c. The local motion responses are shown in maps representing the right visual hemisphere. The orientation of each arrow represents the local preferred direction and its length the normalised local motion sensitivity. The response fields of all VS neurons correspond to rotational optical flow fields with roughly horizontal axes oriented at different angles of azimuth (ψ), (Krapp & Hengstenberg, 1996)

Studies by (M. V. Srinivasan, 2006; Barron & Srinivasan, 2006; Ruffier & Franceschini, 2005; Franceschini, Ruffier & Serres, 2007) have investigated Opto-Motor Response (OMR) involved behaviour in flying insects. By training bees to fly through a tunnel towards a sucrose feeder, (M. V. Srinivasan, 2006) and (Barron & Srinivasan, 2006) have shown that the geometric properties of the pattern are largely irrelevant to the flight performance. Even very subtle, low contrast textures give nearly the same performance. This shows that OF provides a very robust way of controlling airspeed. Only when virtually no OF cues can be extracted, such as with a radial pattern, will the bees change their speed to about three times faster than before. They can still control their speed, but need faster motion to get the same level of interneuron firing.
The diameter of the tunnel cross-section is linearly proportional to the groundspeed. This means the bee will fly slower in a narrower tunnel. This can be explained by Eq. (1-1).

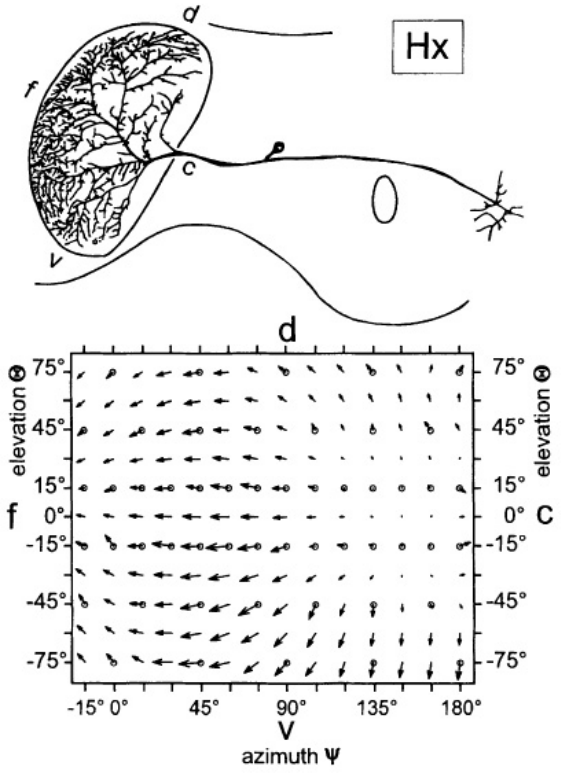


Figure 1-5: Anatomy and response of the Hx neuron. It is highly sensitive to horizontal back-to-front motion around $\psi \approx 45°$, (Krapp & Hengstenberg, 1996)

Consider the optical flow in the direction of flight, viewed perpendicular to the tunnel wall. The bee tries to fly at a certain OF-setpoint, Ω_{set}. In the tunnel, the rotations will be small: $\omega \approx 0$, so $\Omega_{set} \approx \frac{V}{D}$. When the distance D decreases, the bee must also decrease its groundspeed V proportionally to keep Ω_{set} constant.
Insects tend to fly through the centre of the tunnel cross-section. They do this by balancing OF. This has been shown by moving one wall relative to the opposing wall. Figure 1-6 shows the resulting flight paths: when the wall moves in the direction of flight, the insect flies closer to that wall and vice versa. Again, size of the pattern does not influence the position of flight paths.
(David, 1982) observed fruit flies flying upstream along the axis of a wind tunnel, attracted by the odour of fermenting banana. The walls of the cylindrical wind tunnel were decorated with a helical black-and-white striped pattern, so that rotation of the cylinder about its axis produced apparent movement of the pattern towards the front or the back. He could tune the pattern movement such as to keep the flies stationary. This reveals the OF setpoint at which the fly chooses to fly. The flies also compensated their airspeed in a headwind to keep the OF constant. Similar behaviour has been observed in honey bees by (Barron & Srinivasan, 2006), which will keep their preferred OF in headwinds with velocities up to 50% of the maximum recorded airspeed of the bee.
Studies of landing behaviour in flies by (M. V. Srinivasan, 2006) have revealed that the expansion of the image of the surface is used to control deceleration, trigger extension of the

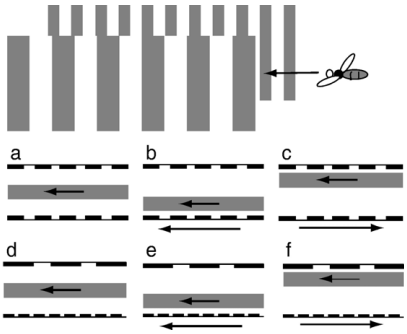


Figure 1-6: Illustration of an experiment which demonstrates that flying bees infer range from OF in a tunnel. The subfigures a-f show a top-down view of the tunnel with the flight path location and arrows showing the motion of the walls. Different pattern sizes between a-c and d-f have no effect on the flight paths, (M. V. Srinivasan, 2006)

legs and estimate time to contact. In Grazing landings, cues from image expansion are weak. The dominant pattern is then a translatory flow. Analysis of the landing trajectories revealed that horizontal speed is roughly proportional to height. The insect achieves this by holding the angular velocity of the image of the ground (ventral OF) constant during landing. The horizonal speed will reduce to zero upon contact, automatically ensuring a smooth landing.

1-3-2 Optical Flow Based Flight Control Strategies

To study the flight control structure of insects, (Franceschini et al., 2007) propose a visual feedback loop controlling the vertical lift component and implement it in a micro helicopter vehicle to compare its behaviour to that observed in the insect world. The hypothesis is that flying insects possess similar OF regulators in their brain.

The visual feedback loop, shown in Figure 1-7, takes the pitch angle θ as input and keeps

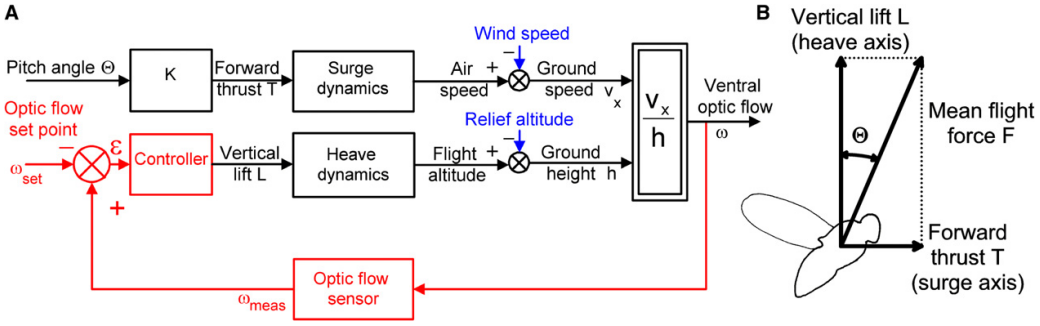


Figure 1-7: (A) Block diagram of the OF regulator, (B) general symmetric situation diagram of a flying insect, showing aerodynamic forces generated by the wings. Body drag and gravity are *not* shown, (Franceschini et al., 2007)

the ventral OF (directly below the vehicle/insect) constant and equal to a setpoint for a simple straight, symmetric flight case. This allows a micro-helicopter vehicle with a single downward looking OF-sensor to perform autonomous take-off, terrain following and landing

when constrained to fly in a vertical plain. The proposed control scheme was found to account for many reported behaviours in various species.

Take-off is achieved by pitching forward: the controller responds by increasing the lift force to keep the OF constant thus causing the vehicle to ascend. When the thrust component equals drag, the forward velocity becomes constant and the helicopter reaches a stationary cruise flight at a certain altitude determined by ω_{set} and the pitch angle. A local increase in relief altitude triggers a response by increasing the lift, thus maintaining a constant groundheight, which corresponds to terrain following.
Landing is achieved by pitching back to a vertical attitude again. The ensuing deceleration automatically initiates a proportional decrease in groundheight until landing occurs. When the vertical attitude is reached, the approach trajectory gets a constant angle and the velocity will automatically approach zero at touchdown. A typical flight profile is shown in figure 1-8.

Flying insects are known to descend under headwind and ascent during tailwind. The

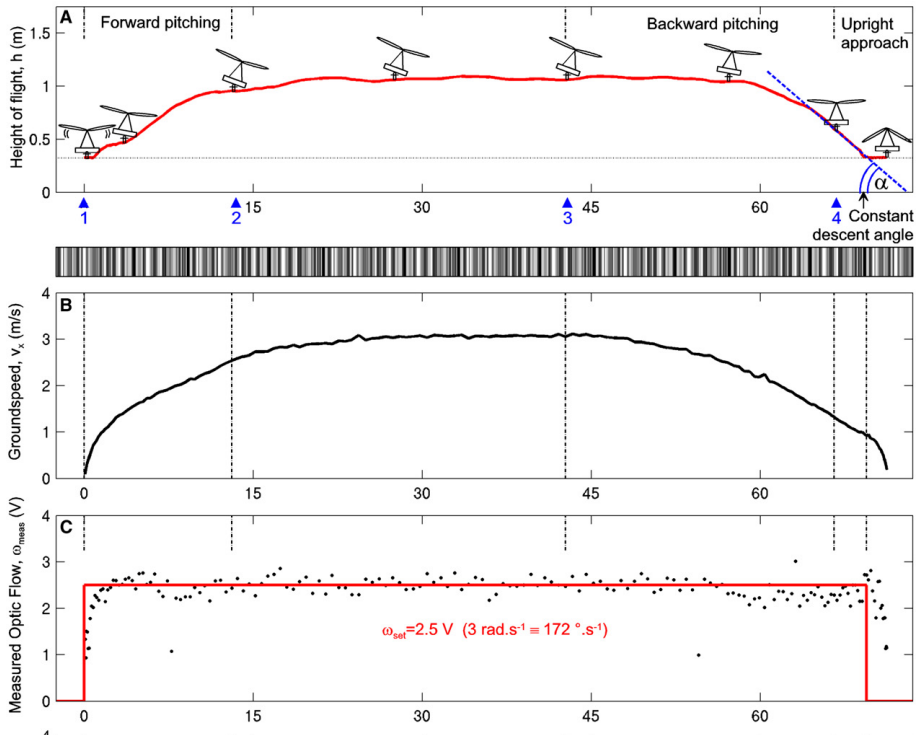


Figure 1-8: (A) flight profile, (B) speed profile, (C) ventral optical flow along the ground distance, (Franceschini et al., 2007)

OF-regulator can also explain this behavior. From Figure 1-7A, one can see that an increase in headwind speed reduces the groundspeed. This produces a lower optical flow ω and the controller responds by reducing the lift force to make the optical flow equal to the setpoint again. This will happen at a lower groundheight. The opposite occurs with reducing headwinds or increasing tailwinds.
Bees crossing mirror-smooth water have been observed to crash into it. This can be explained by considering that the OF-sensor may not pick up any optical flow from the water surface.

This causes $\omega_{meas} = 0$ and $\varepsilon = -\omega_{set} < 0$. The controller responds by decreasing the lift force until the insect hits the water.
The visual based control scheme proposed by (Franceschini et al., 2007) involves "OF-holding". It has the interesting effect that the groundheight becomes automatically proportional to the groundspeed. Insects may use this type of OF-regulator to fly manoeuvres like taking-off, terrain following, landing and responding appropriately to wind without being informed about groundheight, groundspeed, airspeed, windspeed or ascend or decent speed. Whether an insect's deceleration results from its intention to land or from the braking effects of a strong headwind, the feedback loop will always ensure smooth touchdowns because it pulls the insect down at a rate that is no faster than that of the decrease in groundspeed. During both take-off and landing, the closed visual-feedback loop will compensate for any disturbances, such as uneven terrain, wind gusts, and ground effects.
The model in Figure 1-7 differs from another one, reported by (M. Srinivasan, Zhang, Chahl, Barth & Venkatesh, 2000; Baird, Srinivasan, Zhang, Lamont & Cowling, 2006; Preiss & Kramer, 1984), where the OF controls the forward thrust T rather than the vertical lift L. Controlling T instead of L would, however, produce strikingly different flight patterns from those reported by previous authors, as follows:

(i) Instead of following a slanting terrain, as migrating butterflies and the micro helicopter built by (Franceschini et al., 2007) do, insects would gradually decelerate until touching the rising slope at a negligible speed and would thus inopportunely interrupt their journey.

(ii) Instead of descending in a headwind and rising in a tailwind, as honeybees, locusts, dung beetles, mosquitoes, and the micro helicopter do, insects would compensate for the unfavourable headwind by increasing their airspeed without changing their groundheight.

These two models can be reconciled, however, if in the block diagram of Figure 1-7A a second OF regulator is added to control the forward thrust by looking at the lateral OF. Experiments on tethered and free-flying flies and bees, as well as on tethered locusts, have long shown that motion detected in the lateral part of the eyes affects the forward thrust, and hence the forward speed. This additional hypothesis amounts to saying that the panoramic compound eye is subdivided into a ventral region and a lateral region, each of which is responsible for measuring the OF for a specific OF regulator: a ventral OF regulator controlling the vertical lift (and thus the groundheight), as suggested above, and a lateral OF regulator controlling the forward thrust (and thus the airspeed).

1-3-3 Conclusion

Using only OF and inertia, flying insects are able to perform a wide variety of manoeuvres while avoiding collisions in diverse environments. They show that OF can be a very robust and rich source of information on self motion.
The fact that they can't cope with glass may be explained partly by considering the relatively short period of time that glass windows have become common in the environment of flying insects. It is also due to the fact that glass cannot be perceived using optical flow. This is

one possible shortcoming of the concept, which may be alleviated by addition of another type of sensor.
The insect brain works like a massively parallel computer for the processing of OF: many neurons directly behind the photoreceptors are involved with the calculation of the OF field $\vec{\Omega}$. A limited number of integrating neurons capture specific components of $\vec{\Omega}$. These components correspond to certain modes of self motion, i.e. translations and rotations. The output of these integrating neurons is then used to control the flight.
Insect flight control is not based on the actual velocities of flight, but instead uses the translatory OF components directly. These OF components are basically the velocities scaled by viewing distances. This gives rise to some very elegant natural behaviours:

- insects fly through the middle of a tunnel or narrow opening by balancing the OF;
- grazing landings (with some forward velocity) are performed smoothly by keeping OF constant while approaching the surface. This means forward velocity reduces to zero at touchdown;
- Insects are known to descend in headwinds. They do this by holding the ventral OF constant by adjusting the lift force. By doing so, they automatically benefit from wind shear;
- Terrain following is also a consequence of the same control loop.

1-4 Research Goal

The problem of autonomous indoor flight requires some form of sensor fusion, as concluded in the problem statement in Section 1-2. Nature suggests the use of vision and to exploit the information contained in the motion of features across the field of view, or OF, as the means to observe the motion of an MAV. This will give information about the distances to obstacles as well.
While insects only separate the translatory and rotatory OF components in Eq. (1-1), and use the translatory components directly for flight control, this is not a convenient concept for the control of an MAV. In order to apply conventional controller algorithms and possibly employ Simultaneous Localization And Mapping (SLAM), it is necessary to know the actual velocity and distances. This may be done by using several optical flow cameras facing in different directions. The body velocity components (u, v, w) and body rotations (p, q, r) are the same for each camera while the viewing distances are different. Solving for these motion components and viewing distances is theoretically possible using six OF cameras and three accelerometers.
The goal of this thesis is to investigate the feasibility and performance of this sensor concept.

Part I

Theory

Chapter 2

System Description

2-1 Optical flow

2-1-1 Introduction

The sensor concept proposed in this work relies heavily on optical flow (OF). OF is known in human and animal perception to support a wide variety of visual tasks, including 3D shape acquisition, oculomotor control, perceptual organization, object recognition and scene understanding, (Fleet & Weiss, 2005). The function of OF useful for this work is the sense of self motion induced by visually perceiving the movement of the environment across the entire field of view. This sense of self motion is used by many species to navigate through diverse environments, be it on foot, in flight or while swimming. A common requirement of the process is the presence of stationary surroundings sufficiently close to the moving observer. The surroundings should also reflect sufficient amounts of light and have some texture. However, looking at the example of flying insects shows that nature is capable of using OF in situations with very little light and sparse texture. This makes it a potentially robust and information rich quantity to exploit for navigation in complex and dense environments such as inside buildings.

2-1-2 Definition

The quantity optical flow is defined as the angular rate at which a point on an object moves in relation to an optical sensor. As the sensor lens projects the light within its field of view onto a two dimensional surface, the angular rate of features in the image is two dimensional as well. When the sensor is attached to a body flying through a three dimensional space with stationary objects in it, the optical flow of points on those objects will be a function of the body motion relative to the objects (translations u, v, w and rotations p, q, r) and the distance from the sensor to the points of interest. This assumes that the surface of the objects has adequate texture and is sufficiently illuminated for the sensor to distinguish and track features.

In general, optical flow will vary across the field of view of the sensor, whose coordinates are the azimuth and elevation angles (Ψ and Θ, respectively), and this gives rise to an optical flow vector field.

> ***Definition 2.1*** *The local optical flow $\vec{\Omega}(\Psi,\Theta,t)$ in viewing direction $\mathbf{n}(\Psi,\Theta)$ experienced by an observer moving at velocity $\mathbf{V}(t)$ and rotational rate $\vec{\omega}(t)$ in a stationary environment is, from (Neumann & Bulthoff, 2001):*
>
> $$\vec{\Omega}(\Psi,\Theta,t) = \frac{\mathbf{V}(t) - \left(\mathbf{V}(t)^T \mathbf{n}(\Psi,\Theta)\right)\mathbf{n}(\Psi,\Theta)}{d(\Psi,\Theta,t)} + \vec{\omega}(t) \times \mathbf{n}(\Psi,\Theta) \tag{2-1}$$
>
> *where $d(\Psi,\Theta,t)$ is the distance to the object seen in direction $\mathbf{n}(\Psi,\Theta)$.*

Eq. (2-1) results in a three dimensional optical flow vector which is orthogonal to the viewing direction $\mathbf{n}$. The sign convention is such that a translation in a positive direction of the body axes has a positive contribution to the optical flow.

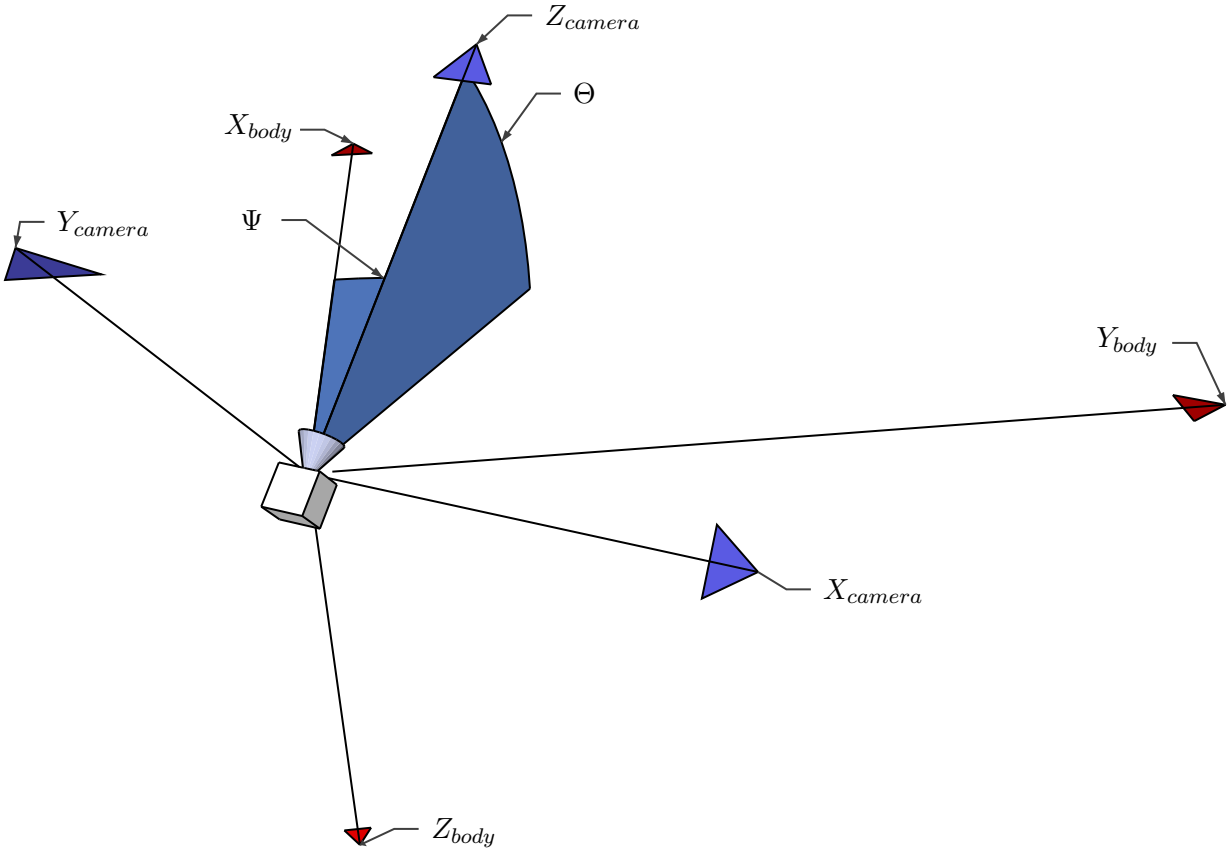


Figure 2-1: Camera rotation

However, $\vec{\Omega}$ is still described in the body reference frame. Assume that no deformations occur in the camera projection, i.e. a homographic projection is applicable. Then the optical flow perceived at angles (Ψ,Θ) from the optical axis is equivalent to projecting $\vec{\Omega}$ onto the camera $X-Y$ plane rotated by (Ψ,Θ) as defined in Figure 2-1. This is done with a projection matrix P:

Let the direction cosine matrix (DCM) using the ZYX order of rotation with Euler angles φ, θ and ψ be defined by

$$DCM = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\varphi\sin\theta\cos\psi - \cos\varphi\sin\psi & \sin\varphi\sin\theta\sin\psi + \cos\varphi\cos\psi & \sin\varphi\cos\theta \\ \cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi & \cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi & \cos\varphi\cos\theta \end{bmatrix} \tag{2-2}$$

Note that the camera axes are interchanged with respect to the body frame such that the camera Z-axis coincides with its optical axis, no φ rotation is applied, $\psi = \Psi$ and $\theta = \Theta$. These facts are taken into account by pre-multiplying DCM by a transformation matrix and substitution of Ψ and Θ:

$$P = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} DCM = \begin{bmatrix} \sin\Psi & -\cos\Psi & 0 \\ \sin\Theta\cos\Psi & \sin\Theta\sin\Psi & \cos\Theta \end{bmatrix} \tag{2-3}$$

The two dimensional optical flow field $\vec{\Omega}^*$ observed by the camera is then:

$$\vec{\Omega}^*(\Psi, \Theta, t) = P\vec{\Omega}(\Psi, \Theta, t) \tag{2-4}$$

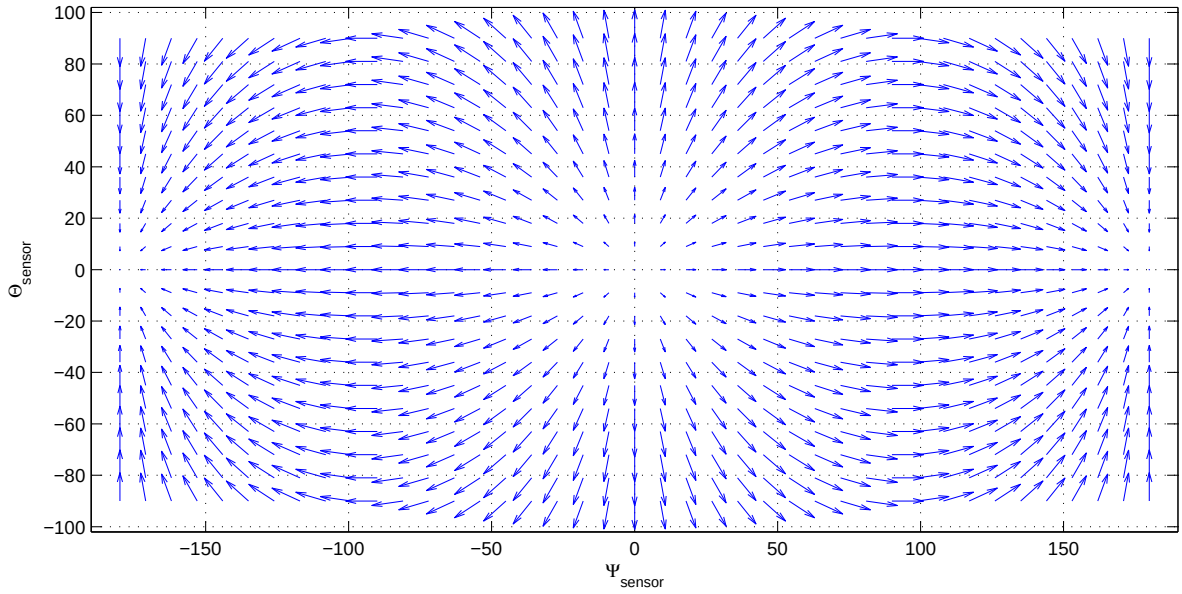
An example of the optical flow field resulting from a translation u and from a rotation q is shown in Figure 2-2 a and b, respectively. Of course, in general, $\vec{\Omega}^*$ will be a superposition of all motion components.

2-2 Concept

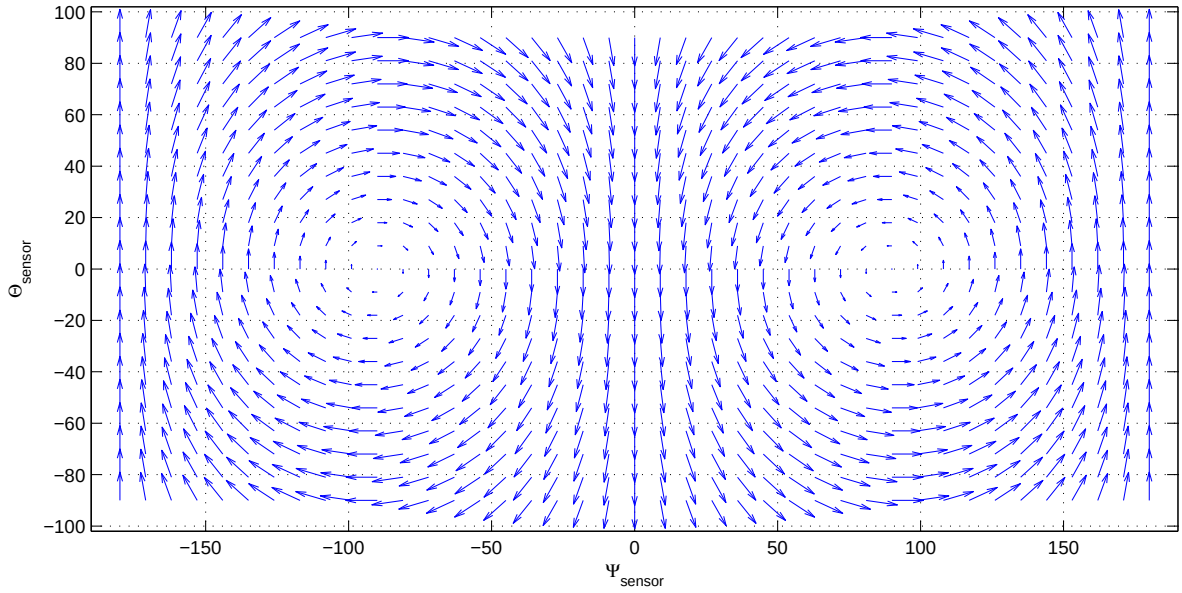
The research goal described in Section 1-4, calls for a state estimation method which will enable indoor flight control and obstacle avoidance. Buildings generally have interiors with little room to manoeuvre and bad Global Positioning System (GPS) reception. These properties of the environment makes traditional state estimation based on an inertial measurement unit (IMU)/GPS sensor package insufficient for a micro aerial vehicle (MAV) with an indoor mission requiring some level of autonomy. Section 1-3 in the introduction suggests to use the concept of OF for indoor state estimation.

As has been explained in Section 2-1, OF consists of a linear combination of translational and rotational motion. The fact that the translations are scaled by the viewing distance complicates the matter of extracting the motion state. However, if one has several cameras pointed in different directions, the same motion will determine each OF signal. By solving the associated observation equations simultaneously, it may be possible to solve for all the motion states and viewing distances. This would serve both the flight control and obstacle avoidance goals.

The simplest camera configuration might be to mount six sensors to the sensor board in opposing, orthogonal directions along the body axes. This is shown in Figure 2-3. In this way, there are three pairs of sensors looking in opposing directions. These sensors will see the same motion components, with the only difference being that the translational motion is divided by a different viewing distance. Six OF-sensors do produce an observable system in the case without gravity, but of course this is not very practical. Because the aircraft is assumed to fly in the presence of a gravitational field, a 3-axis accelerometer is also required. As the



(a) translation u



(b) rotation q

Figure 2-2: Optical flow fields resulting from different motion components

gravitational acceleration and its direction enter into the system dynamics equations, one must solve for the attitude as well. As will be shown in Chapter 3, the system is observable, when the three accelerometer measurements are available. As long as the aircraft moves, the sensor package provides enough data to solve for the six motion components (u, v, w, p, q, r), the six viewing directions ($d_1 - d_6$) and the two Euler angles involved with the gradient of the gravitational field, that is pitch angle θ and roll angle φ.

The type of optical flow sensor used in this work estimates the optical flow at the point where

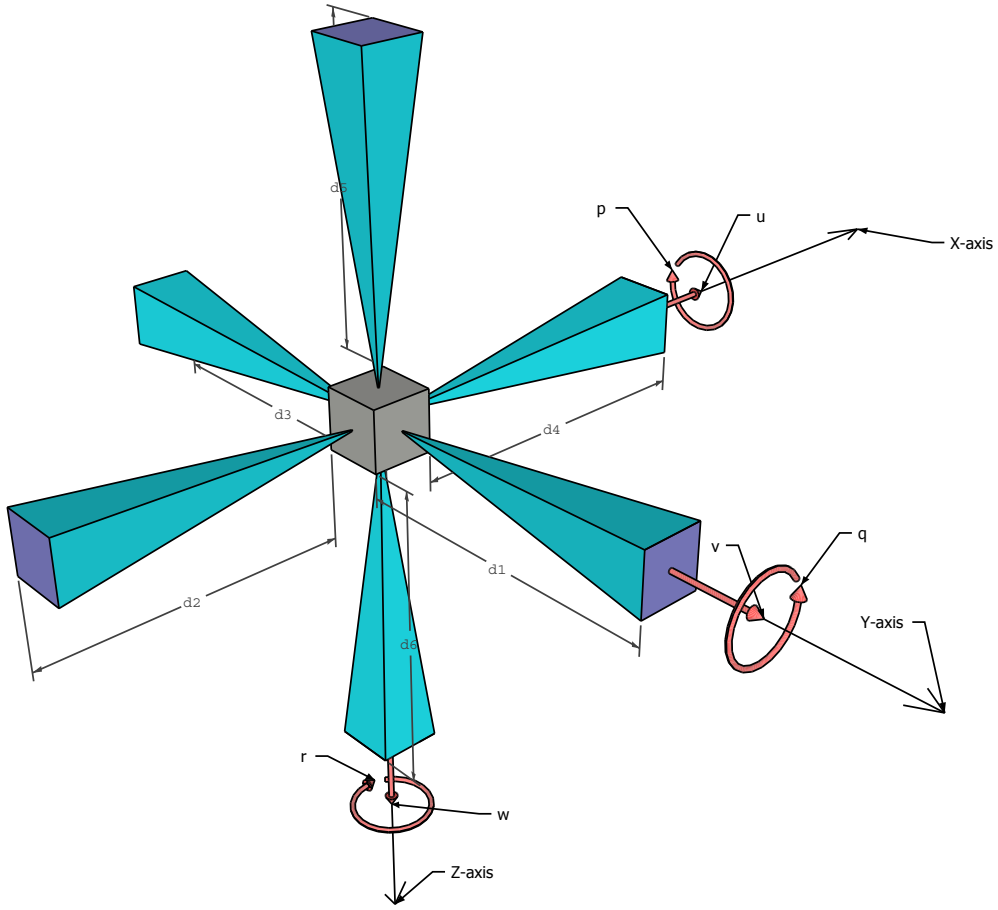


Figure 2-3: Diagram of the geometry, the distance subscripts refer to the sensor number

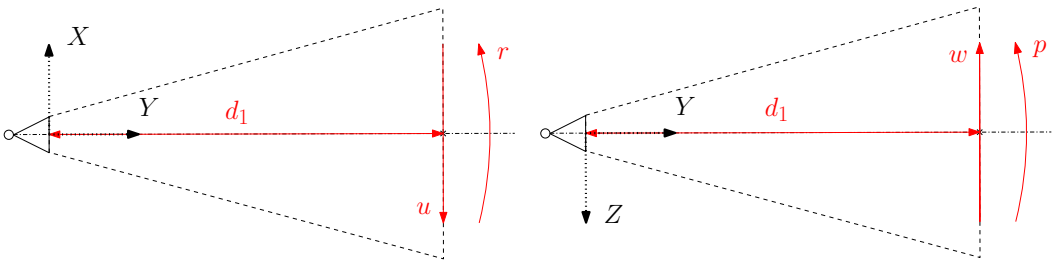


Figure 2-4: Sensor 1 optical flow

its optical axis intersects the surface of the object it is looking at. It does this by averaging the optical flow across its field of view. As explained in Section 2-1, in general, the optical flow will vary across the sensor field of view. It does so in a continuous and differentiable manner as long as $d(\Psi, \Theta)$ in Eq. (2-1) is continuous and differentiable with Ψ and Θ. The accuracy of this mean value optical flow estimate depends on mild changes of distance. The probability that greatly varying distances are present in the field of view reduces as the field of view becomes more narrow. Therefore, in this respect, a narrow field of view is beneficial to the accuracy of the estimate. However, considering the small aperture (1.5 mm), the light flux on the sensor surface may become too small if the field of view is very narrow.

An example diagram to derive the optical flow for sensor 1 in the schematic Figure 2-3 is shown in Figure 2-4. The arrows show the positive directions of the body motion from the sensor perspective. Applying Eq. (2-1) and Eq. (2-4) results in two optical components along the body X- and Z-axes, denoted as Ω_{X_1} and Ω_{Z_1}, respectively. The resulting equations for all optical flow sensors are:

$$\begin{array}{lll} X-Y\ plane & X-Z\ plane & Y-Z\ plane \\ \Omega_{X_1} = \frac{u}{d_1} - r & \Omega_{Z_2} = \frac{w}{d_2} + q & \Omega_{Z_1} = \frac{w}{d_1} + p \\ \Omega_{Y_2} = \frac{v}{d_2} - r & \Omega_{X_6} = \frac{u}{d_6} + q & \Omega_{Y_6} = \frac{v}{d_6} - p \\ \Omega_{X_3} = \frac{u}{d_3} + r & \Omega_{Z_4} = \frac{w}{d_4} - q & \Omega_{Z_3} = \frac{w}{d_3} - p \\ \Omega_{Y_4} = \frac{v}{d_4} + r & \Omega_{X_5} = \frac{u}{d_5} - q & \Omega_{Y_5} = \frac{v}{d_5} + p \end{array} \tag{2-5}$$

2-3 Sensors

2-3-1 Optical Flow Sensors

The OF-sensors used (the ADNS-5030 (Avago, 2008)) were specifically designed for optical computer mice. In that application, the environment is well conditioned. The sensors look at a flat surface at a very specific distance, the light source illuminating the surface is constant and in phase with the sensor electronic shutter. In contrast, the application for which they are to be used brings widely varying conditions. The light source has a wide range of intensity and intensity fluctuations, e.g. from the power grid. Also, the viewing distances may vary from close to zero to the largest distance in the building. Suffice to say that the challenge will be to adapt the sensor system to cope with these varying conditions.
One indication that this type of OF-sensor may be employed successfully for state estimation purposes in MAVs is the research performed by (Barber, Griffiths, McLain & Beard, 2007) on an autonomous landing system for a blended wing body (BWB) UAV. They use one downward pointing Avago type OF-camera in combination with GPS groundspeed data to calculate height above ground. In this case, the method yields accurate and reliable estimates. To use these sensors in a motion state estimation application for indoor flight, the lens must be changed in order to focus the light from longer distances. The original lenses have a focal length of 2.7 mm. The sensor surface has sides of 0.8 mm and 15 x 15 pixels. The required lens properties can be calculated from the lens formula relating focal distance f, subject distance v and image distance b:

$$\frac{1}{f} = \frac{1}{b} + \frac{1}{v} \tag{2-6}$$

We need to know f and b. A field of view (fov) of 10° gives a good trade off between required sampling frequency, light intensity and surface feature size. A relation between sensor length B, image distance b and fov angle α can be found from the geometry in Figure 2-5:

$$\tan\left(\frac{\alpha}{2}\right) = \frac{B}{2b}$$

$$\Rightarrow b = \frac{B}{2\tan\left(\frac{\alpha}{2}\right)}$$

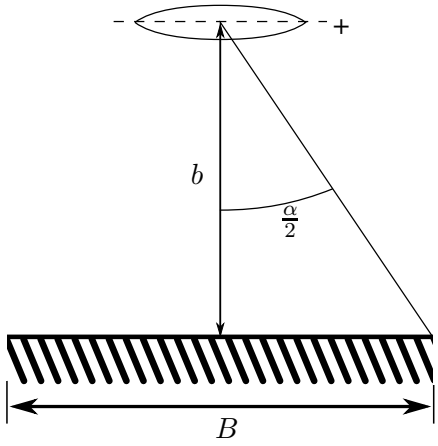


Figure 2-5: Simple lens geometry

Then $f = \frac{1}{\frac{1}{b}+\frac{1}{v}}$.

The range of focused distances, also known as depth of field (dof), is determined by the

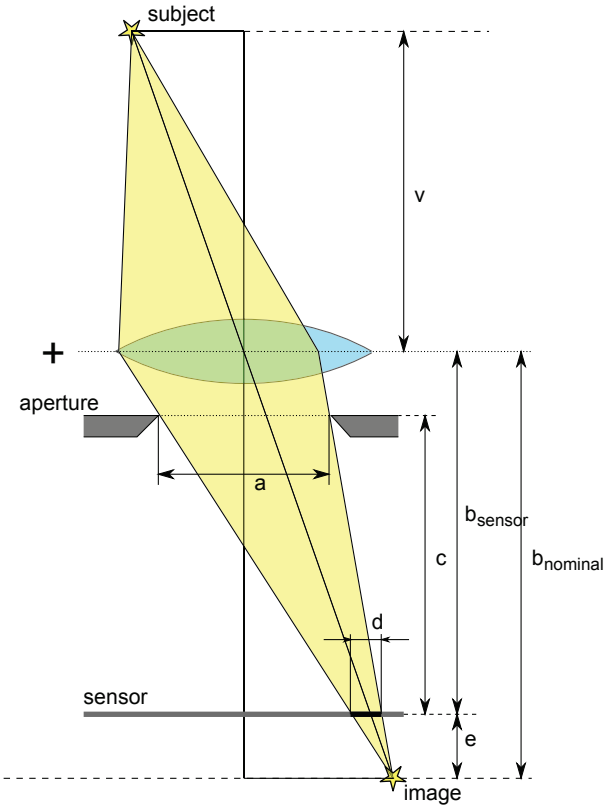


Figure 2-6: Circle of confusion geometry

circle of confusion (CoC) and the pixel size. The CoC is the area on the sensor at which light from a point on the subject is projected. Using a ray optics model, this area has a sharp circular limit and constant light intensity distribution. In reality, the area is soft-edged due to diffraction and aberrations and the shape is determined by the shape of the aperture, (Ray,

2002). For estimating the depth of field in this simple lens system however, ray optics gives a good approximation. To get the same level of sharpness on a sensor with smaller pixels, the CoC diameter must be smaller as well. The CoC diameter d can be derived from the geometry in Figure 2-6. Given the lens focal length f, distance from lens to sensor b_{sensor}, aperture to sensor c, aperture diameter a and the subject distance v, the true image distance $b_{nominal}$ follows from Eq. (2-6):

$$b_{nominal} = \frac{1}{\frac{1}{f} - \frac{1}{v}}$$

The sensor distance mismatch is

$$e = b_{nominal} - b_{sensor}$$

and the CoC diameter

$$d = \left| \frac{e}{c+e} \right| a \tag{2-7}$$

The criterion for a focused image depends on the purpose of the image. In photography, the

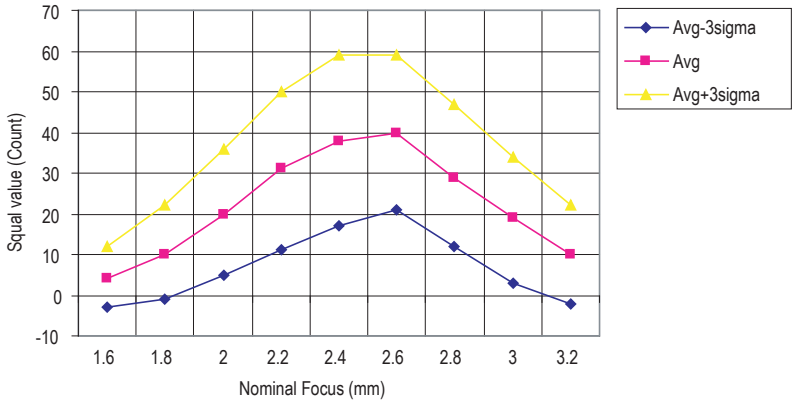


Figure 2-7: Surface quality vs subject distance from (Avago, 2008)

maximum allowable CoC is determined by the human eye observing the printed photograph. But in this case, a measure directly related to the sensor should be chosen. From (Avago, 2008) $c = 2.96$ mm and $a = 1.5$ mm and Figure 2-7 shows that the SQUAL value of the sensor, which is a measure of the number of features tracked by the digital signal processor (DSP), drops significantly (25%) when $e = 0.3$ mm. If this is taken as the threshold for an acceptable sensor performance, the maximum CoC d_{max} can be found:

$$d_{max} = \frac{0.3}{296 + 03} 15 = 0.18 \text{ mm}$$

Note that the pixels have sides of 0.053 mm, so d_{max} is more than 3 times the pixel size. d_{max} together with the optical geometry (a, c, b_{sensor} and f) determines the depth of field defined as $[v_{min}, v_{max}]$. Rewriting Eq. (2-6) and Eq. (2-7), this can be expressed as

$$\begin{aligned} v_{min} &= \left\{ \frac{1}{f} - \frac{1}{b_{sensor} + \frac{dc}{a-d}} \right\}^{-1} \\ v_{max} &= \left\{ \frac{1}{f} - \frac{1}{b_{sensor} - \frac{dc}{a+d}} \right\}^{-1} \end{aligned} \tag{2-8}$$

Choosing a field of view of 10° and the dimensions of the ADNS-5030 chip, the graph of the CoC diameter versus v in Figure 2-8 shows that the depth of field is very large ($v_{min} = 56$ mm and $v_{max} = \infty$). This means that the sensor can be used at any distance larger than 5.6 cm.

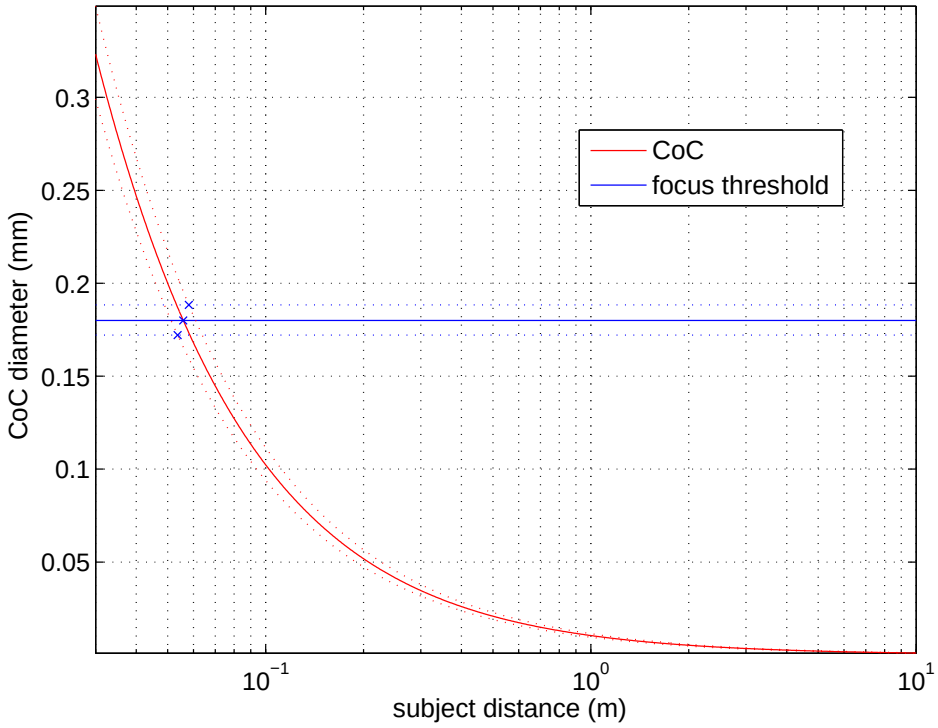


Figure 2-8: Depth of field for an optical flow sensor based on the Avago ADNS-5030 chip with 10° field of view

2-3-2 Accelerometers

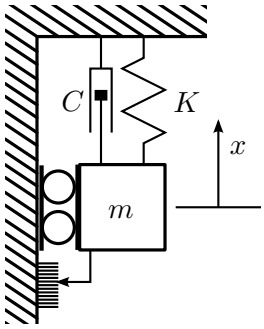


Figure 2-9: Schematic of the accelerometer

An accelerometer is a device measuring specific force f_s. This is not equal to the true acceleration a as the name suggests, but rather the true acceleration minus gravitational acceleration

g. As shown in Figure 2-9, conventional accelerometers are based on a proof mass m connected to its surroundings through a spring-damper suspension (K,C) which only allows movement in the x-direction. Any resulting force acting on the proof mass in the sensitive direction results in a deformation of the spring. This is then measured by e.g. a capacitance- or induction-based position pickoff. An analog circuit translates the signal to an output voltage, which may then be sampled by an A/D-converter for use in a digital avionics system. It should be noted here that the mass-spring-damper system comprising the core of the accelerometer has its own dynamics:

$$\ddot{x} + \frac{C}{m}\dot{x} + \frac{K}{m}x = f_s$$

So x does not have a linear relation with the specific force. However, one may avoid these problems by choosing the natural frequency ω_n well above the Nyquist frequency which is twice the sample frequency ω_s and by using an appropriate analog low pass filter. So if $\omega_n = \sqrt{\frac{K}{m}} \gg 2\omega_s$ one has a good approximation for f_s by looking directly at x.
The fact that an accelerometer measures the difference between acceleration and gravity may be explained as follows. The force of gravity is a field force acting equally on all objects with mass, while accelerations of the accelerometer casing caused by other external forces are transmitted to the proof mass through its suspension. So the suspension will only be deformed by external forces other than gravity, as any gravitational force acting on the proof mass is not transmitted to it through the suspension. With only gravity present, the accelerometer would measure zero signal, because the proof mass suspension would not deform. The acceleration of the accelerometer would be equal to the gravitational acceleration, cancelling each other.
In three dimensional space the specific force is a 3 element vector $\vec{f_s}$:

$$\vec{f_s} = \vec{a} - \vec{g} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} - DCM_{e\to b} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2-9}$$

So for the directions orthogonal to $\vec{g}$, the accelerometer measures $\vec{a}$ directly.
It follows that to measure the specific force vector, one requires three measurement directions. Preferably, those directions would be mutually orthogonal, but theoretically the accelerometer provides sufficient information to construct the specific force vector as long as the directions span three space. This means that most commercially available accelerometers are packaged as a set with three proof masses mounted in three mutually orthogonal directions. The package is marked on the outside indicating those directions to facilitate correct mounting.
Like any measuring instrument, accelerometers generally suffer from a number of measurement errors as illustrated in Figure 2-10:

- bias error b: a (quasi-)constant value being added to the measurement signal;
- scale error Δa: a (quasi-)constant value denoting the difference with the expected slope of the output graph;
- higher order errors: the output graph is assumed to be linear, any deviations from this assumed linear relation produce errors;
- direction errors: alignment errors of the proof masses showing up as cross-coupling effects between the output directions;
- noise: often modelled as white noise with standard deviation σ.

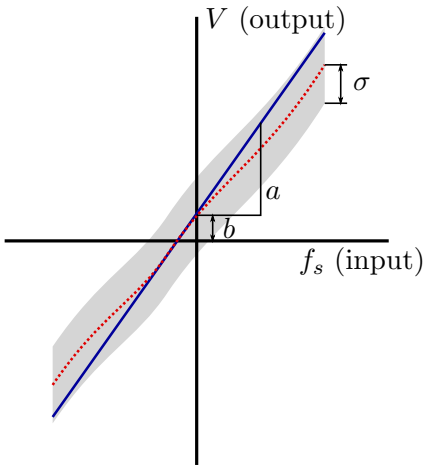


Figure 2-10: Calibration graph with error types common to accelerometers

2-3-3 Observation Equations

The sensors described above provide input which can be used to estimate the state of the aircraft on which they are mounted. To do so, it is necessary to know how the observation signals $\vec{z}$ depend on the state variables$\vec{x}$. This is described by the observation equations, denoted by $\vec{h}(\vec{x})$.
The equations of the optical flow sensors have been derived in Sections 2-1 and 2-2 as Eq. (2-5). The equations for the accelerometers should be added to complete $\vec{h}(\vec{x})$. They can be derived by elaborating Eq. (2-9), which was written in earth reference frame. The true acceleration of a vehicle in the body reference frame is

$$\vec{a} = \dot{\vec{V}} + \vec{\omega} \times \vec{V} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

and the gravitational acceleration vector is

$$\vec{g} = DCM_{e\to b}(\theta, \varphi) \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

where $DCM_{e\to b}(\theta, \varphi)$ is the transformation matrix from earth to body reference frame involving only pitch and roll in this case, because $\vec{g}$ is vertical and therefore independent of heading angle ψ. $DCM_{e\to b}(\theta, \varphi)$ can be expressed as

$$DCM_{e\to b}(\theta, \varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} =$$

$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ \sin\varphi\sin\theta & \cos\varphi & \sin\varphi\cos\theta \\ \cos\varphi\sin\theta & -\sin\varphi & \cos\varphi\cos\theta \end{bmatrix}$$

So the accelerometer output becomes

$$\vec{f_s} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} - g \begin{bmatrix} -\sin\theta \\ \sin\varphi\cos\theta \\ \cos\varphi\cos\theta \end{bmatrix} \tag{2-10}$$

Taking Eq. (2-5) and Eq. (2-10) together, the observation equations can be written as

$$\vec{z} = \vec{h}(\vec{x}) = \begin{bmatrix} \frac{u}{d_1} - r \\ \frac{w}{d_1} + p \\ \frac{v}{d_2} - r \\ \frac{w}{d_2} + q \\ \frac{u}{d_3} + r \\ \frac{w}{d_3} - p \\ \frac{v}{d_4} + r \\ \frac{w}{d_4} - q \\ \frac{u}{d_5} - q \\ \frac{v}{d_5} + p \\ \frac{u}{d_6} + q \\ \frac{v}{d_6} - p \\ \dot{u} + qw - rv + g\sin\theta \\ \dot{v} + ru - pw - g\sin\varphi\cos\theta \\ \dot{w} + pv - qu - g\cos\varphi\cos\theta \end{bmatrix} \tag{2-11}$$

Eq. (2-11) refers to the case where the sensor package is aligned with the body frame of reference of the aircraft. In general, this is not the case however, and $\vec{h}(x)$ can be modified to accommodate a sensor package which has a fixed rotation of (φ_s, θ_s, ψ_s) with respect to the body frame of reference and which is displaced by $\vec{\delta}$ from the center of gravity (CoG). The rotation matrix, or DCM, from the body frame of reference to the sensor frame of reference is given by

$$DCM_{b\to s}(\varphi_s, \theta_s, \psi_s) =$$

$$\begin{bmatrix} \cos\theta_s\cos\psi_s & \cos\theta_s\sin\psi_s & -\sin\theta_s \\ \sin\varphi_s\sin\theta_s\cos\psi_s - \cos\varphi_s\sin\psi_s & \sin\varphi_s\sin\theta_s\sin\psi_s + \cos\varphi_s\cos\psi_s & \sin\varphi_s\cos\theta_s \\ \cos\varphi_s\sin\theta_s\cos\psi_s + \sin\varphi_s\sin\psi_s & \cos\varphi_s\sin\theta_s\sin\psi_s - \sin\varphi_s\cos\psi_s & \cos\varphi_s\cos\theta_s \end{bmatrix} \tag{2-12}$$

Let $\vec{V}_{body}$, $\vec{\omega}_{body}$ denote the motion of the aircraft in the body frame of reference.
The optical flow sensors will be looking in different directions, so the rotational rate should be rotated by $DCM_{b\to s}$ to express it in the sensor frame of reference:

$$\vec{\omega}_{sensor} = DCM_{b\to s}\,\vec{\omega}_{body} \tag{2-13}$$

The velocity must also be rotated by (φ_s, θ_s, ψ_s), but in addition, it is affected by $\vec{\delta}$:

$$\vec{V}_{sensor} = DCM_{b\to s}\left(\vec{V}_{body} + \vec{\omega}_{body} \times \vec{\delta}\right) \tag{2-14}$$

With these transformations, the optical flows for the rotated and displaced sensor package can be expressed in $\vec{V}_{body}$ and $\vec{\omega}_{body}$ by substitution of Eq. (2-13) and Eq. (2-14) into Eq. (2-5).

The accelerometer output $\vec{f_s} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix}^T$ will be affected by (φ_s, θ_s, ψ_s) and $\vec{\delta}$ as follows:

$$\vec{f}_{s_{sensor}} = DCM_{b\to s}\left(\vec{f}_{s_{body}} - \begin{bmatrix} q^2+r^2 & \dot{r}-pq & -\dot{q}-pr \\ -\dot{r}-pq & p^2+r^2 & \dot{p}-qr \\ \dot{q}-qr & -\dot{p}-qr & p^2+q^2 \end{bmatrix}\vec{\delta}\right) \tag{2-15}$$

where $\vec{f}_{s_{body}}$ refers to the specific force at the CoG in the body frame of reference, Eq. (2-10).

2-4 Dynamics

The sensor concept proposed here is intended to be used with a MAV, which is specifically designed to fly indoors.
In order to apply a Kalman Filter for the estimation of the motion state of this aircraft, its dynamics equations or equations of motion (EoM) need to be known. In the case of indoor aircraft two important assumptions can be made:

1. The majority of MAVs can be considered symmetrical about a symmetry plane spanned by their body X- and Z-axes. This is not critical, but it simplifies the EoM;
2. By approximation, the aerodynamic moments $\vec{\mathcal{M}}$ can be considered to only depend on the controller input.

An MAV designed for indoor flight must be capable of hover and very slow flight and it will be limited to low velocities in the confined environment of a building interior. Common design solutions use moving wings to generate all the aerodynamic forces and moments required for flight. In such a design the moments will be generated by varying torques or applying differential thrust. Within the limited flight envelope the control derivatives can be considered constant. External disturbances like wind gust and turbulence are also expected to be negligible inside most buildings. This means that the aerodynamic moments do not depend significantly on the states and can be directly calculated from the control inputs provided that the control derivatives are known.
As this is a conceptual study, aiming to investigate the general performance of a new sensor concept, no specific aerodynamic model has been used in the simulation. Therefore $\vec{\mathcal{M}}$ is treated as a known quantity.
In the pendulum experiment, described in Section 9-1, the Kalman filter runs without moment inputs as the motion is autonomous. Once the pendulum is moving, there are no outside disturbances apart from a slight drag moment. The dynamics model in the Kalman filter includes Eq. (2-16) in this case:

$$\mathcal{M}_y = \frac{(A_x - g\sin\theta)\, I_y}{R} \tag{2-16}$$

where A_x is expressed in the body-fixed frame of reference.

The second assumption enables the use of a generic dynamics model requiring only vehicle inertias (mass m and the moments of inertia I_x, I_y, I_z and J_{xz}) and aerodynamic moment inputs which can be directly calculated from the control of the aircraft. The accelerometer

measurements can be substituted for the aerodynamic forces: $X = A_x m$, $Y = A_y m$ and $Z = A_z m$. So the input vector becomes

$$\vec{u} = \begin{bmatrix} A_x & A_y & A_z & \mathcal{M}_x & \mathcal{M}_y & \mathcal{M}_z \end{bmatrix}^T$$

The resulting generic nonlinear EoM from (Mulder, Staveren & Vaart, 2000) are:

$$\begin{aligned} -W\sin\theta + X &= m\left(\dot{u} + qw - rv\right) \\ W\cos\theta\sin\phi + Y &= m\left(\dot{v} + ru - pw\right) \\ W\cos\theta\cos\phi + Z &= m\left(\dot{w} + pv - qu\right) \\ \mathcal{M}_x &= I_x\dot{p} + \left(I_z - I_y\right)qr - J_{xz}\left(\dot{r} + pq\right) \\ \mathcal{M}_y &= I_y\dot{q} + \left(I_x - I_z\right)rp + J_{xz}\left(p^2 - r^2\right) \\ \mathcal{M}_z &= I_z\dot{r} + \left(I_y - I_x\right)pq - J_{xz}\left(\dot{p} - rq\right) \end{aligned} \tag{2-17}$$

The derivatives of the motion states can be written explicitly as

$$\begin{aligned} \dot{u} &= rv - qw - g\sin\theta + A_x \\ \dot{v} &= pw - ru + g\cos\theta\sin\varphi + A_y \\ \dot{w} &= qu - pv + g\cos\theta\cos\varphi + A_z \\ \dot{p} &= \frac{\mathcal{M}_x I_z - qr{I_z}^2 + qrI_z\,I_y - J_{xz}\,pq I_y + J_{xz}\,pq I_x - J_{xz}^2 rq + J_{xz}\,\mathcal{M}_z + J_{xz}\,pq I_z}{-J_{xz}^2 + I_x\,I_z} \\ \dot{q} &= \frac{\mathcal{M}_y - (I_x - I_z)rp - J_{xz}\left(p^2 - r^2\right)}{I_y} \\ \dot{r} &= \frac{\mathcal{M}_z I_x - pqI_x\,I_y + pqI_x^2 - J_{xz}\,rqI_z + J_{xz}\,rqI_y - J_{xz}^2 pq + J_{xz}\,\mathcal{M}_x - J_{xz}\,rqI_x}{-J_{xz}^2 + I_z\,I_x} \end{aligned} \tag{2-18}$$

where $W = mg$.
Recall from Section 2-2 that the state vector to be estimated is

$$\vec{x} = \begin{bmatrix} u & v & w & p & q & r & \theta & \varphi & d_1 & \dots & d_6 \end{bmatrix}^T$$

The kinematics for $\dot{\theta}$ and $\dot{\varphi}$ are

$$\begin{aligned} \dot{\theta} &= q\cos\varphi - r\sin\varphi \\ \dot{\varphi} &= p + q\sin\varphi\tan\theta + r\cos\varphi\tan\theta \end{aligned} \tag{2-19}$$

and the derivatives of the camera distances are unknown, so a noise term w_i can be substituted. The differential equation describing the EoM can be summarised as

$$\dot{\vec{x}} = \vec{f}\left(\vec{x}, \vec{u}\right) = \begin{bmatrix} A_x - qw + rv - g\sin\theta \\ A_y - ru + pw + g\sin\varphi\cos\theta \\ A_z - pv + qu + g\cos\varphi\cos\theta \\ \frac{\mathcal{M}_x I_z - qr{I_z}^2 + qrI_z\,I_y - J_{xz}\,pqI_y + J_{xz}\,pqI_x - J_{xz}^2 rq + J_{xz}\,\mathcal{M}_z + J_{xz}\,pqI_z}{-J_{xz}^2 + I_x\,I_z} \\ \frac{\mathcal{M}_y - (I_x - I_z)rp - J_{xz}\left(p^2 - r^2\right)}{I_y} \\ \frac{\mathcal{M}_z I_x - pqI_x\,I_y + pqI_x^2 - J_{xz}\,rqI_z + J_{xz}\,rqI_y - J_{xz}^2 pq + J_{xz}\,\mathcal{M}_x - J_{xz}\,rqI_x}{-J_{xz}^2 + I_z\,I_x} \\ q\cos\varphi - r\sin\varphi \\ p + q\sin\varphi\tan\theta + r\cos\varphi\tan\theta \\ w_1 \\ \vdots \\ w_6 \end{bmatrix} \tag{2-20}$$

This will be used in the observability analysis in Section 3-3-2 and the Kalman filter implementations used with the simulated and hardware generated data.

Chapter 3

Observability

3-1 LTI Definition

The observability properties of linear state space systems are a special case of those for nonlinear systems. Therefore, it is convenient to look at the well-understood case of linear systems, before defining the problem for nonlinear systems. (Olsder & Woude, 2005, p.73) give the following definition of observability for linear time invariant (LTI) systems:

$$(A, B, C, D): \quad \begin{array}{l} \dot{\vec{x}} = A\vec{x} + B\vec{u} \\ \vec{y} = C\vec{x} + D\vec{u} \end{array}$$

with $\vec{u} \in \mathbb{R}^m$, $\vec{x} \in \mathbb{R}^n$ and $\vec{y} \in \mathbb{R}^p$.

> ***Definition 3.1*** *An LTI system (A, B, C, D) is called observable if the initial state $\vec{x}(t_0)$ can be reconstructed from the knowledge of the input $\vec{u}$ and output $\vec{y}$ on the interval $[t_0, t_1]$ for any finite $t_1 > t_0$.*

Because $\vec{u}$ is given, once $\vec{x}(t_0)$ is known, the state $\vec{x}$ on the whole interval $[t_0, t_1]$ can be determined. Note that observability is a global concept, it holds for the whole state space. To test the observability of an LTI system, the observability matrix $\mathcal{O}$, given by

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{3-1}$$

is used.

> **Theorem 3.1** (A, B, C, D) is observable if $\operatorname{rank} \mathcal{O} = n$, where n is the dimension of the state vector $\vec{x}$.

3-2 Nonlinear Observability

The definition of observability for linear systems has to be adapted in order to extend it to the class of nonlinear state space systems:

$$\Sigma : \quad \begin{array}{c} \dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u}) \\ \vec{y} = \vec{h}(\vec{x}) \end{array} \tag{3-2}$$

where $\vec{u} \in L$, a subset of $\mathbb{R}^m$, $\vec{x} \in M$, the state space with dimension n, $\vec{y} \in \mathbb{R}^p$ and $\vec{f}$ and $\vec{h}$ are vector functions of appropriate dimensions.
Whereas in the linear case the observability condition holds for the entire domain of $\vec{x}$, for nonlinear systems observability is state dependent and has to be determined locally (Hedrick & Girard, 2005). First, the definition of *distinguishability* is required:

> ***Definition 3.2*** *Given Σ. Two states $\vec{x}_0$ and $\vec{x}_1$ are distinguishable if and only if there exists an input function $\vec{u}^*$ such that: $\vec{h}(\vec{x}_0) \neq \vec{h}(\vec{x}_1)$.*

Local observability can then be defined as follows:

> ***Definition 3.3*** *Σ is locally observable at $\vec{x}_0$ if and only if there exists a neighbourhood of $\vec{x}_0$ such that every $\vec{x}$ in that neighbourhood other than $\vec{x}_0$ is distinguishable from $\vec{x}_0$.*

To test whether a system is locally observable at $\vec{x}_0$, an observability matrix similar to the linear case can be constructed. (Hermann & Krener, 1977) describe this in terms of the Lie derivative:

> ***Definition 3.4*** *Let $\vec{f}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^n$ be a vector field in $\mathbb{R}^n$,*
> *let $g(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ be a scalar function differentiable up to degree $n-1$ and*
> *let the gradient of g with respect to $\vec{x}$ be $\frac{\partial g}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \dots & \frac{\partial g}{\partial x_n} \end{bmatrix}$.*
>
> *Then the Lie derivative of g with respect to $\vec{f}$ is: $L_f(g) = \frac{\partial g}{\partial \vec{x}} \vec{f}$.*
> *Higher order Lie derivatives are defined as $L_f^2(g) = \frac{\partial}{\partial \vec{x}} \left\{ L_f^1(g) \right\} \vec{f}$, ...*
> *$L_f^n(g) = \frac{\partial}{\partial \vec{x}} \left\{ L_f^{n-1}(g) \right\} \vec{f}$. By definition $L_f^0(g) = g$.*

Note that the Lie derivative is a scalar valued quantity.
The matrix to test for local observability of Σ at $\vec{x}_0$ for some constant control $\vec{u}^*$ is defined by

$$\mathcal{O}(\vec{x}_0, \vec{u}^*) \equiv \left[\begin{array}{c} dL_f^0(h_1) \\ \vdots \\ dL_f^0(h_p) \\ \vdots \\ dL_f^{n-1}(h_1) \\ \vdots \\ dL_f^{n-1}(h_p) \end{array} \right] \Bigg|_{\vec{x}=\vec{x}_0,\, \vec{u}=\vec{u}^*} \tag{3-3}$$

where d is the gradient operator with respect to x, $\vec{h}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^p$ and h_i is the i^{th} element of $\vec{h}$.
$\mathcal{O}(\vec{x}_0, \vec{u}^*)$ determines the local observability of Σ at $\vec{x}_0$. Σ is said to satisfy the observability rank condition at $\vec{x}_0$ if the dimension of the image of $\mathcal{O}(\vec{x}_0, \vec{u}^*)$ is n.

> **Theorem 3.2** If and only if Σ satisfies the observability rank condition at $\vec{x}_0$ then Σ is locally observable at $\vec{x}_0$.

The proof can be found in (Hermann & Krener, 1977).

(Walcott, Corless & Żak, 1987) gives the following elaboration of the gradient of the Lie derivative of g with respect to $\vec{f}$:

$$dL_f(g) = L_f(dg) = \left(\frac{\partial (dg)^T}{\partial \vec{x}} \vec{f}\right)^T + dg \frac{\partial \vec{f}}{\partial \vec{x}} = \vec{f}^T \frac{\partial (dg)^T}{\partial \vec{x}} + dg \frac{\partial \vec{f}}{\partial \vec{x}} \tag{3-4}$$

This may be used to expand the rows of Eq. (3-3), yielding an approximation to $\mathcal{O}(\vec{x}, \vec{u})$ using the Jacobians of $\vec{f}$ and $\vec{h}$:

Theorem 3.3 Let

$$F(\vec{x}, \vec{u}) = \frac{\partial \vec{f}(\vec{x}, \vec{u})}{\partial \vec{x}}$$
$$H(\vec{x}) = \frac{\partial \vec{h}(\vec{x})}{\partial \vec{x}}$$

then

$$\mathcal{O}(\vec{x}, \vec{u}) = \begin{bmatrix} H(\vec{x}) \\ H(\vec{x}) F(\vec{x}, \vec{u}) \\ H(\vec{x}) \{F(\vec{x}, \vec{u})\}^2 \\ \vdots \\ H(\vec{x}) \{F(\vec{x}, \vec{u})\}^{n-1} \end{bmatrix} + \begin{bmatrix} \varnothing^{p \times n} \\ E(\vec{x}, \vec{u}) \\ E(\vec{x}, \vec{u}) F(\vec{x}, \vec{u}) \\ \vdots \\ E(\vec{x}, \vec{u}) \{F(\vec{x}, \vec{u})\}^{n-2} \end{bmatrix} + \begin{matrix} higher \\ order \\ terms \end{matrix} \tag{3-5}$$

where

$$E(\vec{x}, \vec{u}) = \begin{bmatrix} \vec{f}^T \frac{\partial (dh_1)^T}{\partial \vec{x}} \\ \vdots \\ \vec{f}^T \frac{\partial (dh_p)^T}{\partial \vec{x}} \end{bmatrix}$$

Proof of theorem 3.3. Using $d\left(\vec{a} \cdot \vec{b}\right) = \vec{a}^T \frac{\partial \vec{b}}{\partial \vec{x}} + \vec{b}^T \frac{\partial \vec{a}}{\partial \vec{x}}$, an arbitrary row of $\mathcal{O}(\vec{x}, \vec{u})$ can be written as

$$\begin{aligned} & dL_f^k(h_i) = d\left[\tfrac{\partial}{\partial \vec{x}} \left\{L_f^{k-1}(h_i)\right\} \vec{f}\right] = dL_f^{k-1}(h_i) \tfrac{\partial \vec{f}}{\partial \vec{x}} + \vec{f}^T \tfrac{\partial \left(dL_f^{k-1}(h_i)\right)^T}{\partial \vec{x}} \\ & = d\left[\tfrac{\partial}{\partial \vec{x}} \left\{L_f^{k-2}(h_i)\right\} \vec{f}\right] \tfrac{\partial \vec{f}}{\partial \vec{x}} + higher\ order\ terms \\ & = dL_f^{k-2}(h_i) \left(\tfrac{\partial \vec{f}}{\partial \vec{x}}\right)^2 + \vec{f}^T \tfrac{\partial \left(dL_f^{k-2}(h_i)\right)^T}{\partial \vec{x}} \tfrac{\partial \vec{f}}{\partial \vec{x}} + higher\ order\ terms \\ & \vdots \\ & = dh_i \left(\tfrac{\partial \vec{f}}{\partial \vec{x}}\right)^k + \vec{f}^T \tfrac{\partial (dh_i)^T}{\partial \vec{x}} \left(\tfrac{\partial \vec{f}}{\partial \vec{x}}\right)^{k-1} + higher\ order\ terms \end{aligned} \tag{3-6}$$

where $k = 1 \dots n-1$ and $i = 1 \dots p$. For $k = 0$, definition 3.4 combined with Eq. (3-4) yields $dL_f^0(h_i) = dh_i$.
The first term in the last expression in Eq. (3-6) is equal to the i^{th} row in $H(\vec{x},\vec{u})\{F(\vec{x},\vec{u})\}^k$ and the second term in the last expression is equal to the i^{th} row in $E(\vec{x},\vec{u})\{F(\vec{x},\vec{u})\}^{k-1}$ □

Returning to the observability rank condition for linear systems for a moment, the following proof shows that it follows from theorem 3.2 as a special case. This approaches the problem from another direction than the usual proof based on Cayley-Hamilton, as can be found in (Olsder & Woude, 2005).

Proof of theorem 3.1. Substitute $\vec{f}(\vec{x},\vec{u}) = A\vec{x} + B\vec{u}$ and $\vec{h}(\vec{x}) = C\vec{x} + D\vec{u}$ in Σ. The jacobians with respect to $\vec{x}$ are: $H(\vec{x}) = C$ and $F(\vec{x},\vec{u}) = A$. The matrix $E(\vec{x},\vec{u}) = \varnothing^{p\times n}$ and all other higher order terms are zero as well, because they all involve second and higher order derivatives of f and h which are equal to zero in this case. Substituting this result in Eq. (3-5) yields the local observability matrix for a linear system (A, B, C, D):

$$\mathcal{O}(\vec{x},\vec{u}) = \mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{3-7}$$

Because Eq. (3-7) doesn't depend on $\vec{x}$ or $\vec{u}$, it follows from theorem 3.2 that the system is observable on the entire state space if $\operatorname{rank}\mathcal{O} = n$. □

3-3 Observability of the Optical Flow Sensor System

This section describes an observability analysis of the optical flow (OF) sensor system based on the first term of Eq. (3-5) for a 2D case and the full 3D problem, both with accelerometers. This first term of $\mathcal{O}(\vec{x},\vec{u})$, henceforth denoted as $\mathcal{O}^1(\vec{x},\vec{u})$, is used often in practice to check for local observability as the full definition (Eq. (3-3)) in terms of Lie derivatives tends to become very complex. It has to be computed analytically and the Lie derivatives expand dramatically as each higher order is the dot product of $\vec{f}$ and the gradient of the previous order.
Moreover, it is my hypothesis that the rank of $\mathcal{O}^1(\vec{x},\vec{u})$ is a lower bound for the rank of $\mathcal{O}(\vec{x},\vec{u})$. That is, we have:

Conjecture For any $x_0 \in M$ and some constant permissible input u^*, it holds that

$$\operatorname{rank}\begin{bmatrix} H(\vec{x}_0) \\ H(\vec{x}_0)F(\vec{x}_0,\vec{u}^*) \\ H(\vec{x}_0)\{F(\vec{x}_0,\vec{u}^*)\}^2 \\ \vdots \\ H(\vec{x}_0)\{F(\vec{x}_0,\vec{u}^*)\}^{n-1} \end{bmatrix} \leq \operatorname{rank}\left.\begin{bmatrix} dL_f^0(h_1) \\ \vdots \\ dL_f^0(h_p) \\ \vdots \\ dL_f^{n-1}(h_1) \\ \vdots \\ dL_f^{n-1}(h_p) \end{bmatrix}\right|_{\vec{x}=\vec{x}_0,\,\vec{u}=\vec{u}^*} \tag{3-8}$$

This would make rank $\mathcal{O}^1(\vec{x},\vec{u})$ a conservative estimate, which is an important condition for its use as it would ensure that a system is locally observable if $\mathcal{O}^1(\vec{x},\vec{u})$ has full rank.
A problem arises when comparing rank $\mathcal{O}(\vec{x},\vec{u})$ applied to the 2D study case with physical arguments. It can be interpreted as an additional argument to use $\mathcal{O}^1(\vec{x},\vec{u})$ instead of $\mathcal{O}(\vec{x},\vec{u})$. This is discussed at the end of Section 3-3-1.

3-3-1 A Two Dimensional Study Case

The observability matrix of the full three dimensional problem is very complex as it depends on 14 states, has 210 rows, involves the matrix product of F to the 13th power and has 14 terms in the form of Eq. (3-5). Therefore, in order to simplify the analytical investigation of the local observability of the optical flow sensor system, one may look at a two dimensional problem resembling the three dimensional optical flow sensor system.
Figure 3-1 shows the geometry and state variables of this 2D study case. There are two velocity components, u and v, defined in the body reference frame (X_{body}, Y_{body}). There is one rotational rate r. There is one attitude angle θ. And there are four distances $d_1 - d_4$ associated with the cameras. These are the states in the vector $\vec{x}$ that must be observed.
To do so, there are four optical flow cameras mounted along the body axes. Each has only one optical flow output $\Omega_1 - \Omega_4$, unlike in the 3D case, where each sensor sees a surface with two optical flow components. Furthermore, there is an accelerometer, measuring specific forces A_x and A_y in body axes directions. Those are the measurements contained in vector $\vec{y}$. The only element in the input $\vec{u}$ is the moment $\mathcal{M}_z$ with direction normal to the plane of motion. Finally, there is a uniform gravitational field with acceleration g in the negative Y_{earth} direction.

The equations of motion of this system can be derived from the full 3D equations, Eq. (2-20), by elimination of terms which become zero because they involve out of plane motion. Kinematics simplify to $\dot{\theta} = r$ and the derivatives of the sensor-wall distances are unknown, so a noise term w_i is used. This yields:

$$\dot{\vec{x}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \\ \dot{\theta} \\ \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \\ \dot{d}_4 \end{bmatrix} = \vec{f}(\vec{x},\vec{u}) = \begin{bmatrix} A_x + rv - g\sin\theta \\ A_y - ru - g\cos\theta \\ \frac{\mathcal{M}_z}{I_z} \\ r \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{3-9}$$

The measurement equations describing $\vec{y}$ can also be derived from the 3D case, 3-14, in a similar way:

$$\vec{y} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \\ A_x \\ A_y \end{bmatrix} = \vec{h}(\vec{x}) = \begin{bmatrix} \frac{u}{d_1} - r \\ \frac{v}{d_2} - r \\ \frac{u}{d_3} + r \\ \frac{v}{d_4} + r \\ \dot{u} - rv + g\sin\theta \\ \dot{v} + ru + g\cos\theta \end{bmatrix} \tag{3-10}$$

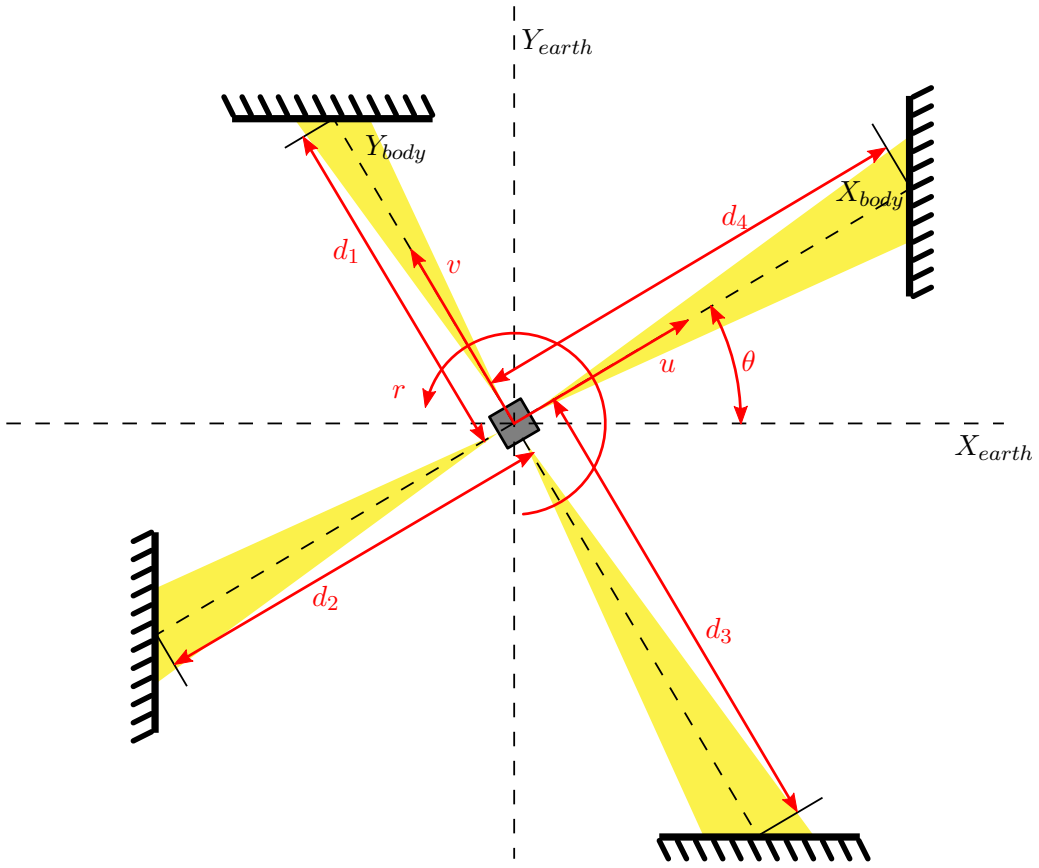


Figure 3-1: Diagram of the 2D problem with state variables

To construct the local observability matrix using Eq. (3-5), the Jacobians of $\vec{f}(\vec{x},\vec{u})$ and $\vec{h}(\vec{x})$ are required. These have been computed using Matlab Symbolic Toolbox:

$$F(\vec{x},\vec{u}) = \frac{\partial}{\partial \vec{x}}\vec{f}(\vec{x},\vec{u}) = \begin{bmatrix} 0 & r & v & -g\cos\theta & 0 & 0 & 0 & 0 \\ -r & 0 & -u & g\sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-11}$$

$$H(\vec{x}) = \frac{\partial}{\partial \vec{x}}\vec{h}(\vec{x}) = \begin{bmatrix} \frac{1}{d_1} & 0 & -1 & 0 & -\frac{u}{d_1^2} & 0 & 0 & 0 \\ 0 & \frac{1}{d_2} & -1 & 0 & 0 & -\frac{v}{d_2^2} & 0 & 0 \\ \frac{1}{d_3} & 0 & 1 & 0 & 0 & 0 & -\frac{u}{d_3^2} & 0 \\ 0 & \frac{1}{d_4} & 1 & 0 & 0 & 0 & 0 & -\frac{v}{d_4^2} \\ 0 & -r & -v & g\cos\theta & 0 & 0 & 0 & 0 \\ r & 0 & u & -g\sin\theta & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-12}$$

Before attempting a state observer including $\vec{f}(\vec{x},\vec{u})$ to solve for the state vector, one may try to solve $\vec{h}(\vec{x})$ directly with a nonlinear solver. However, in this case the observations alone

can never provide full observability as there are less equations then unknowns to solve for $(n = 8)$. Indeed, for nonzero values of $\vec{x}$, $\operatorname{rank}(H(\vec{x})) = 6$. If one of either u, v or r is zero then the rank reduces to 5. If both u and v are zero the rank is equal to 4. The distances are assumed to have nonzero positive values. A distance may become practically ∞ if the sensor is looking at open sky for example. This reduces the rank by 1 if at least one other state is zero. θ may reduce the rank of $H(\vec{x})$ when one or more distances are ∞.
These findings have been calculated by taking a grid of sample points across the state space and substituting these numerical values in the symbolic Jacobian $H(\vec{x})$. This is not a definitive method as certain areas of interest may be very small and may not get sampled. However, by choosing the grid such as to include points that can be expected to produce lower rank, such as values which cause terms to cancel out, one can still gain useful insight into the local observability behaviour.
The next step would be to compute the local observability matrix and repeat the sampling method described above to investigate the local observability behaviour of the system including dynamics. The symbolic observability matrix $\mathcal{O}^1(\vec{x}, \vec{u})$ calculated using the symbolic toolbox from Matlab is too large to be printed in this report. The maximum of $\operatorname{rank}\mathcal{O}^1(\vec{x}, \vec{u}) = 7$. If this would be equal to the rank of the full observability matrix, it would mean that the "two dimensional version"of the optical flow sensor system will not work, regardless which filtering method is applied as there is too little data available to reconstruct the state $\vec{x}$. However, when applying the full observability matrix rank condition, it yields full rank. Note that this does not falsify the conjecture on page 32 as $\operatorname{rank}\mathcal{O}^1(\vec{x}, \vec{u}) < \operatorname{rank}\mathcal{O}(\vec{x}, \vec{u})$.
When evaluating $\operatorname{rank}\mathcal{O}(\vec{x}, \vec{u})$ across the state space, a problem appears. The rank of $\mathcal{O}$ is equal to 8 for all $\vec{x}$, except when one or more distances go to ∞. This means that, according to the full local observability rank condition, one would be able to estimate the distances $d_1 - d_4$ even when there is no motion, i.e. when $u = v = r = 0$. This is impossible from a physical point of view. One might be able to establish that there is no movement relative to the walls, but this gives no information about the distances. The distances appear in the optical flow components in Eq. (3-10) only in a quotient with the translations u and v. So if all quotients are zero, nothing can be inferred about $d_1 - d_4$. Formally, the system is indistinguishable (definition 3.2) at $\vec{x} = \begin{bmatrix} 0 & 0 & 0 & \theta & d_1 & d_2 & d_3 & d_4 \end{bmatrix}^T$ as there is a set of states producing the same output regardless of the input $\mathcal{M}_z$. It follows from definition 3.3 that the system is locally unobservable at $\vec{x} = \begin{bmatrix} 0 & 0 & 0 & \theta & d_1 & d_2 & d_3 & d_4 \end{bmatrix}^T$. This case appears to falsify the observability rank test using $\mathcal{O}$ (theorem 3.2) as it produces full rank at that point in the state space.
Unfortunately, literature has not clarified how this discrepancy may be explained. As $\operatorname{rank}\mathcal{O}^1(\vec{x}, \vec{u})$ appears to produce the correct results, it will be used in this thesis.

3-3-2 Local Observability Analysis of the Full Three Dimensional Problem

The analysis in Section 3-3-1 can also be applied to the full three dimensional problem. Referring to Section 2-3-3, Eq. (3-13) and Eq. (3-14) give a summary of the 3D problem with the sensors package placed in the center of gravity (CoG) and aligned with the body frame of reference:

$$\vec{x} = \begin{bmatrix} u & v & w & p & q & r & \theta & \varphi & d_1 & \dots & d_6 \end{bmatrix}^T$$

$$\vec{u} = \begin{bmatrix} \mathcal{M}_x & \mathcal{M}_y & \mathcal{M}_z \end{bmatrix}^T$$

$$\vec{z} = \begin{bmatrix} \Omega_{X_1} & \Omega_{Z_1} & \Omega_{Y_2} & \Omega_{Z_2} & \Omega_{X_3} & \Omega_{Z_3} & \Omega_{Y_4} & \Omega_{Z_4} & \Omega_{X_5} & \Omega_{Y_5} & \Omega_{X_6} & \Omega_{Y_6} & A_x & A_y & A_z \end{bmatrix}^T$$

$$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u}) = \begin{bmatrix} A_x - qw + rv - g\sin\theta \\ A_y - ru + pw + g\sin\varphi\cos\theta \\ A_z - pv + qu + g\cos\varphi\cos\theta \\ \frac{\mathcal{M}_x I_z - qr{I_z}^2 + qrI_z I_y - J_{xz}\,pqI_y + J_{xz}\,pqI_x - J_{xz}^2 rq + J_{xz}\,\mathcal{M}_z + J_{xz}\,pqI_z}{-J_{xz}^2 + I_x I_z} \\ \frac{\mathcal{M}_y - (I_x - I_z)rp - J_{xz}\left(p^2 - r^2\right)}{I_y} \\ \frac{\mathcal{M}_z I_x - pqI_x I_y + pqI_x^2 - J_{xz}\,rqI_z + J_{xz}\,rqI_y - J_{xz}^2 pq + J_{xz}\,\mathcal{M}_x - J_{xz}\,rqI_x}{-J_{xz}^2 + I_z I_x} \\ q\cos\varphi - r\sin\varphi \\ p + q\sin\varphi\tan\theta + r\cos\varphi\tan\theta \\ w_1 \\ \vdots \\ w_6 \end{bmatrix} \tag{3-13}$$

$$\vec{z} = \vec{h}_{aligned}(\vec{x}) = \begin{bmatrix} \frac{u}{d_1} - r \\ \frac{w}{d_1} + p \\ \frac{v}{d_2} - r \\ \frac{w}{d_2} + q \\ \frac{u}{d_3} + r \\ \frac{w}{d_3} - p \\ \frac{v}{d_4} + r \\ \frac{w}{d_4} - q \\ \frac{u}{d_5} - q \\ \frac{v}{d_5} + p \\ \frac{u}{d_6} + q \\ \frac{v}{d_6} - p \\ \dot{u} + qw - rv + g\sin\theta \\ \dot{v} + ru - pw - g\sin\varphi\cos\theta \\ \dot{w} + pv - qu - g\cos\varphi\cos\theta \end{bmatrix} \tag{3-14}$$

Again, using Matlab Symbolic Toolbox, jacobians $F(\vec{x},\vec{u})$ and $H_{aligned}(\vec{x})$ have been computed. $F(\vec{x},\vec{u})$ doesn't fit on a page, but $H_{aligned}(\vec{x})$ gives:

$H_{aligned}(\vec{x}) = \frac{\partial}{\partial \vec{x}}\vec{h}(\vec{x}) =$

$$\begin{bmatrix}
\frac{1}{d_1} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -\frac{u}{d_1^2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{d_1} & 1 & 0 & 0 & 0 & 0 & -\frac{w}{d_1^2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{d_2} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -\frac{v}{d_2^2} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{d_2} & 0 & 1 & 0 & 0 & 0 & 0 & -\frac{w}{d_2^2} & 0 & 0 & 0 & 0 \\
\frac{1}{d_3} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -\frac{u}{d_3^2} & 0 & 0 & 0 \\
0 & 0 & \frac{1}{d_3} & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{w}{d_3^2} & 0 & 0 & 0 \\
0 & \frac{1}{d_4} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -\frac{v}{d_4^2} & 0 & 0 \\
0 & 0 & \frac{1}{d_4} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{w}{d_4^2} & 0 & 0 \\
\frac{1}{d_5} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{u}{d_5^2} & 0 \\
0 & \frac{1}{d_5} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{v}{d_5^2} & 0 \\
\frac{1}{d_6} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{u}{d_6^2} \\
0 & \frac{1}{d_6} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{v}{d_6^2} \\
0 & -r & q & 0 & w & -v & g\cos\theta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r & 0 & -p & -w & 0 & u & g\sin\varphi\sin\theta & -g\cos\varphi\cos\theta & 0 & 0 & 0 & 0 & 0 & 0 \\
-q & p & 0 & v & -u & 0 & g\cos\varphi\sin\theta & g\sin\varphi\cos\theta & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The sensor package may also be positioned at an offset from the CoG and with an arbitrary orientation as explained in Section 2-3-3. This results in modified observation equations and has consequences for the observability as explained below.

The method of sampling the state space used in Section 3-3-1 to investigate the local observability is repeated here for select values of $\vec{x}$ with a focus on zeros. Zero values appear to be the only cases where the observability rank drops, with the exception of the attitude angles which cause vanishing terms at some other values as well. Table 3-1 lists the findings for the case where the sensor package is aligned with the body frame and positioned in the CoG and for the case with a fixed rotation and displacement of the sensor package. Both the rank of the Jacobian of the observations, $H(x)$, and the local observability matrix $\mathcal{O}^1(x,u)$ is listed. The first conclusion from Table 3-1 is that, in general, motion is required to observe all the states. At least one component of both $\mathbf{V}$ and $\vec{\omega}$ should be nonzero in the case of the rotated sensor package. The fact that the aligned sensor package has lower observability rank in some cases can be explained by considering that the alignment means that some of the measurements are, at least in theory, exactly zero. This means that the rotated sensor package also has critical directions where the observability rank drops, but those directions aren't tested in this table.
Furthermore, the Jacobian of the measurement equations $H(\vec{x})$ already gives full rank in most cases. So it should be possible to solve for $\vec{x}$ with a nonlinear solver, using only $h(\vec{x})$. However, nonlinear solvers are not adapted to stochastic processes and will not produce a minimum variance solution as does the Kalman filter. Also, incorporating knowledge of the system inertia and dynamics should produce more accurate results. Results in support of

Table 3-1: Local observability

number of zero states			rank condition aligned case		rank condition rotated case	
V	$\vec{\omega}$	θ & φ	rank H	rank $\mathcal{O}^1$	rank H	rank $\mathcal{O}^1$
0	0	0	14	14	14	14
0	0	2	14	14	14	14
0	1	0	14	14	14	14
0	2	0	14	14	14	14
0	3	0	13	13	13	13
1	0	0	13	14	14	14
1	1	0	13	14	14	14
1	2	0	13	14	14	14
1	3	0	13	13	13	13
2	0	0	12	12	14	14
2	0	2	11	12	13	14
2	1	0	12	12	14	14
2	1	1	12	12	14	14
2	1	2	11	11	13	14
2	2	0	12	12	14	14
2	2	2	11	11	13	13
2	3	0	11	11	13	13
3	0	0	8	8	8	8
3	3	0	8	8	8	8
3	3	2	8	8	8	8

these assumptions have been obtained from attempts to solve Eq. (3-14) directly using several nonlinear solver algorithms available in Matlab. The "best" result was achieved using `fminunc` with the jacobian as given above. The solution took several hours to compute on a 2+ GHz Core 2 Duo machine for only 30 seconds of data. Moreover the solution was very inaccurate compared to the Kalman filter performance.
A note about the use of the accelerometer data in the equations of motion. Ideally, the equations for $\dot{u}$, $\dot{v}$ and $\dot{w}$ should include bias values for A_x, A_y and A_z respectively. These are then extra states with a derivative equal to zero. The current sensor configuration does however not support this, because the problem including bias states is not observable. The addition of 3 biases brings the state vector to a length of 17 elements. The highest value of rank $\mathcal{O}^1$ = 16. This is found by substituting into $\mathcal{O}^1$ a large set of random numerical values from the domain of the state space.
To alleviate this problem, the accelerometer has been calibrated prior to each series of experiment runs as described in Section 8-1.

3-3-3 Two Quantities to Gauge the Observability Condition

To gauge how well a system is observable, the observability condition number $\mathcal{C}_O$ may be used, (Chen, 1991; Driels & Pathre, 1990). It is derived from the singular value decomposition of the observability matrix. For nonlinear, time-invariant systems Σ, this will depend on $\vec{x}$ and

$\vec{u}$, i.e. $\mathcal{C}_O(\vec{x},\vec{u})$ is a local quantity for nonlinear systems.

> ***Definition 3.5*** *Let a nonlinear state space system Σ be given as in Eq. (3-2) and its local observability matrix $\mathcal{O}(\vec{x},\vec{u})$ as in Eq. (3-3). Then the observability condition number $\mathcal{C}_O$ is defined as the ratio between the maximum and minimum nonzero singular values of $\mathcal{O}(\vec{x},\vec{u})$. So,*
>
> $$\mathcal{O}(\vec{x},\vec{u}) = USV^T$$
>
> *where $U \in \mathbb{R}^{np\times np}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and $S \in \mathbb{R}^{np\times n}$ has its only nonzero elements along the diagonal. These elements σ_i are ordered such that*
>
> $$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0, \quad \sigma_r > \sigma_{r+1} = 0$$
>
> *where $r = \text{rank}(\mathcal{O}(\vec{x},\vec{u}))$. Then $\mathcal{C}_O = \frac{\sigma_1}{\sigma_r}$.*

A closely related measure, proposed by (Hong, Chun, Kwon & Lee, 2008), indicates the smallest perturbation in $\mathcal{O}$ which makes $\mathcal{O}$ rank deficient:

> ***Definition 3.6*** *Let $\mathcal{O}, \Delta \in \mathbb{R}^{np\times n}$, then*
>
> $$\underline{\mu}(\mathcal{O}) \equiv \min_{\text{rank}(\mathcal{O}-\Delta)<n} \|\Delta\|_2$$

The following theorem provides a means to compute $\underline{\mu}(\mathcal{O})$:

> **Theorem 3.4** (Golub & Loan, 1996, Th. 2.5.3) Let $\mathcal{O}, \Delta \in \mathbb{R}^{np\times n}$ and σ_i as in definition 3.5 then,
>
> $$\underline{\mu}(\mathcal{O}) = \sigma_n$$

Note that these two observability condition measures may also be based on $\mathcal{O}^1(\vec{x},\vec{u})$ under the same conditions as described at the end of Section 3-3-1. $\mathcal{C}_O$ and $\underline{\mu}(\mathcal{O})$ will be used to analyse the observability condition development over time during simulation in the final thesis report.

Chapter 4

Kalman Filters

4-1 Basic Kalman filter derivation

This section describes the derivation of the Kalman filter problem for linear systems. Although the application to this work requires another type suited for nonlinear systems, the Conventional Kalman filter problem still forms the basis for these as well. Its derivation clearly shows the purpose of the Kalman filter as a zero bias, minimum variance estimator based on the least squares cost function.

4-1-1 The state observer

The Kalman filter is part of the class of filters known as *observers* that reconstruct missing information, such as part of the state vector, from measured output and input of a state-space model.
To start the derivation of the Kalman filter problem, we start out with a deterministic linear time invariant (LTI) model.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{4-1}$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, $y(k) \in \mathbb{R}^l$ is the output vector and $k \in \mathbb{Z}$ is the discrete-time sequence index.

Consider the approximation $\hat{x}(k)$ of $x(k)$ for $k \geq 0$ with $x(0)$ unknown. Using only $u(k)$ with initial state $\hat{x}(0) \neq x(0)$, we have

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k)$$

and the error $\varepsilon_x(k) = \hat{x}(k) - x(k)$ will diminish if and only if A is asymptotically stable. This becomes apparent considering that

$$\hat{x}(k+1) - x(k+1) = A\hat{x}(k) + Bu(k) - Ax(k) - Bu(k)$$

$$\Rightarrow \varepsilon_x(k+1) = A\varepsilon_x(k)$$

However, using only $u(k)$ one has no control over the rate at which $\hat{x}(k)$ converges to $x(k)$. The estimate can be improved by adding a correction term based on the difference between the measured output $y(k)$ and the estimated output $\hat{y}(k) = C\hat{x}(k) + Du(k)$:

$$K(y(k) - \hat{y}(k))$$

where K is a gain matrix. This results in

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K(y(k) - C\hat{x}(k) - Du(k)) \tag{4-2}$$

K should be chosen such that the error $\varepsilon_x(k)$ goes to zero for $k \to \infty$. The state error propagation equation satisfies

$$\varepsilon_x(k+1) = (A - KC)\varepsilon_x(k)$$

from which it follows that K should make $A - KC$ asymptotically stable. For the existence of K the following condition holds:

> **Lemma 4.1** (Kailath, 1968) *Given matrices $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{l\times n}$, if the pair (A, C) is observable, then there exists a matrix $K \in \mathbb{R}^{n\times l}$ such that $A-KC$ is asymptotically stable.*

4-1-2 the Conventional Kalman filter

The conventional Kalman filter originally described by (Kalman, 1960) is applicable to the discrete-time LTI system corrupted with process and measurement noise vectors $w(k)$ and $v(k)$, respectively:

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{4-3}$$
$$z(k) = Cx(k) + v(k) \tag{4-4}$$

where the direct feed-through term $Du(k)$ has been omitted. This could be included as well, but for a lot of problems the term is not needed. The logical step now is to apply the asymptotical observer from Eq. (4-2). This yields

$$\varepsilon_x(k+1) = (A - KC)\varepsilon_x(k) + Kv(k) - w(k)$$

for the error in the estimated state. However, the condition that the pair (A, C) must be observable is not sufficient to obtain a minimal error ε_x, in contrast to the deterministic case where the estimate will converge asymptotically to the true state if this condition holds. In addition, we need to minimise the influence of $w(k)$ and $v(k)$ on the state estimate $\hat{x}(k)$. An important assumption here is that $w(k)$ and $v(k)$ are zero mean white-noise sequences, i.e. they have a normal distribution with a joint covariance matrix

$$E\left[\begin{bmatrix} v(k) \\ w(k) \end{bmatrix} \begin{bmatrix} v(j)^T & w(j)^T \end{bmatrix}\right] = \begin{bmatrix} R(k) & S(k)^T \\ S(k) & Q(k) \end{bmatrix} \Delta(k-j) \geq 0$$

If the assumption that $w(k)$ and $v(k)$ are uncorrelated holds, than $S(k)$ is equal to the zero matrix of appropriate dimensions.

To derive the estimated state at time step k, from time $k-1$ the predicted state estimate $\hat{x}(k|k-1)$ at time k is available, and furthermore $u(k)$ and $z(k)$. Since $\varepsilon_x(k)$ is a stochastic signal, we can require that its mean should equal zero. Secondly, in order to make the fluctuations of ε_x as small as possible, the second order moment of the probability density function of the error must be minimised. These two requirements mean that the desired estimate $\hat{x}(k|k)$ is unbiased

$$E[x(k)] = E[\hat{x}(k|k)] \tag{4-5}$$

and the covariance matrix $P(k|k)$ is minimal

$$\min P(k|k) = \min E\left[(x(k) - \hat{x}(k|k))(x(k) - \hat{x}(k|k))^T\right] \tag{4-6}$$

For the prediction to time step $k+1$, similar conditions hold. Note that the Kalman filter can still be applied when the problem contains noises with other distributions than Gaussian, but the resulting estimate will no longer satisfy the minimum variance condition, Eq. (4-6). To derive expressions for $\hat{x}(k|k)$, $P(k|k)$, $\hat{x}(k+1|k)$ and $P(k+1|k)$, first rewrite Eq. (4-3) and Eq. (4-4) as

$$\begin{bmatrix} z(k) \\ -Bu(k) \end{bmatrix} = \begin{bmatrix} C & 0 \\ A & -I_n \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} + L(k)\nu(k) \tag{4-7}$$

with $L(k)$ from

$$\begin{bmatrix} R(k) & S(k)^T \\ S(k) & Q(k) \end{bmatrix} = L(k)L(k)^T$$

and $\nu(k)$ an auxiliary variable representing the noise sequences. The problem can be written in one equation where matrix M must be found to satisfy the conditions Eq. (4-5) and Eq. (4-6):

$$\begin{bmatrix} \hat{x}(k|k) \\ \hat{x}(k+1|k) \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \end{bmatrix} \begin{bmatrix} z(k) \\ -Bu(k) \\ \hat{x}(k|k-1) \end{bmatrix} \tag{4-8}$$

Substitution of Eq. (4-7) into Eq. (4-8) yields

$$\begin{aligned} \begin{bmatrix} \hat{x}(k|k) \\ \hat{x}(k+1|k) \end{bmatrix} = \begin{bmatrix} M_{11}C + M_{12}A & -M_{12} \\ M_{21}C + M_{22}A & -M_{22} \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} + \\ \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} L(k)\nu(k) + \begin{bmatrix} M_{13} \\ M_{23} \end{bmatrix} \hat{x}(k|k-1) \end{aligned} \tag{4-9}$$

To apply the unbiasedness condition, we need to evaluate the mean

$$E[\hat{x}(k|k)] = (M_{11}C + M_{12}A)E[x(k)] - M_{12}E[x(k+1)] + M_{13}E[\hat{x}(k|k-1)],$$

$$E[\hat{x}(k+1|k)] = (M_{21}C + M_{22}A)E[x(k)] - M_{22}E[x(k+1)] + M_{23}E[\hat{x}(k|k-1)].$$

Both equations satisfy the unbiasedness condition in Eq. (4-5) provided that

$$\begin{aligned} M_{12} = 0, \quad M_{13} = I_n - M_{11}C, \\ M_{22} = -I_n, \quad M_{23} = A - M_{21}C \end{aligned}$$

After substitution of these results into Eq. (4-9) and reordering, we obtain

$$\begin{bmatrix} x(k) - \hat{x}(k|k) \\ x(k+1) - \hat{x}(k+1|k) \end{bmatrix} = \begin{bmatrix} I_n - M_{11}C \\ A - M_{21}C \end{bmatrix} (x(k) - \hat{x}(k|k-1)) - \begin{bmatrix} M_{11} & 0 \\ M_{21} & -I_n \end{bmatrix} L(k)\,\nu(k)$$

This leaves M_{11} and M_{21} to be determined. The covariance matrices that are to be minimised can be found by noting that $\nu(k)$ and $(x(k) - \hat{x}(k|k-1))$ are uncorrelated, as follows

$$\begin{aligned} P(k|k) &= E\left[(x(k) - \hat{x}(k|k))(x(k) - \hat{x}(k|k))^T\right] \\ &= P(k|k-1) - M_{11}CP(k|k-1) - P(k|k-1)C^T M_{11}^T \\ &+ M_{11}\left(CP(k|k-1)C^T + R(k)\right)M_{11}^T \end{aligned} \tag{4-10}$$

and

$$\begin{aligned} P(k+1|k) &= E\left[(x(k+1) - \hat{x}(k+1|k))(x(k+1) - \hat{x}(k+1|k))^T\right] \\ &= AP(k|k-1)A^T + Q(k) - M_{21}\left(CP(k|k-1)A^T + S(k)\right) \\ &- \left(AP(k|k-1)C^T + S(k)\right)M_{21}^T + M_{21}\left(CP(k|k-1)C^T + R(k)\right)M_{21}^T \end{aligned} \tag{4-11}$$

To apply the completion-of-squares argument from (Kailath, Sayed & Hassibi, 2000), Eq. (4-10) can also be written as

$$P(k|k) = (I_n - M_{11}C)P(k|k-1)(I_n - M_{11}C)^T + M_{11}R(k)M_{11}^T$$

$$= \begin{bmatrix} I_n & -M_{11} \end{bmatrix} \underbrace{\begin{bmatrix} P(k|k-1) & P(k|k-1)C^T \\ CP(k|k-1) & CP(k|k-1)C^T + R(k) \end{bmatrix}}_{N} \begin{bmatrix} I_n \\ -M_{11}^T \end{bmatrix} \tag{4-12}$$

The underbraced matrix N can be factorised using the Schur complement (Verhaegen & Verdult, 2007, p.19), where P is written in place of $P(k|k-1)$ for brevity:

$$\begin{aligned} N = & \begin{bmatrix} I_n & PC^T\left(CPC^T + R(k)\right)^{-1} \\ 0 & I_m \end{bmatrix} \\ & \times \begin{bmatrix} P - PC^T\left(CPC^T + R(k)\right)^{-1}CP & 0 \\ 0 & CPC^T + R(k) \end{bmatrix} \\ & \times \begin{bmatrix} I_n & 0 \\ \left(CPC^T + R(k)\right)^{-1}CP & I_m \end{bmatrix} \end{aligned}$$

Substituting this factorisation into Eq. (4-12) yields

$$\begin{aligned} P(k|k) = \; & P(k|k-1) - P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1}CP(k|k-1) \\ & + \left(P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1} - M_{11}\right)\left(CP(k|k-1)C^T + R(k)\right) \\ & \times \left(\left(CP(k|k-1)C^T + R(k)\right)^{-1}CP(k|k-1) - M_{11}^T\right) \end{aligned}$$

Since the covariance matrix $R(k)$ of the measurement noise $v(k)$ and a quadratic form are both positive-definite, the term $CP(k|k-1)C^T + R(k)$ is also positive definite. Therefore the third term on the righthand side has a positive definite contribution to the covariance

$P(k|k)$. Note that the first two terms on the righthand side do not depend on M_{11}. Therefore the minimum occurs when the last term vanishes. This occurs when

$$M_{11} = P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1}$$

A similar derivation on $P(k+1|k)$ yields

$$M_{21} = \left(AP(k|k-1)C^T + S\right)\left(CP(k|k-1)C^T + R(k)\right)^{-1}$$

With all sub-matrices of M determined, substitution into the linear estimate equation Eq. (4-8) gives the solution to the conventional Kalman filter problem for the unbiased, minimum variance state estimate at time k

$$\begin{aligned}\hat{x}(k|k) = \; & P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1} z(k) \\ & + \left(I_n - P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1} C\right)\hat{x}(k|k-1)\end{aligned} \tag{4-13}$$

with covariance matrix

$$P(k|k) = P(k|k-1) - P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1} CP(k|k-1) \tag{4-14}$$

and the one step ahead estimate

$$\begin{aligned}\hat{x}(k+1|k) = \; & \left(AP(k|k-1)C^T + S(k)\right)\left(CP(k|k-1)C^T + R(k)\right)^{-1} z(k) \\ & + Bu(k) + \left(A - \left(AP(k|k-1)C^T + S(k)\right)\right. \\ & \left.\times\left(CP(k|k-1)C^T + R(k)\right)^{-1} C\right)\hat{x}(k|k-1)\end{aligned} \tag{4-15}$$

with covariance matrix

$$\begin{aligned}P(k+1|k) = \; & AP(k|k-1)A^T + Q(k) - \left(AP(k|k-1)C^T + S(k)\right) \\ & \times\left(CP(k|k-1)C^T + R(k)\right)^{-1}\left(CP(k|k-1)A^T + S(k)^T\right)\end{aligned} \tag{4-16}$$

The common notation for $\hat{x}(k|k)$ and $P(k|k)$ contains a matrix $K(k)$ called the Kalman gain. The expression in Eq. (4-13) can be written in terms of $K(k)$ as follows:

$$K(k) = P(k|k-1)C^T\left(CP(k|k-1)C^T + R(k)\right)^{-1} \tag{4-17}$$

$$\rightarrow \hat{x}(k|k) = K(k)z(k) + (I_n - K(k)C)\hat{x}(k|k-1)$$

$$\rightarrow \hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\{z(k) - C\hat{x}(k|k-1)\} \tag{4-18}$$

and Eq. (4-14) becomes

$$P(k|k) = P(k|k-1) - K(k)CP(k|k-1) \tag{4-19}$$

4-2 Extended Kalman Filter

The original Kalman filter was developed to deal with linear systems. In the physical reality however, the vast majority of processes have nonlinearities in their behaviour. Therefore many attempts have been made to develop suitable minimum mean squared error (MMSE) estimators for nonlinear problems. These are generally suboptimal solutions however, because the optimal solution requires that the complete conditional probability density is known, which may require an unbounded number of parameters (Kushner, 1967). A stochastic discrete-time nonlinear state-space system can be modelled as follows:

$$\begin{aligned} x(k+1) &= f_d(x(k), u(k), k) + Fw(k), \\ z(k) &= h(x(k), u(k), k) + Gv(k) \end{aligned} \tag{4-20}$$

where f_d and h are nonlinear vector functions. The process and measurement noises $w(k)$ and $v(k)$ are assumed to appear linearly in the model. The system whose states are to be estimated is often continuous in time. In that case it can be described by a continuous time process model. The system dynamics are then represented in generic continuous state space form along with the discrete measurement equation in Eq. (4-21),

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) + Fw(t), \qquad x(t_0) = x_0 \\ y(t) &= h(x(t), u(t), t) \\ z(k) &= y(k) + Gv(k), \qquad k = 1, \ldots, N \end{aligned} \tag{4-21}$$

In this case the one step ahead prediction, Eq. (4-25), can be calculated by directly integrating the nominal state equation using a numerical integration routine. Note that the input vector $u(t)$ should be averaged or interpolated over the integration interval as denoted by $\bar{u}$.
The Extended Kalman filter (EKF) has been developed to apply the Kalman filter to the system in Eq. (4-21). The idea is to linearise all the nonlinear models so that the conventional Kalman filter can be applied.
A Taylor series expansion of the discrete time stochastic process* $x(k+1) = f_d(x(k))$ around $\bar{x}(k) = \bar{x}_k$ can be expressed as

$$x(k+1) = f_d(x(k)) = f_d(\bar{x}_k + \delta x_k) = f_d(\bar{x}_k) + \sum_{n=1}^{\infty} \frac{1}{n!} \nabla^n f_d(\bar{x}_k)\, \delta x_k^n \tag{4-22}$$

where δx_k is a zero mean Gaussian variable with covariance P_{xx} and $\bar{x}_k$ is the mean state vector at time k. The mean and covariance of $x(k+1)$ are, from (S. Julier & Uhlmann, 2001)

$$\bar{x}(k+1) = f_d(\bar{x}_k) + \frac{1}{2}\nabla^2 f_d(\bar{x}_k)\, P_{xx}(k) + \frac{1}{4!}\nabla^4 f_d(\bar{x}_k)\, E\left\{\delta {x_k}^4\right\} + \ldots \tag{4-23}$$

$$\begin{aligned} P_{xx}(k+1) = {} & \nabla f_d(\bar{x}_k)\, P_{xx}(k) \left(\nabla f_d(\bar{x}_k)\right)^T + \tfrac{1}{2\times 4!}\nabla^2 f_d(\bar{x}_k) \left[E\left\{\delta {x_k}^4\right\}\right. \\ & \left. - E\left\{\delta x_k^2 P_{xx}(k)\right\} - E\left\{P_{xx}(k)\, \delta x_k^2\right\} + P_{xx}^2(k)\right] \\ & \times \left(\nabla^2 f_d(\bar{x}_k)\right)^T + \tfrac{1}{3!}\nabla^3 f_d(\bar{x}_k)\, E\left\{\delta x_k^4\right\} \left(\nabla^3 f_d(\bar{x}_k)\right)^T + \ldots \end{aligned} \tag{4-24}$$

Observe that the nth order term in the series for $\bar{x}(k+1)$ is a function of the nth order moment of x multiplied by the nth order derivative of $f_d(x)$ evaluated at $x = \bar{x}(k)$. So the

*for brevity, the dependence on $u(k)$ and k has been omitted. This doesn't change the form of the expansion however

mean is correct up to nth order provided that the moments and derivatives can be evaluated correctly up to nth order. A similar reasoning holds for the covariance as well, although the structure of the terms is more complex. Because the scaling factor multiplying each term becomes progressively smaller for larger n, the lowest order terms will likely have the largest impact on $\bar{x}(k+1)$ and $P_{xx}(k+1)$. Therefore, the filter algorithm should be concentrated on evaluating the lowest order terms.
The EKF does exactly this: it truncates the Taylor series after the first term, assuming that the second and higher order terms of δx_k in Eq. (4-22) can be neglected. It follows that

$$\bar{x}(k+1) \approx f_d(\bar{x}_k)$$
$$P_{xx}(k+1) \approx \nabla f_d(\bar{x}_k) P_{xx}(k) (\nabla f_d(\bar{x}_k))^T$$

The observation function h is linearised along similar lines. The first order derivatives of f_d and h, the Jacobian matrices F_x and H_x, need to be evaluated at each time step. This means that the EKF assumes both functions are differentiable. The EKF algorithm can be summarised as follows:

1. the one step ahead prediction

$$\hat{x}(k+1|k) = \hat{x}(k|k) + \int_{t_k}^{t_{k+1}} f(\hat{x}(t|t_k), \bar{u}(t), t)\, dt \tag{4-25}$$

2. the error covariance matrix of the prediction

$$P(k+1 \mid k) = \Phi(k) P(k \mid k) \Phi^T(k) + Q_d \tag{4-26}$$

where

$$\Phi(k) = \exp(F_x(k) \cdot (t_{k+1} - t_k))$$
$$F_x(k) = \frac{\partial}{\partial x} f(\hat{x}(k \mid k), u(k), k)$$
$$Q_d = \Gamma Q \Gamma^T, \qquad \Gamma = \frac{\partial f}{\partial u}$$

3. the Kalman gain matrix

$$\begin{aligned} K(k+1) = {} & P(k+1 \mid k) H_x^T(k+1) \\ & \times \left\{ H_x(k+1) P(k+1 \mid k) H_x^T(k+1) + R \right\}^{-1} \end{aligned} \tag{4-27}$$

where

$$H_x(k+1) = \frac{\partial}{\partial x} h(\hat{x}(k+1 \mid k), u(k+1), k+1)$$

4. the measurement update equation

$$\begin{aligned} \hat{x}(k+1 \mid k+1) = {} & \hat{x}(k+1 \mid k) + K(k+1) \\ & \times \{ z(k+1) - h(\hat{x}(k+1 \mid k), u(k+1), k+1) \} \end{aligned} \tag{4-28}$$

5. the error covariance matrix of the state estimate

$$\begin{aligned} P(k+1 \mid k+1) = {} & [I_n - K(k+1) H_x(k+1)] P(k+1 \mid k) \\ & \times [I_n - K(k+1) H_x(k+1)]^T + K(k+1) R K^T(k+1) \end{aligned} \tag{4-29}$$

4-3 Iterated Extended Kalman Filter

Depending on the problem, the EKF may not always converge. Especially, EKF performance is sensitive to the choice of the initial state estimate. If this is not close enough to the true state at that time, the filter may not converge. This also means that the estimate may diverge at a later time, if the measurements are momentarily of poor quality. This limits the practical usefulness of the EKF.
A method to reduce the effect of measurement function nonlinearity, thereby improving filter performance and convergence interval, is due to J. V. Breakwell, although first published by (Denham & Pines, 1966). It comprises local iteration of the measurement update, re-linearising about the updated state $\hat{x}\left(k+1 \mid k+1\right)$ to compute a new updated state which is presumably closer to the true state. The iteration may continue until the difference between two consecutive iteration steps is below a threshold ε. The algorithm is similar to the EKF, with Eq. (4-28) replaced by Eq. (4-30). Let η_i be the ith iteration to the updated state, then

$$\begin{aligned}\eta_{i+1} = \; & \hat{x}\left(k+1 \mid k\right) + K\left(k+1\right) \\ & \times \left[z\left(k+1\right) - h\left(\eta_i, u\left(k+1\right), k+1\right) - H_x\left(k+1, \eta_i\right)\left\{\hat{x}\left(k+1|k\right) - \eta_i\right\}\right]\end{aligned} \tag{4-30}$$

So at each time step, η_1 is set equal to $\hat{x}\left(k+1|k\right)$ and the stopping criterion is

$$\frac{\left\|\eta_{i+1} - \eta_i\right\|_\infty}{\left\|\eta_i\right\|_\infty} < \varepsilon$$

The last iteration is taken for $\hat{x}\left(k+1 \mid k+1\right)$ and Eq. (4-29) is then evaluated based on this last iteration.

(Breakwell, 1967) has pointed out that this local iteration produces a biased estimate. However, as the error covariance becomes smaller, so does the bias in the estimate. This presumes that the filter is converging of course.
(Jazwinski, 1970) reports significant performance gains of the Iterated Extended Kalman filter (IEKF) over the EKF in some simulated nonlinear systems. The example of a highly nonlinear reentry simulation shows that most performance improvement is achieved with the first two iterations.

4-4 Unscented Kalman Filter

To address the inaccuracies arising from the fundamental first order approximation inherent to the EKF implementation, (S. J. Julier & Uhlmann, 1997) have introduced the concept of Unscented Transforms and extended it to the problem of recursive estimation. The result is known as the Unscented Kalman filter (UKF).
The UKF is based on the idea that

> *"it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation"*, (S. Julier & Uhlmann, 2001).

The algorithm is based on propagating a carefully selected set of state vector variations, called sigma points, through the system nonlinear dynamics and then approximating the first

two moments of the distribution through weighted sample mean and covariance calculations. No linear approximation of the nonlinear transformation is applied and the order of the approximation can be scaled by choosing the number of sigma points. Furthermore, the UKF does not require the calculation of any Jacobian or Hessian matrices, not only simplifying implementation, but also making it suitable for black box applications, e.g. in a filtering toolbox and applications involving non-differentiable functions. When using the conventional $L+1$ number of sigma points, the accuracy of the UKF can be compared to the second-order Gauss filter and the computational order is $O\left(L^3\right)$, which is comparable to the EKF. L is the number of states to be estimated plus the process and measurement noise disturbances.

The UKF algorithm may be summarised as follows:

1. The state estimate and covariance are augmented with the mean and covariance of the process noise
$$x^a(k\,|\,k) = \left[\ \hat{x}^T(k\,|\,k) \quad E\left[w^T(k+1)\right]\ \right]^T$$
$$P^a(k\,|\,k) = \begin{bmatrix} P\,(k\,|\,k) & 0 \\ 0 & Q \end{bmatrix}$$

2. A set of $2L+1$ prediction sigma points is derived from the augmented state and covariance where L is the dimension of the augmented state
$$\chi_{p,0}(k\,|\,k) = x^a(k\,|\,k)$$
$$\chi_{p,i}(k\,|\,k) = x^a(k\,|\,k) + \left(\sqrt{(L+\lambda)\,P^a(k\,|\,k)}\right)_i \qquad i = 1\ldots L$$
$$\chi_{p,i}(k\,|\,k) = x^a(k\,|\,k) - \left(\sqrt{(L+\lambda)\,P^a(k\,|\,k)}\right)_{i-L} \qquad i = (L+1)\ldots 2L$$
where
$$\left(\sqrt{(L+\lambda)\,P^a(k\,|\,k)}\right)_i$$
is the ith column of the matrix square root of
$$(L+\lambda)\,P^a(k\,|\,k)$$
using the definition: The matrix square root A of B satisfies $B \equiv AA^T$.

3. The prediction sigma points are propagated through the equations of motion
$$\chi_{p,i}(k+1\,|\,k) = f\left(\chi_{p,i}(k\,|\,k)\right) \qquad i = 0\ldots 2L$$

4. The propagated sigma points are recombined to produce the predicted state and covariance
$$\hat{x}\,(k+1\,|\,k) = \sum_{i=0}^{2L} W_s(i)\,\chi_{p,i}(k+1\,|\,k)$$
$$P\,(k+1\,|\,k) = \sum_{i=0}^{2L} W_c(i)\left[\chi_{p,i}(k+1\,|\,k) - \hat{x}\,(k+1\,|\,k)\right]\left[\chi_{p,i}(k+1\,|\,k) - \hat{x}\,(k+1\,|\,k)\right]^T$$

where the weights for the state and covariance are given by

$$W_s(0) = \frac{\lambda}{L+\lambda}$$

$$W_c(0) = \frac{\lambda}{L+\lambda} + \left(1 - \alpha^2 + \beta\right)$$

$$W_s(i) = W_c(i) = \frac{1}{2\left(L+\lambda\right)}$$

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

Values for α, β and κ have to be chosen to tune the prediction step. The constant α determines the spread of the sigma points around $x^a(k\,|\,k)$ and is usually set to small positive values less than one (typically in the range 0.001 to 1). The secondary scaling parameter κ is usually set to either 0 or $3-L$. When κ is set to 0, weights of the sigma points are directly related to L. When $\kappa = 3 - L$, the 4^{th} order moment information is mostly captured in the true Gaussian case. β is used to incorporate prior knowledge of the distribution of x in the computation of $W_c\left(0\right)$. In the case of the Gaussian distribution, the optimum value is $\beta = 2$. Some guidelines to choose these constants for a particular problem are given in (S. J. Julier & Uhlmann, 2004).

5. For the update step the predicted state and covariance are augmented with the mean and covariance of the measurement noise

$$x^a(k+1\,|\,k) = \left[\begin{array}{cc} \hat{x}^T(k+1\,|\,k) & E\left[v^T(k+1)\right] \end{array}\right]^T$$

$$P^a(k+1\,|\,k) = \left[\begin{array}{cc} P\left(k+1\,|\,k\right) & 0 \\ 0 & R \end{array}\right]$$

6. A set of $2L+1$ update sigma points is derived from $x^a(k+1\,|\,k)$ and $P^a(k+1\,|\,k)$ where L is the dimension of the augmented state

$$\chi_{u,0}(k+1\,|\,k) = x^a(k+1\,|\,k)$$

$$\chi_{u,i}(k+1\,|\,k) = x^a(k+1\,|\,k) + \left(\sqrt{\left(L+\lambda\right)P^a(k+1\,|\,k)}\right)_i \qquad i = 1\ldots L$$

$$\chi_{u,i}(k+1\,|\,k) = x^a(k+1\,|\,k) - \left(\sqrt{\left(L+\lambda\right)P^a(k+1\,|\,k)}\right)_{i-L} \qquad i = (L+1)\ldots 2L$$

7. Alternatively, the prediction sigma points propagated through the equations of motion (step 3) can be used directly

$$\chi_{u,i}(k+1\,|\,k) = \left[\begin{array}{cc} \chi^T_{p,i}(k+1\,|\,k) & E\left[v^T(k+1)\right] \end{array}\right]^T \pm \sqrt{\left(L+\lambda\right)R^a}$$

where

$$R^a = \left[\begin{array}{cc} 0 & 0 \\ 0 & R \end{array}\right]$$

8. The update sigma points are fed to the observation function h

$$\gamma_i(k+1\,|\,k) = h\left(\chi_{u,i}(k+1\,|\,k)\right) \qquad i = 0\ldots 2L$$

9. The result is recombined to yield the predicted measurement and predicted measurement covariance

$$\hat{z}\left(k+1\,|\,k\right) = \sum_{i=0}^{2L} W_s(i)\,\gamma_i(k+1\,|\,k)$$

$$P_{zz} = \sum_{i=0}^{2L} W_c(i)\left[\gamma_i(k+1\,|\,k) - \hat{z}\left(k+1\,|\,k\right)\right]\left[\gamma_i(k+1\,|\,k) - \hat{z}\left(k+1\,|\,k\right)\right]^T$$

10. the UKF Kalman gain is computed as

$$K_{k+1} = P_{xz}P_{zz}^{-1}$$

where the state-measurement cross-covariance matrix is expressed as

$$P_{xz} = \sum_{i=0}^{2L} W_c(i)\left[\chi_{u,i}(k+1\,|\,k) - \hat{x}\left(k+1\,|\,k\right)\right]\left[\gamma_i(k+1\,|\,k) - \hat{z}\left(k+1\,|\,k\right)\right]^T$$

11. The familiar state update equation is

$$\hat{x}\left(k+1\,|\,k+1\right) = \hat{x}\left(k+1\,|\,k\right) + K_{k+1}\left(z\left(k+1\right) - \hat{z}\left(k+1\,|\,k\right)\right)$$

12. Finally, the updated covariance is

$$P\left(k+1\,|\,k+1\right) = P\left(k+1\,|\,k\right) - K_{k+1}P_{zz}K_{k+1}^T$$

4-5 Hybrid Kalman Filter, a combination of IEKF and UKF

The overall structure of the Kalman filter lends itself to a simple combination of two filters. As shown in Section 4-1, the Kalman filter has an observer structure consisting of a prediction and a correction part. The prediction part yields $\hat{x}\left(k+1|k\right)$ and $P\left(k+1|k\right)$ which can then be corrected with the second part of another filter algorithm.
There is a tradeoff between accuracy and computation time when comparing the (I)EKF with the UKF. Applied to the optical flow (OF) problem in this thesis, it takes the UKF about 6 times longer than the IEKF. The UKF does give better results however, so it may be interesting to see whether the performance gain is achieved mainly in the prediction part or in the correction part. In order to investigate this, the Hybrid Kalman Filter (HKF) algorithm is introduced. It uses the UKF prediction (steps 1 - 4) and the IEKF correction (steps 3 - 5 with Eq. (4-30)).
If the performance of this filter is much like the UKF, it may be concluded that the largest performance gain is due to the use of the Unscented Transform (UT) in the prediction part. If, on the other hand, the HKF performs much like the IEKF, then the correction part is most affected by the use of the UT. Of course, these results will only apply to this particular problem as research results in literature show a large dependence of filter algorithms on the specific problem they are solving.

Part II

Simulation

Chapter 5

Structure

This chapter explains the goals which the simulation intends to fulfil and how the problem has been approached. The simulated sensors require information not only about the vehicle state, for which the equation of motion are solved, but also about the environment. Particularly, the geometry of the walls surrounding the vehicle. An algorithm to calculate the required camera viewing distances is described. Also, the digital nature of the data collection has been modelled. Finally, the whole simulation process is summarised in a flow diagram.

5-1 Goals & Approach

The first goal of the simulations is to explore the problem in general. Defining the sensor configuration has to be done based on which states should be estimated and the sensitivity of the observability of those states to properties of the environment and the sensors themselves. This includes e.g. the geometry and scale of the room, sensor noise level and camera field of view.
The simulation should also provide a convenient and fast way to test various algorithms for filtering the required state variables from the sensor data.
Finally, the simulations can be used to choose an experiment setup. It should give an idea whether the experiment will be able to show the performance differences between the filter algorithms.

The algorithms developed for filtering can later be used in the hardware-in-the-loop case. To this end, it is important that the simulated data resembles the hardware as closely as possible. This has been done by concurrently developing the simulation and the hardware. In the first stage the simulation was used to define the concept and later on the simulation has been refined using knowledge about the e.g. the sensor properties gained from building the hardware.

5-2 modelling

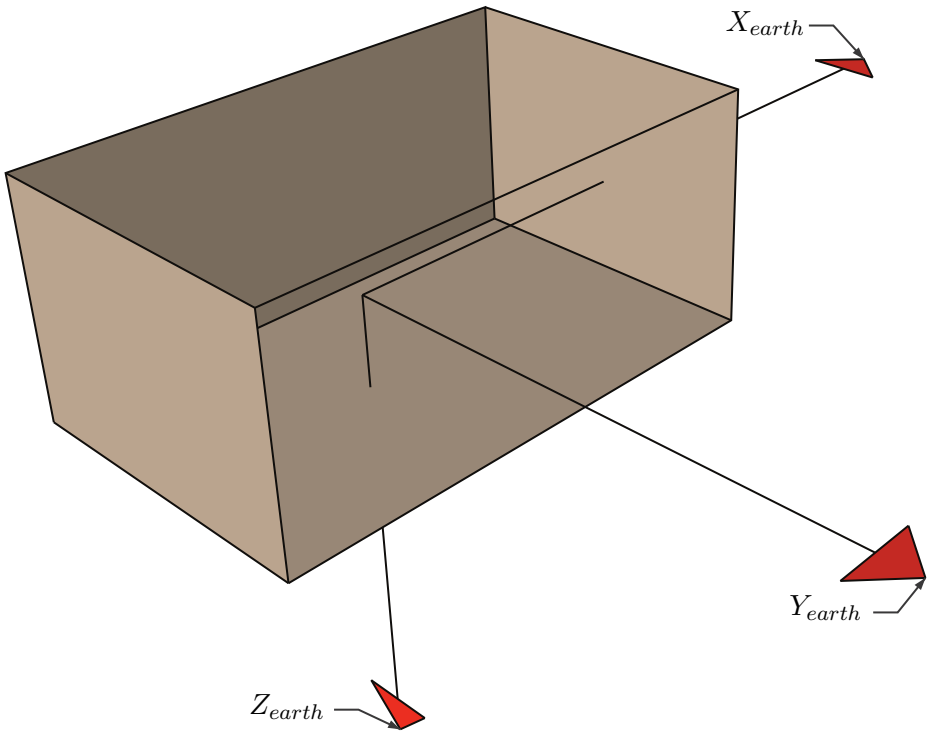


Figure 5-1: Diagram of the room geometry

The first task was to generate optical flow signals. As the sensor concept is designed with indoor flight in mind, the environment should be an enclosed space with walls and corners. A generic room model has been created as in Figure 5-1. The origin of the earth frame of reference is the starting position for the simulation. The direction of gravity is in the positive Z_{earth} direction. Six distances along the positive and negative directions of the axes define the position of the walls.
The next step is to calculate the distance from an optical flow (OF)-camera to a wall along the camera optical axis (the centerline of its field of view). The situation is sketched in Figure 5-2. The position of the camera is at $\vec{p}$, the unit vector defining its optical axis is $\vec{u}$, the line-of-sight distance is s_I, the unit vector defining the wall orientation is $\vec{n}$ and the position of the wall is defined by its distance to the origin, d_{wall}. A point on the plane of the wall can be defined as: $\vec{a} = -\vec{n}\, d_{wall}$.
The line-of-sight can be described by the parametric equation $\vec{L}(s) = \vec{p} + s\,\vec{u}$. $\vec{L}(s)$ can be parallel to the wall in which case $\vec{n} \cdot \vec{u} = 0$. So this possibility should be checked. In a numerical simulation, a threshold should be defined, as the dot product may never yield exactly zero. Assuming that $\vec{L}(s)$ is not parallel to the wall, the line-of-sight distance s_I may be calculated as follows. At the intersection point $\vec{b}$, $\vec{L}(s) - \vec{a}$ is perpendicular to $\vec{n}$. This condition is equivalent to $\vec{n} \cdot \left(\vec{L}(s) - \vec{a}\right) = 0$. Solving for s yields:

$$\vec{n} \cdot (\vec{p} - \vec{a}) + s\,\vec{n} \cdot \vec{u} = 0$$

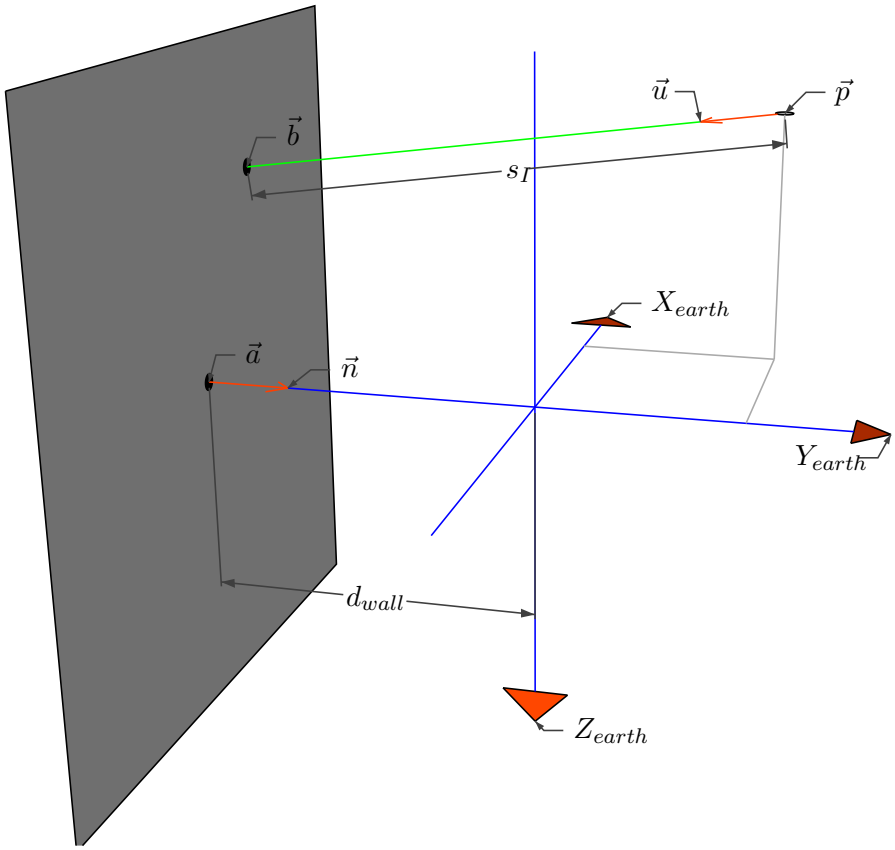
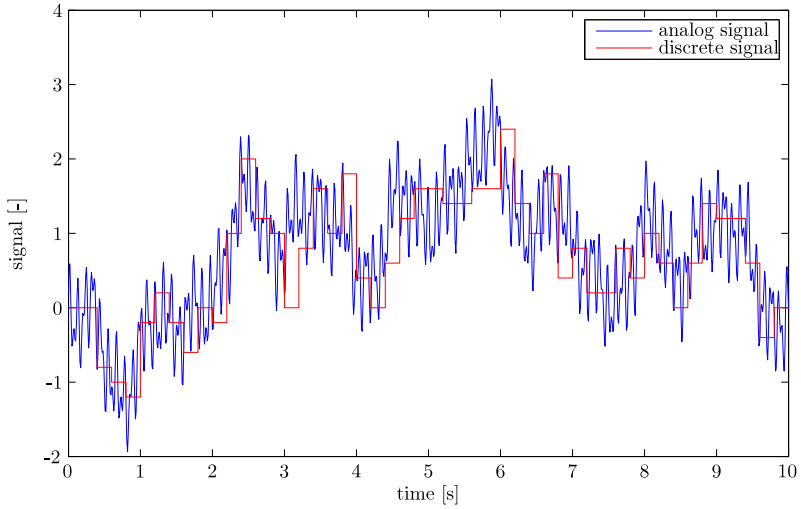


Figure 5-2: Line-of-sight calculation

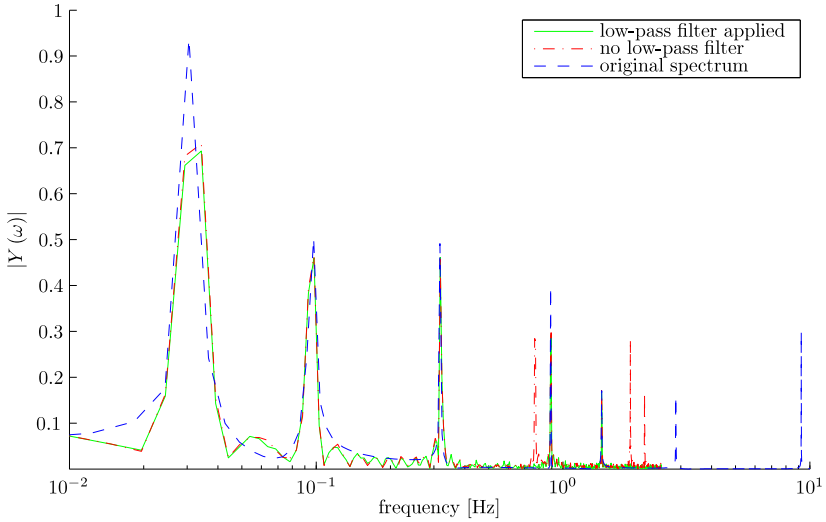
$$\rightarrow s_I = \frac{-\vec{n} \cdot (\vec{p} - \vec{a})}{\vec{n} \cdot \vec{u}} \tag{5-1}$$

However, since there are six walls and six OF-cameras, each camera has to be tested for each wall. This yields six distances for each camera, some of which will be negative. The negative ones correspond to walls in the opposite direction of $\vec{u}$. The wall, which the sensor is looking at, can be found by selecting the smallest positive distance. The distances found for each sensor can then be substituted in Eq. (2-1) together with the vehicle motion to yield the OF signals.

The model for the accelerometer signals simply uses Eq. (2-10) for the specific force vector. The scale factor and bias error are assumed to remain constant and to have been measured immediately before the experiment. This means they can be subtracted from the measurement data and are therefore not required in the simulation.
Sofar only those physical processes have been described which are desirable to be measured. Unfortunately though, all practical sensors are subject to additional undesirable processes which affect their output. The most general one is measurement noise. This has been modelled by adding a (band-limited) gaussian noise term to the signals. The magnitude of the noise is determined as a fraction of the standard deviation of the signals themselves.
Anther important mechanism altering the ideal signals is the fact that the sensors have a



(a) time domain



(b) frequency domain

Figure 5-3: Sampling a continuous signal

digital output, which, in the case of the OF-cameras, originates in the array of photoreceptors and the digital signal processor (DSP). The accelerometers generate continuous signals which are subsequently sampled in an analog-to-digital converter (ADC). Both types of sensor output can be simulated by rounding the continuous signals to a set of discrete values which are multiples of the step size. The rounding should be towards the zero, i.e. selecting the first discrete value which is closer to zero than the continuous signal.
Also, the discrete signals are encoded using a finite number of bits. This means there is a maximum and minimum value determined by the number of bits used. For example, an eight bit ADC can produce $[10000000, 01111111]_{output} \Rightarrow [-10000000, +01111111]_{binary} = [-128, +127]_{decimal}$, using standard two's complement encoding. Multiplying by the step size, this yields the saturation limits of the digitised sensor system.

In order to properly simulate the sample rate of the hardware, it may be necessary to choose a lower sample rate for the sensor output than the dynamics requires for accurate numerical integration. No low pass filtering has been applied prior to downsampling in the simulation, because no indication has been found that the hardware does apply an analog low pass filter before sampling. This means that high frequency signal content may show up in the sampled signal as low frequency oscillations which aren't present in the real process. This effect is known as aliasing and it is an additional disturbance which has been intentionally simulated, because it is expected to be present in the hardware as well.
An example of what happens when a continuous time analog signal is sampled at 5 Hz and a sample step of 0.2, rounding towards the zero, is shown in Figure 5-3. The "continuous" signal is simulated by taking a very small timestep. There are three additional peaks in the spectrum plot of the signal which has been sampled without first applying a lowpass filter. These are absent in the plot which has first been filtered using an 8th order Chebyshev Type I lowpass filter with a cut-off frequency of 0.4 times the sample frequency. The additional peaks are caused by the aliasing effect. No spectral content is present above $5/2 = 2.5$ Hz for the sampled signals. This means that when sampling at 50 Hz, the OF-cameras and accelerometers cannot measure oscillations faster than 25 Hz, but they will get aliasing effects from that region. So if there is a significant amount of power present at frequencies above 25 Hz, such as from sources like an electric motor, it will cause major measurement errors. The hardware experiment doesn't contain such vibration sources, thus limiting the problem. The only high frequency source is the measurement noise originating in the sensors themselves.

The OF-sensors used in the hardware exhibit a peculiarity concerning the sampling: The step size is a function of the sample timestep used. This is described in Chapter 7. The relation is described by Eq. (7-1):

$$\delta\Omega = \frac{\delta\alpha}{dt_s}, \quad \delta\alpha = 5.48 \cdot 10^{-3}\,\text{rad}$$

e.g. choosing $dt_s = 0.02$ s gives $\delta\Omega = 0.274$ rad/s $= 15.7\,^\circ/s$, while choosing $dt_s = 0.05$ s gives $\delta\Omega = 0.110$ rad/s $= 6.28\,^\circ/s$. Clearly, there is a trade-off between high sample rates and good sample resolution. This trade-off has been investigated by running simulations at three sample rates: 3000 Hz, 50 Hz and 20 Hz. The first one is intended to provide an ideal case and the lower two are realistic frequencies to run the hardware on.

To quickly implement simulations, the platform of choice is Matlab/Simulink. It provides access to all the required tools for modelling a system and analysing data. Initially, the system was simulated in Matlab only, but later it was moved to Simulink as it handles the integration of the state derivatives, making the modelling easier. Also, it provides Embedded Matlab (EML) functions, which are compiled to run much faster than the regular m-code and output can be viewed while the simulation is still running.
The flow diagram of the simulation is shown in Figure 5-4. It gives an overview of the major elements of the simulation and the data flows from one element to another. The process starts by loading some parameters like aircraft inertias, room dimensions and initial conditions into the Matlab workspace. Then the *data generation* Simulink model is called. The blocks *Pendulum*, *Helix* and *Random Force* are three options for the input model. These calculate the input vector $\vec{u} = \begin{bmatrix} X & Y & Z & \mathcal{M}_x & \mathcal{M}_y & \mathcal{M}_z \end{bmatrix}^T$ as a function of the current state of the aircraft. *Pendulum* generates the input for a pendulum motion in one plane with drag

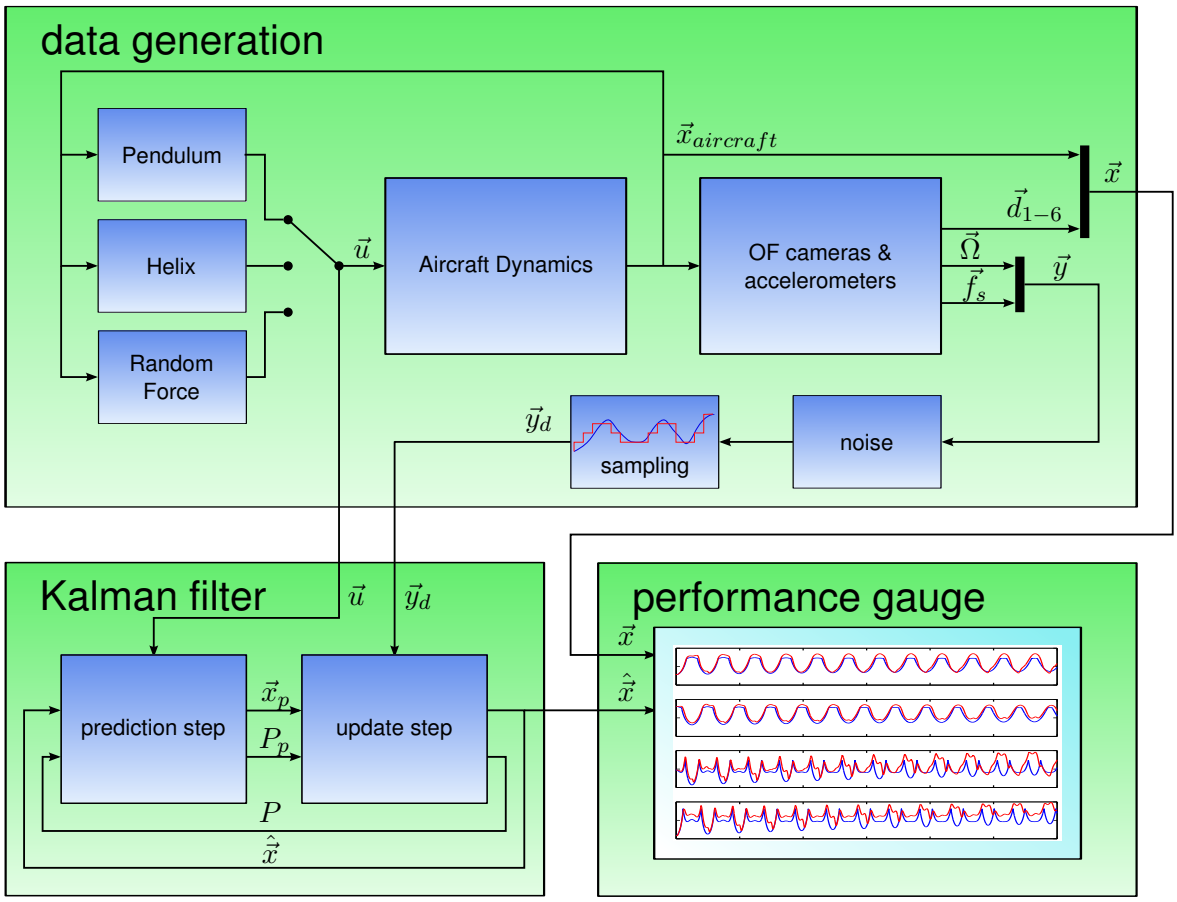


Figure 5-4: Flow diagram of the simulation

simulating the hardware experiment setup, *Helix* produces an ascending helical flight path with varying diameter and *Random Force* generates forces and moments from a gaussian distribution with predetermined standard deviations.
Aircraft Dynamics calculates the aircraft motion using the general nonlinear equations of motion (EoM) (Eq. (2-18)). The result is $\vec{x}_{aircraft} = \begin{bmatrix} u & v & w & p & q & r & \psi & \theta & \varphi \end{bmatrix}^T$ and in addition the position and accelerations are output. The position is used by the helix input and the accelerations are required for the accelerometer equations. The *OF cameras & accelerometers* block generates the sensor signals $\vec{y} = \begin{bmatrix} \vec{\Omega}^T & \vec{f}_s^T \end{bmatrix}^T$ and distances to walls $\vec{d}_{1-6}$ as described above. These are only the ideal processes without noise and generated at the same frequency as $\vec{x}_{aircraft}$. These effects are added to the output by *noise* and *sampling*, respectively. The *sampling* block uses a sampling timestep, which is an integer multiple of the simulation timestep and an equidistant ADC step size with saturation limits.
The time sequences of $\vec{u}$, $\vec{x}$ and $\vec{y}_d$ are stored in a MAT-file for processing by the various Kalman filter types, which are separate Simulink models, and finally presentation of the data in plots to gauge the performance of the filters.
The Kalman filter is written in one EML function block which computes $\hat{\vec{x}}$ and the state covariance matrix P for one timestep. These are then fed back using a unit delay to the *prediction step* together with $\vec{u}$ for computation of the predicted state $\vec{x}_p$ and covariance P_p

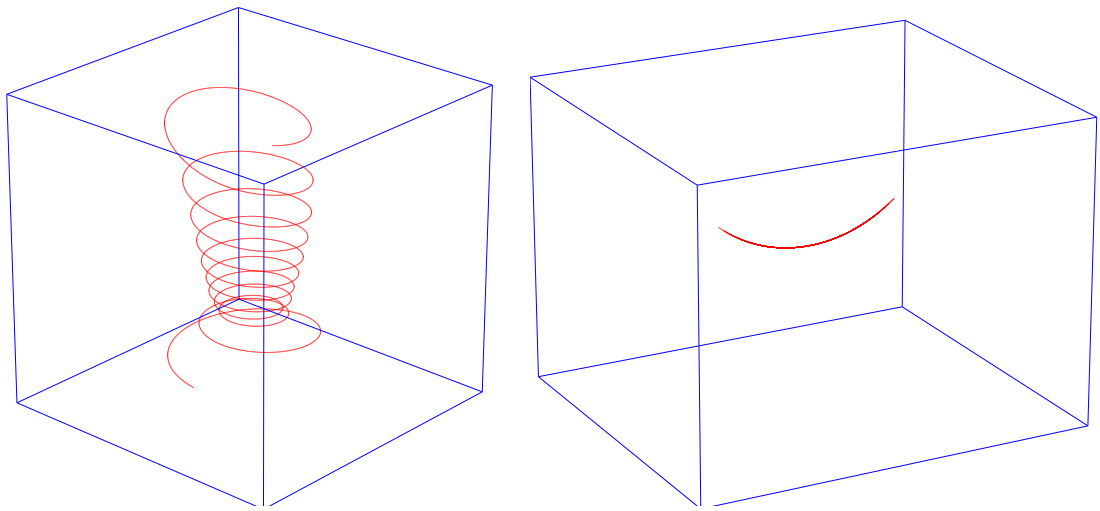
of the next timestep. The only components of $\vec{u}$ being used as input by the Kalman filter are $\mathcal{M}_x$, $\mathcal{M}_y$ and $\mathcal{M}_z$. The *update step* uses $\vec{y}_d$ together with R and P_p to correct $\vec{x}_p$ as described in Chapter 4.

Chapter 6

Results

This chapter describes which conditions have been simulated and discusses the resulting filter output. To investigate the state estimation performance of the filter algorithms, some important parameters have been varied. The combinations of parameter values creates a number of condition cases for each of which simulated sensor and reference state data has been generated. The graphs with the results are referred to in the text and have been included with the appendix. Some simulation specific conclusions are drawn from the results, leaving the overall conclusions to Chapter 10.

6-1 cases



(a) helix (b) pendulum

Figure 6-1: Tracks of the simulated vehicle with respect to the room (wireframe)

Three filter algorithms have been tested using simulated data: the Iterated Extended Kalman filter (IEKF), Hybrid Kalman Filter (HKF) and Unscented Kalman filter (UKF). Two of the input conditions, namely the helical path and the pendulum motion have been applied to the filters. The tracks of these motion cases are shown in Figure 6-1. The random force option is not physically realistic and doesn't add sufficiently to the results to include it in the analysis. The data from the remaining two motion cases has been corrupted by measurement noise. It is band-limited white noise, generated by Simulink and the standard deviation is 10% of each signal standard deviation. Four sampling cases have been defined: case A uses a very high sampling frequency of 3000 Hz and double precision numerical format (64 bits), case B has been down-sampled to 50 Hz with double precision, case C is 50 Hz and 8-bit precision and case D is 20 Hz and 8-bit. Table 6-1 gives an overview of the conditions. The last two sampling cases use the same data rates and precision as produced by the ATmega1281 microcontroller and can therefore serve as a test to compare the simulated pendulum case with the hardware experiment.
The initial conditions have been chosen as a generic estimate assuming some idea of the dimensions of the environment and the vehicle motion. The helix has been estimated using

$$\vec{x}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}^T$$

and the pendulum using

$$\vec{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -40\frac{\pi}{180} & 0 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}^T$$

Table 6-1: Simulation conditions

Sampling case				Motion case	
Case	Noise-to-signal ratio	dt_s [s]	Num. format	Helix	Pendulum
A	10%	$\frac{1}{3000}$	double (64-bit)	IEKF HKF UKF	IEKF HKF UKF
B	10%	$\frac{1}{50}$	double (64-bit)	IEKF HKF UKF	IEKF HKF UKF
C	10%	$\frac{1}{50}$	signed char (8-bit)	IEKF HKF UKF	IEKF HKF UKF
D	10%	$\frac{1}{20}$	signed char (8-bit)	IEKF HKF UKF	IEKF HKF UKF

6-2 results

The resulting sensor output and motion state generated by the helix and pendulum simulations is included with Appendix A-1 and Appendix B-1, respectively. They show the clean

states without any noise or downsampling applied.
The helix provides a non periodic motion case with a frequency sweep and strongly varying distances with discontinuous derivatives caused by the corners of the room. The rotational rates are not synchronised with the "orbit", in fact the moments have been generated with Matlab's white noise generator.
The pendulum is included for comparison material with the hardware experiment and so the parameters have been chosen to match the experiment as closely as possible. This includes the sensor board attitude with respect to the pendulum and its distance to the pendulum centre of gravity. The resulting motion is a damped oscillation with one dominating frequency. A number of velocities and rotations and φ remain zero, because the motion remains in one plane.
The helix motion provides measured moments as inputs, while the pendulum is an autonomous motion, i.e. there is no external input once the motion has started. With the pendulum, the Kalman filters have to work without moments being supplied as known quantities. However, two moments ($\mathcal{M}_x$ and $\mathcal{M}_z$) are zero and $\mathcal{M}_y$ follows from the states. This facilitates the experiment, simplifying measurements and repeatability. Other than concerning the moments, the filter algorithms have not been modified to account for planar motion in the pendulum case. The out-of-plane attitude of the sensor board with respect to the pendulum results in sufficient signals on all optical flow (OF) components and specific forces.

Table 6-2 provides an overview of the performance of the filters in terms of mean squared error (MSE) of the dimensionless signals. All estimated states have been made dimensionless through division by the mean square of the corresponding simulated states. It also shows the mutual differences between the filters by the mean squared difference (MSD) of the dimensionless estimated states.

Table 6-2: MSEs of the dimensionless estimated states and mutual differences between the filters

Simulation Run	MSE of the IEKF	MSE of the HKF	MSE of the UKF	MSD between IEKF & HKF	MSD between IEKF & UKF	MSD between UKF & HKF
helix case A	0.2319	0.1746	0.1937	0.2383	0.2600	0.0388
helix case B	1.5433	0.5111	0.6293	1.4029	1.5050	0.4744
helix case C	0.7271	0.5719	0.7331	0.8090	0.9189	0.4963
helix case D	3.1604	0.6092	0.9111	3.0702	3.2928	0.7987
pendulum case A	0.2310	0.0875	0.1052	0.2712	0.3010	0.0450
pendulum case B	0.5463	0.1670	0.1764	0.5373	0.5493	0.0447
pendulum case C	0.6640	0.2506	0.2603	0.5760	0.5879	0.0492
pendulum case D	11.7048	0.1920	0.2044	11.7343	11.7349	0.0520

The mean values presented in the table are composed of all elements of the state vector. That is also the reason for first making all values dimensionless. This condenses the information, but each individual value has little meaning as such. Rather, the idea is to compare the MSEs between filters and sample cases. This gives a quick indication of the relative performance of the filters under the various conditions.
The pendulum case is only based on the nonzero states. The simulated states which remain

zero, because of the in-plane motion, cannot be used for division to make the corresponding estimated states dimensionless.
There is a general trend (with 3 exceptions) of increasing MSEs going from case A to D. This is expected as the quality of the data also decrease from A to D. The IEKF results of the pendulum case D have diverged which explains the very high MSE.
The MSDs show how closely the estimated states follow each other. This metric has been chosen instead of the more common correlation, because it divides the differences by the same value as the MSEs thus allowing comparison between MSE and MSD as well.
The MSDs give an answer to the question posed in Section 4-5, namely which part of the UKF filter provides the largest contribution to the performance gain observed between IEKF and UKF. Since the HKF has similar MSE as the UKF and has a much smaller MSD with the UKF than with the IEKF, it may be concluded that the effect of the Unscented Transform (UT) in the prediction part may be attributed with the largest performance gain. After all, the UT has been implemented in the prediction part of the HKF. This is also the most computationally intensive part since it involves $4n + 1$ function evaluations of the equations of motion (EoM), steps 2 and 3 in Section 4-4, whereas the IEKF requires only one. Application of the UT to the update part uses the propagated sigma points from the prediction which are then fed to the observation function. This is algebraic however, requiring no expensive integration algorithm and is therefore much quicker.
The HKF does even more than just approach the UKF performance, it slightly improves on it. The iterations in the update part (Eq. (4-30) on page 48) may be the cause of this improvement. In any case, it is apparent that the nonlinear nature of the observation equations from Section 2-3-3 does not provide a performance advantage to the UT based update part. This is a remarkable and unexpected finding. (S. Julier & Uhlmann, 2001) describe a benefit for systems involving coordinate transformations. The observation equations in this work also involve a transformation from body frame to the rotated sensor frame. Moreover, the prediction does not provide any information on the distances, which are the hardest states to estimate. So these have to be estimated entirely by the update part. Nevertheless, the iterated linearised algorithm of the IEKF, using the predicted state and covariance from the UKF, does this even a little better than the UT based algorithm. Whether the small performance difference is significant cannot be concluded from these results however.

A more detailed overview of the results is given by graphs. The *errors* of the estimated states of the helix motion have been plotted in Figures A-8 - A-23 and of the pendulum motion in Figures B-7 - B-22. All three filter types have been plotted together allowing for performance comparison.
Looking at these figures, the first point of observation is that the IEKF produces the largest errors and that UKF and HKF have very similar performance, although the HKF converges a bit faster. These conclusions have also been drawn based on Table 6-2. The IEKF has a smaller convergence zone, an example of which can be seen in the graphs of helix case B, Figures A-12 - A-15. Initially the errors are large until the solution happens to get close, after which it converges and the errors stay smaller.
Rotational rates are estimated well by the UKF/HKF, even at 20 Hz, while the attitude estimation deteriorates more quickly at low signal resolutions. The distance estimates tend to lag a bit when abrupt changes in distance occur. The pendulum results show a clear periodicity in the distance errors because of this lagging.

Among the sample parameters which were changed between the cases, the sample rate has the largest impact on performance. The benefit of a better OF resolution, $\delta\Omega$, with 20 Hz, compared to 50 Hz doesn't show in the results, because the lower sample rate itself has a larger negative influence. The graphs show the change from double precision to signed 8-bit integers to hardly have an effect on the performance.
The observability condition number $\mathcal{C}_O$ yields an interesting observation in the pendulum case. Although it has not been included with the plots explicitly, $\mathcal{C}_O$ shows peaks when the pendulum velocity crosses zero. The peaks indicate points in the state space with difficult observability. The velocity error graphs of the UKF/HKF show the largest errors just after this zero crossing. This suggests that the difficult observability condition at the direction reversals may drive the errors in these estimates. More on $\mathcal{C}_O$ will follow in the hardware experiment results, where the observability condition numbers have been plotted.

Part III

Hardware

Chapter 7

Hardware Description

The hardware, which was built for the experiment, is shown in Figure 7-1. Its overall dimensions are $66 \times 51 \times 41$mm. There are four double sided copper circuit boards consisting of two copper layers with an insulating polymer substrate in between them. Three of them are identical and contain the optical flow (OF)-cameras. The fourth is the main board which connects the sensors to the microcontroller (µC), the µC to an external computer via serial link, it provides a clock signal for the µC and a direct current (DC) voltage conversion. In addition, a small board containing the 3-axis accelerometer has been glued onto the main board such that the accelerometer axes are parallel to the optical axes of the OF-cameras. The accelerometer is connected to two pins of the µC using an I^2C serial interface.
The surface of the boards in the picture is textured with the "blueprint" of the circuitry. This blueprint has been used by the CNC mill at Aerospace Software and Technologies Institute (ASTI) to remove copper at the red or blue coloured areas creating separate conduction paths. Red is for one side and blue for the opposite side. This method allows for rapid prototyping of new electronics designs. Compared to conventional etched boards it has the disadvantage that the copper on the conduction paths may tear during milling which can interrupt the connection. Also while in use the board remains vulnerable. A number of bypasses needed to be soldered onto the boards to repair such damage.
The central component of the system is an AVR ATmega1281 µC with an 8-bit Reduced Instruction Set Computer (RISC) architecture. This means it runs only using simple instructions such as bit shifting, addition and multiplication of 8-bit integers. These instructions are each executed in a single clock cycle. If higher precision integer operations are required, it will take more than one clock cycle. The integer arithmetic poses severe limitations on the ability to do onboard Kalman filtering, in addition to the fact that the computational power is at least two orders of magnitude too low. The µC is still very useful for collecting measurement data and forwarding this to the host PC. It is well possible to do this at very precise intervals using the onboard counters which can act as timers and run independently of the CPU. There are several other such peripheral features such as ADCs, Pulse Width Modulation (PWM) and a number of general purpose and specific I/O ports. These ports have been used to communicate with the sensors and with the host PC.

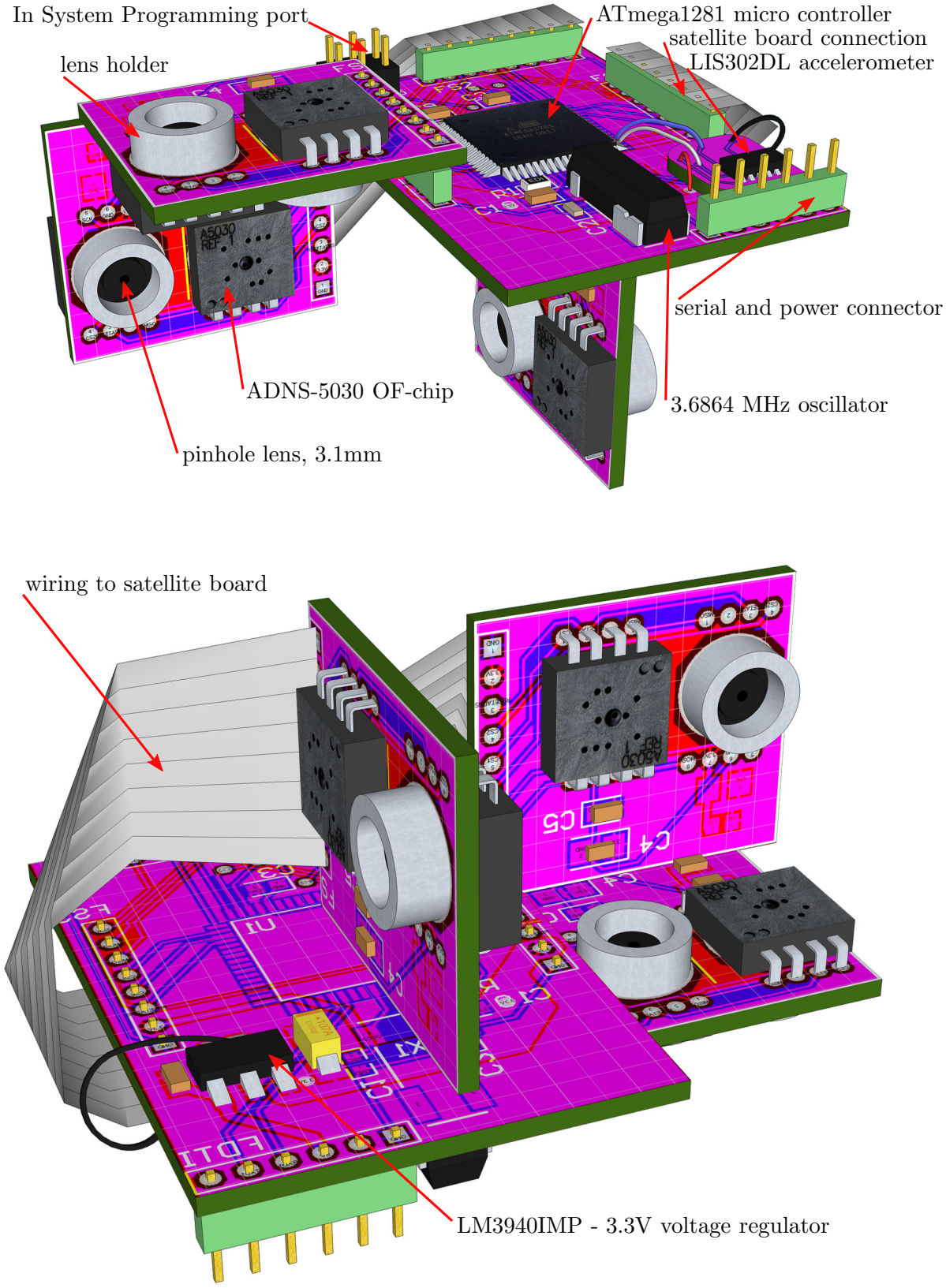


Figure 7-1: 3D render of the sensor board with 6 OF-sensors and a 3-axis accelerometer

The six ADNS-5030 OF Integrated Circuits (ICs) are mounted on three satellite boards as shown in the 3D images. The configuration is designed such that the sensor pairs on each board look in opposite directions and the satellite boards point in three mutually orthogonal directions. They are connected to the µC using a 4-wire serial bus using the Serial Peripheral Interface Bus (SPI) protocol. There is the Master out Slave in (MOSI) pin for sending bits to the ADNS-5030, the Master in Slave out (MISO) pin for sending bits back to the µC, the Serial Clock (SCK) providing the clock signal for synchronous data transfer and a Chip Select (CS) pin for each ADNS-5030. The slaved ADNS-5030 chips only respond when their CS pin is low. This means collecting data from all 6 can be done sequentially through the same two pins (MISO and MOSI) on the µC by pulling each CS down in turn, while pulling up the other Chip Selects. First, an address byte is sent to specify the register from which the contents must be read. Then a zero byte is sent while at the same time the data from the register comes in through the MISO.
The OF signal from each ADNS-5030 has two components which are stored in two registers. The data sequence to read out all OF sensors in one timestep is as follows:

1. Pull the CS pin of chip x low and the CS of chip $x-1$ high;
2. write the power & resolution register to select normal operation and high resolution (1000 cpi);
3. read the motion register to check whether new motion data is available;
4. read the `Delta_X` register for the first OF component;
5. read the `Delta_Y` register for the second OF component;
6. read the quality register to see whether a sufficient number of features are being tracked;
7. and finally continue to the next chip returning to step 1.

To get the OF data from all six sensors takes 4.18 milliseconds. Strictly, the data is not recorded at the same time instant, but the small time delay between the sensors has been neglected.
The ADNS-5030 outputs a count in its `Delta_X` and `Delta_Y` registers representing units of displacement of the mouse (1 counter increment equals $1 \cdot 10^{-3}$ inch displacement). This count accumulates until the registers are read out resetting the value to zero. In order to convert these counts to OF, one needs to know the sample timestep dt_s and the geometry of the mouse assembly which is implicitly assumed by the manufacturer when they define a unit count value. This geometry is shown in Figure 7-2.
The quantity actually measured by the sensor is an angle. A feature tracked across the sensor surface represents a certain angular displacement. When this angular displacement is combined with the surface-to-lens distance it yields the translatory displacement referred to by the manufacturer. In the mouse assembly, the surface to lens distance is 4.90 mm. One counter increment is then equivalent to an angular displacement $\delta\alpha$ of $\frac{25.4}{1000 \cdot 4.90} = 5.18 \cdot 10^{-3}$ rad. This quantity holds for the mouse assembly which has a field of view angle of 0.180 rad. The field of view of the OF-camera is 0.190 rad. Therefore, in this application a unit counter

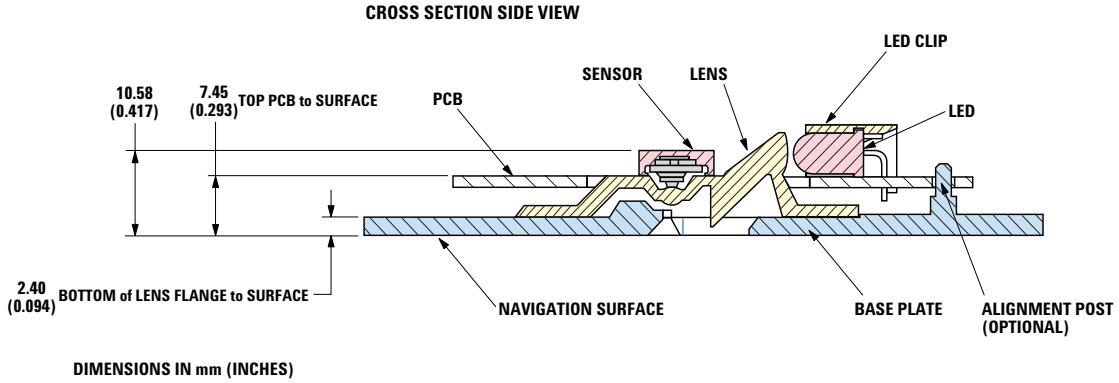


Figure 7-2: The ADNS-5030 IC in its intended mouse application assembly, (Avago, 2008)

increment corresponds to $\delta\alpha = \frac{0.190}{0.180} 5.18 \cdot 10^{-3} = 5.48 \cdot 10^{-3}$ rad.
The OF step size produced by the ADNS-5030 can now be expressed as

$$\delta\Omega = \frac{\delta\alpha}{dt_s} \tag{7-1}$$

e.g. choosing $dt_s = 0.02$ s gives $\delta\Omega = 0.274$ rad/s $= 15.7\,^\circ/s$, while choosing $dt_s = 0.05$ s gives $\delta\Omega = 0.110$ rad/s $= 6.28\,^\circ/s$. Clearly, there is a trade-off between high sample rates and good sample resolution. A large $\delta\Omega$ leads to large rounding errors, especially since the registers are reset to zero after a read operation, effectively producing a round-towards-zero behaviour. This trade-off has been investigated by running the experiment at several sample rates.

The 3-axis accelerometer on the sensor board is the LIS302DL. It is a compact ($3 \times 5 \times 0.9$mm) Micro-Electro-Mechanical Systems (MEMS) device with built-in ADCs. This means that the output is stored in digital registers as signed 8-bit integers, like in the case of the ADNS-5030. There are two sensitivity settings available, namely ± 2 g or ± 8 g. These are actually minimum values and in reality the values are typically ± 2.3 g or ± 9.2 g, respectively. For this application, ± 2 g is the most appropriate setting, because a high sensitivity is critical for good filter performance. The signed 8-bit integer encoding means that the sampling step size is about 18 mg. Of course, the actual values must be measured in order to get correct results. The calibration procedure is described in Chapter 8. Also, as explained in Chapter 5, aliasing may be an issue with this sensor, as there is no mention of any low pass filter being applied. Only a high pass filter may be enabled, but this doesn't solve aliasing and is not desirable for this application.
An Inter-Integrated Circuit Bus (I^2C) interface connects the accelerometer with the µC. This is a two wire serial interface with a clock line (SCL) and a data line (SDA) which are both bidirectional. Each bit transferred on the SDA is accompanied by a pulse on the SCL. The µC is configured as the master device and the accelerometer as a slave. The master initiates all communication by generating a start code. In order to read from a register on the LIS302DL, the Master sends the address of the slave device ending with a zero to signify that the following data will be "written" to the slave. This is followed by the address of the register to be read. Message acknowledgement bits are returned by the slave after both. A

new start code follows and then the slave address is sent again, but this time ended with a one to indicate a read request. The slave should respond with an acknowledge bit followed by the data byte in the register of interest. Finally the master sends an acknowledge bit followed by a stop code.
In this case, the I^2C has been implemented in software using two general purpose I/O pins on the µC. It is generally slower since the CPU has to handle all operations, while a hardware peripheral implementation can run parallel to other CPU tasks. That is not a problem however, since the data rate is limited by the consideration of OF resolution ($\delta\Omega$) which leaves enough time to handle software I^2C for the accelerometer. The details of this interface, especially the timings, are very device specific and custom code was already available for the LIS302DL making it the convenient option as well.

As mentioned before, the experiment has been run at several values of sample timestep dt_s. The shortest possible dt_s is determined by the time it takes to get the data from the sensors and to send it to the PC. This takes on average 38227 clock cycles which at 3.6864 MHz amounts to 10.37 ms or a sample frequency of 96 Hz. The first versions of the onboard software worked with this timestep, simply executing all the tasks in the loop without waiting. However, as explained above, the OF resolution ($\delta\Omega$) depends on dt_s. It is therefore important to have control over the time between readouts of the sensors. To achieve this, one of the onboard counters has been used. It has a 16 bit memory register for the count value and can be connected to the system clock signal, or external source, either directly or through a 1/8 frequency reduction. In this case, the counter has been hooked up with the system clock through the frequency reduction. Furthermore, it has an output compare register to enable counting a predetermined number of clock cycles. Although it is possible to let the counter trigger an interrupt when the count reaches the value in the output compare register, this was not necessary in this case since the CPU doesn't have any other tasks to attend to while waiting for the start of the next sample timestep. Therefore, after the data has been forwarded to the PC, the CPU starts polling the flag which signals that the number of clock cycles is equal to the output compare register. When this occurs the flag and the counter value are reset and the program continues with the next sampling cycle. Figure 7-3 shows the situation when $dt_s = 0.02$s. Each segment amounts to the time spent on one task with respect to the duration of the entire sampling cycle.

The communication with the host PC is taken care of using the Universal Asynchronous Receiver/Transmitter (UART) interface. Asynchronous means that there is no common clock signal between the two communicators. The receiver has to synchronise its internally generated baud rate with the phase of the received start bit. If the transmitter is sending frames at too fast or too slow bit rates, the receiver will not be able to synchronise the frames to the start bit. Frequency mismatch is introduced if the baud rate generator can not do an exact division of the system frequency to get the baud rate wanted. The resulting errors become larger when a higher baud rate is requested. Therefore, the system clock source has been set to the external full swing crystal oscillator on the main board in Figure 7-1, which runs at 3.68640 MHz. At this frequency all standard baud rates up to 230.4 kbps are exact divisions of the system clock. This allows for faster communication with the host PC.
The bytes from a received frame are put in a buffer until the CPU has time to process them or the buffer becomes full. An interrupt flag may be enabled to signal the arrival of new

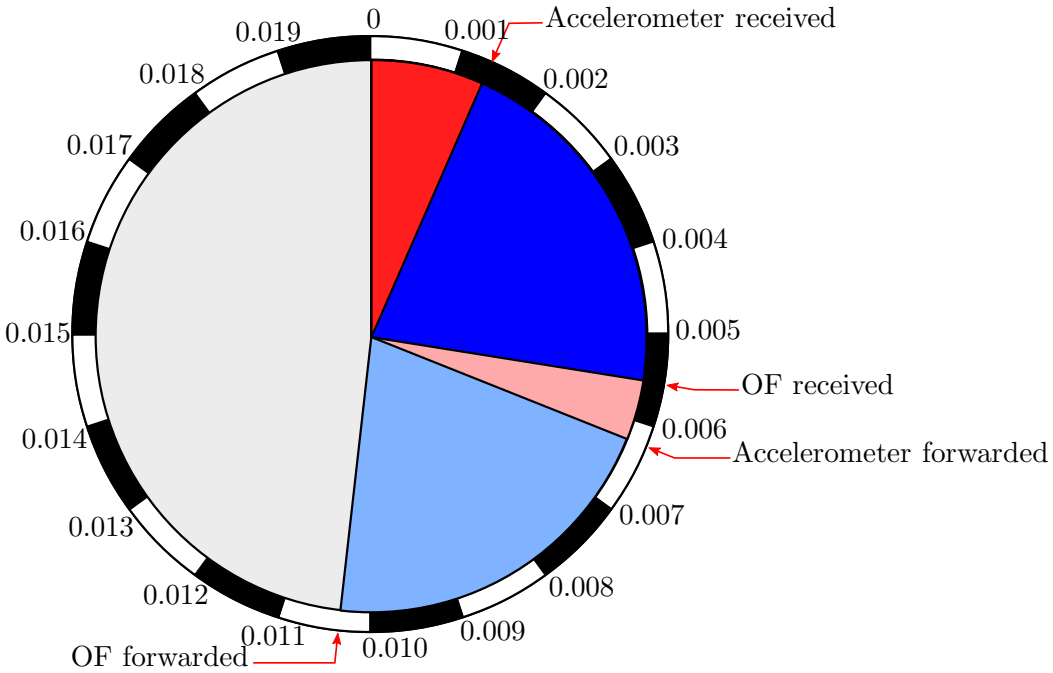


Figure 7-3: Sampling cycle pie chart, the numbers represent time in seconds from the start of the cycle

bytes. This has been done in the µC to let the program react to a command from the user immediately. Only at some stages it is preferable not to interrupt the CPU, such as while communicating with the sensors. Therefore, interrupts are temporarily disabled during those intervals.
It is most convenient to use a USB to TTL serial converter cable for the connection with the PC/laptop. These are available with a 0.1 inch 6 pin single in line female connector which plugs readily into the serial and power connector. The pin layout starting from pin 1 is GND, CTS#, VCC, TXD, RXD and RTS#. GND is the ground pin. VCC should provide a voltage in the range of 3.3 - 5V in order for the voltage regulator to work. TXD is the transmit pin from the PC perspective. It connects to the RXD pin on the µC. The RXD pin on the cable receives data from the µC TXD. CTS# and RTS# are not used.
In order to write new code into the Flash memory of the µC while it is in-system, there is a 6 pin In-System Programming (ISP) port available. The AVRISP mkII In-System Programmer by Atmel has been used in combination with AVR Studio 4 to upload code to the Atmega1281. This allows for rapid debugging and changing of functionality while testing the sensor board. AVR Studio is a software development environment based on the C language with a compiler for the AVR µC family. It has the capability to test run code by emulating the µC, showing exactly what happens at each clock cycle and linking these actions to the C statements. This is very useful for debugging and checking the functionality prior to loading the software onto the µC.
A functional flow diagram of the final version of the C-code has been included as Figure D-1 in Section D-1 of the Appendix.

Chapter 8

Sensor Calibration

The output of real-world measurement instruments cannot be expected to correspond to the quantities being measured without a proper calibration procedure. Many factors influence the raw signal. A calibration procedure aims to correct for the quasi-constant factors, i.e. the ones with a sufficiently constant value over the calibration interval. These may include manufacturing imperfections, slow processes in the sensor or environmental factors, e.g. the effect that a change of local gravity has on a pendulum clock.
In order to calibrate an instrument, one needs a standard. This may be another instrument with an accuracy which is known to be better than the required tolerance. In some cases it may also be a physical quantity of known magnitude, e.g. a proof mass.
This chapter discusses the calibration procedures which have been employed for the sensor board in this thesis work.

8-1 Accelerometer Calibration Procedure and Results

Some accuracy characteristics of the LIS302DL accelerometer from the data sheet are given in table 8-1. It is clear that both the sensitivity and bias are not very accurately known

Table 8-1: Some LIS302DL Mechanical Characteristics, ("LIS302DL Datasheet", 2007)

Parameter	Min.	Typ.	Max.	Unit
Measurement range	±2.0	±2.3	±2.5	g
Sensitivity	16.2	18	19.8	mg/digit
Sensitivity change vs temperature		±0.01		%/°C
Typical zero-g level offset accuracy		±40		mg
Zero-g level change vs temperature		±0.5		mg/°C

by the manufacturer. Also, there is a significant temperature dependency. The calibration procedure will attempt to determine the current sensitivity and bias for each of the three axes. It is assumed that the temperature remains sufficiently constant during the experiment session following a calibration. It means that some warmup time must be allowed before the calibration procedure is performed and that the power remains turned on throughout the experiment session.
Gravity is used as the standard for calibrating the accelerometer. The local gravity magnitude has been taken to be 9.81276 m/s^2, which is the typical value measured in Soest* from (Anderson, 1998). Note that the units of g used by the manufacturer are assumed to refer to the conventional standard value of 9.80665 m/s^2.
The procedure to calibrate the direct digital output $\vec{A}_{raw}$ (in 8-bit signed integer form) and obtain the specific force vector $\vec{f}_s$, is as follows:

1. With the power and serial data connection plugged in, the sensor board is clamped to a lab stand in an orientation such that the X-axis output is maximised. This is done by manipulating the clamp and watching the output with a short period moving average to get a more stable readout.
2. When the maximum is found, the setup is left stationary and the subsequent data are logged over a period of one minute.
3. The mean of this dataset is recorded as $A_{x_{max}}$.
4. Steps 1 - 3 are repeated for the opposite direction and then for the Y and Z-axes, yielding an additional 5 values.
5. The bias and scale values for each axis are calculated as follows:

$$\begin{aligned} bias_x &= \frac{A_{x_{max}}+A_{x_{min}}}{2} \\ bias_y &= \frac{A_{y_{max}}+A_{y_{min}}}{2} \\ bias_z &= \frac{A_{z_{max}}+A_{z_{min}}}{2} \end{aligned} \tag{8-1}$$

$$\begin{aligned} scale_x &= \frac{2g}{A_{x_{max}}-A_{x_{min}}} \\ scale_y &= \frac{2g}{A_{y_{max}}-A_{y_{min}}} \\ scale_z &= \frac{2g}{A_{z_{max}}-A_{z_{min}}} \end{aligned} \tag{8-2}$$

6. Finally the calibrated accelerometer output is:

$$\vec{f}_s = \begin{bmatrix} scale_x & 0 & 0 \\ 0 & scale_y & 0 \\ 0 & 0 & scale_z \end{bmatrix} \left(\vec{A}_{raw} - \begin{bmatrix} bias_x \\ bias_y \\ bias_z \end{bmatrix} \right) \tag{8-3}$$

This is a simple calibration procedure which assumes that the axes are exactly mutually orthogonal. To also measure the deviations from orthogonality one needs to use a two-axis turn table and measure output at many different attitudes. Although such a turn table is available, it is not practical to use it for the following reason: The experiment is carried out in

*Soest is the geographically closest value from a reliable source that the author could find. It should be accurate for Delft to within ±0.01%

a different building and in several sessions. The accelerometer must be calibrated before each session as its internal temperature has an influence on the measurements. Disconnecting the accelerometer and carrying it outdoors and into a room at another temperature might very well have more adverse effects on the calibration than the benefits afforded by using the turn table. The simple calibration procedure described above can be performed quickly and in the same environment as the experiment itself ensuring more constant conditions. Therefore, only the simple calibration has been performed.

8-2 Sensor Board Attitude Determination

The sensor board has been attached to the pendulum at a fixed orientation. In order to measure this orientation, the calibrated accelerometer signals have been used. The pendulum is constrained to swing only in one plane. By measuring the gravity vector at several pendulum positions in this plane, the attitude of the sensor board with respect to the pendulum may be found. In order to measure only gravity, the pendulum has to be stationary. So additional strings have been attached to fix the pendulum at an attitude of about 40° to either side. These angles don't have to be precise as long as the pendulum is in its plane of swing. Measurements were taken for a period of at least one minute at these attitudes and at the straight down equilibrium (0°) attitude. The mean of the resulting data has been taken to find g-vectors at these three attitudes:

$$\begin{aligned} \vec{g}\,(\theta \approx -40^\circ) &= \left[\ -9.129054766208 \quad 0.3234418409320 \quad 3.759077245266\ \right]^T \\ \vec{g}\,(\theta = 0^\circ) &= \left[\ -6.774889040408 \quad 7.1458360400000 \quad 1.268402844240\ \right]^T \\ \vec{g}\,(\theta \approx +40^\circ) &= \left[\ -1.267322016168 \quad 9.5017698009360 \quad -2.221252216848\ \right]^T \end{aligned}$$

The pendulum- (or body-) fixed frame of reference has been defined with the X_B-axis pointing in the direction of motion, the Z_B-axis pointing straight towards the centre of the earth when the pendulum is in its equilibrium position and the Y_B-axis normal to the plane of swing such that it forms a right-handed coordinate system.
The directions of the axes of the pendulum frame can be found in the sensor board frame of reference as follows. The three measured gravity vectors should all lie in one plane. This is the plane of swing of the pendulum. There are three combinations into pairs of vectors possible. The vector cross product of each pair yields a normal vector to the plane. Of course these normal vectors aren't exactly the same, so the mean of the three results is taken as the best estimate of the vector normal to the plane of swing. The result is then divided by its 2-norm to yield a unit length vector $\vec{n}_{Y_B,sensor}$ defining the direction of the Y_B-axis in the sensor board frame. $\vec{g}\,(\theta = 0^\circ)$ is also divided by its length to yield $\vec{n}_{Z_B,sensor}$ and finally $\vec{n}_{X_B,sensor}$ is found by taking the cross product of the other two directions, taking care of the order to get a right-handed coordinate system.
The goal of this procedure can be most conveniently expressed as a direction cosine matrix (DCM). The DCM from body to sensor board frame may be expressed in the directions found as follows:

$$DCM_{b\to s} = \left[\ \vec{n}_{X_B,sensor} \quad \vec{n}_{Y_B,sensor} \quad \vec{n}_{Z_B,sensor}\ \right] = \begin{bmatrix} -0.6077 & -0.4060 & 0.6824 \\ -0.6557 & -0.2278 & -0.7197 \\ 0.4477 & -0.8848 & -0.1278 \end{bmatrix}$$

Of course, $DCM_{s\to b} = DCM_{b\to s}^T$ because the DCM is orthogonal by definition.

8-3 OF Sensor Calibration Procedure and Results

The optical flow (OF) sensors produce counts of angular displacements of features as described in Chapter 7. Knowing the original lens geometry, the present field of view (fov) angle and the sample timestep one may calculate the expected OF step size as has been done in that chapter. However, the actual value may be slightly different due to tolerances in the geometry and possibly other factors. Therefore, a calibration curve is required relating the scale factor to the measured OF.
The sample timestep dt_s is expected to have an influence on the calibration curve, since it affects the scale factor:

$$\delta\Omega = \frac{\delta\alpha}{dt_s}$$

Also, the registers containing the count are flushed when they are read, effectively creating a round-towards-zero behaviour. Thus, a larger $\delta\Omega$ can be expected to increase this effect thereby increasing the lower threshold for accurate OF measurements.
The experiment has been carried out using primarily TL tubes as light source. In addition, some light bulbs were directed at the ceiling. These light sources are all connected to the same AC power source. This means the light intensity oscillates at 50 Hz. These oscillations might have an effect on the perceived OF. The effect is hard to predict because the internal frame rate of the ADNS-5030 is self-adjusting according to an unknown algorithm.
Another factor which is known to strongly influence the sensor output is the number of texture features present on the surface of the objects in the sensor fov. Also, light intensity is an important factor. For the experiment, these last two factors have been dealt with by making sure that both are sufficient. The ADNS-5030 has not been designed for the task it is being used for and future applications may have access to better quality sensors. In order for the Kalman filter to work, the signal should be as good as possible, so the experiment aims to provide optimal lighting and texture conditions.
The calibration procedure has been carried out once before the experiments. That should be sufficient, because the factors affecting this sensor calibration are expected to be very constant in time and reasonably independent of other environmental variables.
The single axis turn table of the instrument lab (room 0.44) at the faculty of Aerospace Engineering in Delft has been used as the standard for the calibration. It can be set to a constant rotational velocity, whose rate is checked using an optical interruptor sensor and a timer. The setup is shown in Figure 8-1. The procedure is as follows:

1. The sensor board is mounted in the centre of the turn table and connected to a laptop through the sliding contacts on the table.

2. Surrounding the table is a metal ring meant as a safety fence. The inside of this ring (the side facing the sensor) is used to attach paper to. The white paper has a black dot print on it with a random pattern. The dots range in diameter between 22 and 44 mm. This size gives a well contrasting texture for the sensor at the distance involved.

3. The table is tuned to a certain angular velocity.

4. Once the angular velocity has been checked and tuned with the timer, measurement data from the sensor board is logged to the laptop for a period of about one minute. Note that the OF data from the sensor board is pre-processed by multiplying the 8-bit

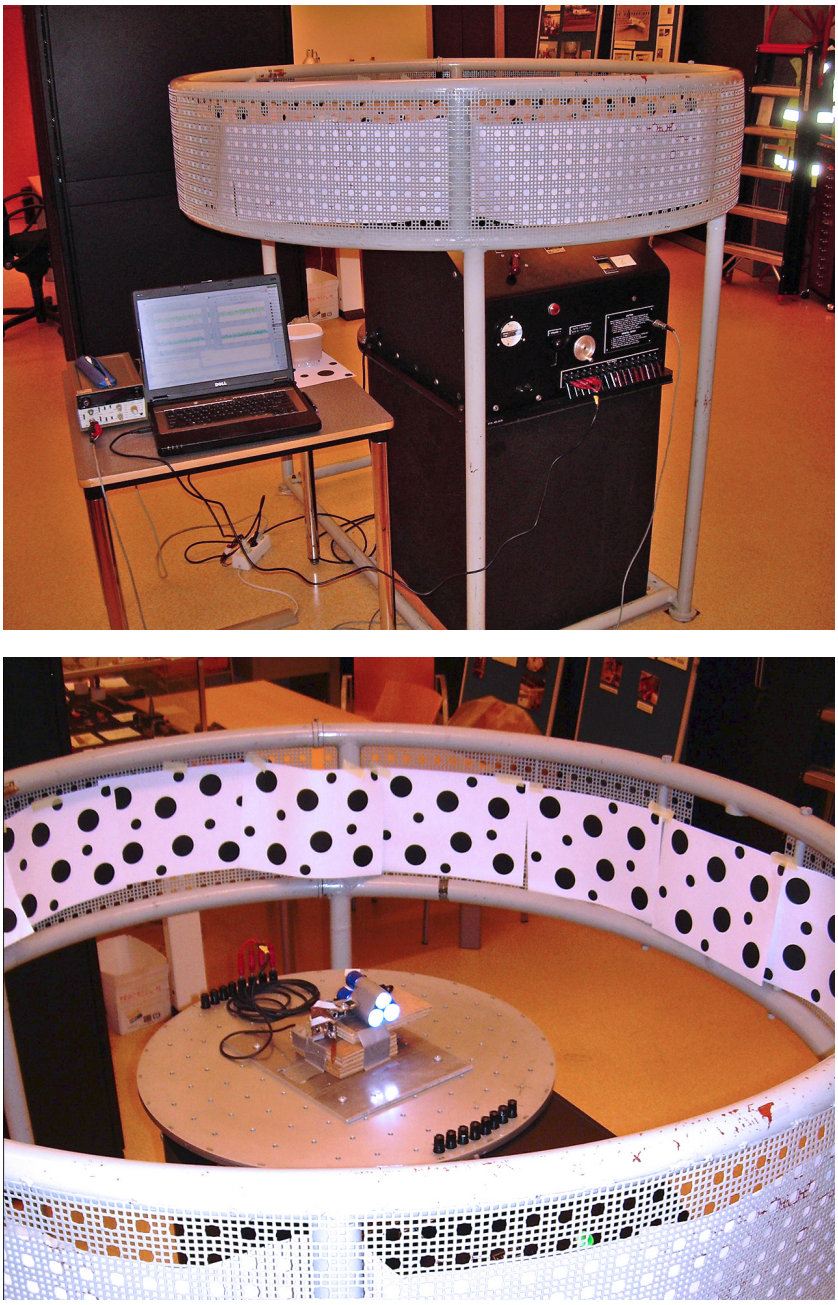


Figure 8-1: The calibration setup. The top image shows the turn table with its controls and the metal fence surrounding it. To the left of it, the laptop for logging the data and to the left of that the timer for tuning the desired angular rate of the turn table. The bottom image shows the top of the turn table with the sensor board mounted to it. The torch next to it is composed of multiple LEDs. The textured paper at which the sensor is looking can be seen on the inside of the fence.

signed integers from the ADNS-5030 by $\delta\Omega$ which yields the expected magnitude of the OF.

5. Steps 3 and 4 are repeated until the following range of angular velocities has been

covered:

$$\omega = \pm \{\ 20 \quad 60 \quad 100 \quad 140 \quad 180 \quad 220 \quad 260 \quad 300 \quad 340\ \} \,\mathrm{deg/s}$$

6. The logged data for each ω is filtered to remove spikes and subsequently the mean is taken. This value is the measured OF (Ω_m) corresponding to that value of ω.

7. Because the sensor motion is purely rotational, the sensor output should equal ω. A calibration graph is constructed by plotting the correction factors $\frac{\omega}{\Omega_m}$ versus Ω_m and interpolating the data points using Matlab's shape-preserving piecewise cubic interpolation.
   To correct new measurement data, a table lookup function may be used to obtain the correction factor corresponding to the pre-processed sensor output Ω_m and multiply that correction factor with Ω_m to obtain the calibrated OF.

The whole procedure has been carried out at 50 Hz sample rate ($dt_s = 0.02$ s) with TL tubes as a light source, at 25 Hz ($dt_s = 0.04$) with TL light and at 50 Hz with a DC light source (to test the effect of oscillations in the light intensity). The graph showing the resulting three calibration curves is shown in Figure 8-2. The first observation is that the minimum reliable Ω depends clearly on dt_s. Both graphs sampled at 50 Hz show a sharp drop around the same value of $|\Omega_m|$, whereas the one at 25 Hz drops away at half the angular rate. Considering the assumption that the output would be rounded towards zero, it is an unexpected effect that the graphs quickly drop away below the point where they pass the unit correction factor value. In that case, one would expect the raw data to undervalue the actual Ω, thereby causing a higher correction factor close to zero. Clearly another effect dominates the signal, if the rounding issue exists at all. The data sheet of the ADNS-5030 offers no information on the exact process of the flushing of the registers and it is possible that the digital signal processor (DSP) continues an internal count.
The general trend is an increasing correction factor with increasing $|\Omega_m|$. Towards the high angular rates this may be explained by assuming that the sensor "misses" some counts. The counts represent angular displacements, so the sensor might loose track of features occasionally and as a result count less angular displacements. This would result in a lower $|\Omega_m|$ and thus a higher correction factor. This effect is strongest in the DC light source graph. At first sight it would appear that the oscillating light intensity from the TL tubes is a better environment than the constant light from the torch on batteries used as DC light source. However, a more probable explanation is that the overall light intensity from the torch was too low. Although the torch covered the field of view of the sensor as it turned with it, the TL tubes may very well have illuminated the paper surface with a greater mean power flux. The DC light graph doesn't look as symmetric as the other two with TL source, especially at higher angular rates. This supports the idea that the sensor skips some angular displacement counts as this effect would evidently become larger when the surface is less well illuminated. Another factor which is not addressed by these calibration experiments is the light spectrum. The ADNS-5030 is most sensitive to red light, but it does respond to other frequencies as well. Both light sources have a red component in their spectra, but the combined effect of the whole spectrum on the signal from the photoreceptors is not easily calculated.
Concluding, the oscillating light from the TL tubes appears not to pose a big problem for the sensor and light intensity is probably a strong factor in sensor performance. More tests at

various light sources with measured properties should be performed to get conclusive answers to these questions. The calibration data with TL tubes do provide a good means to correct the sensor output as the lighting conditions are similar to the room where the experiment has been carried out.

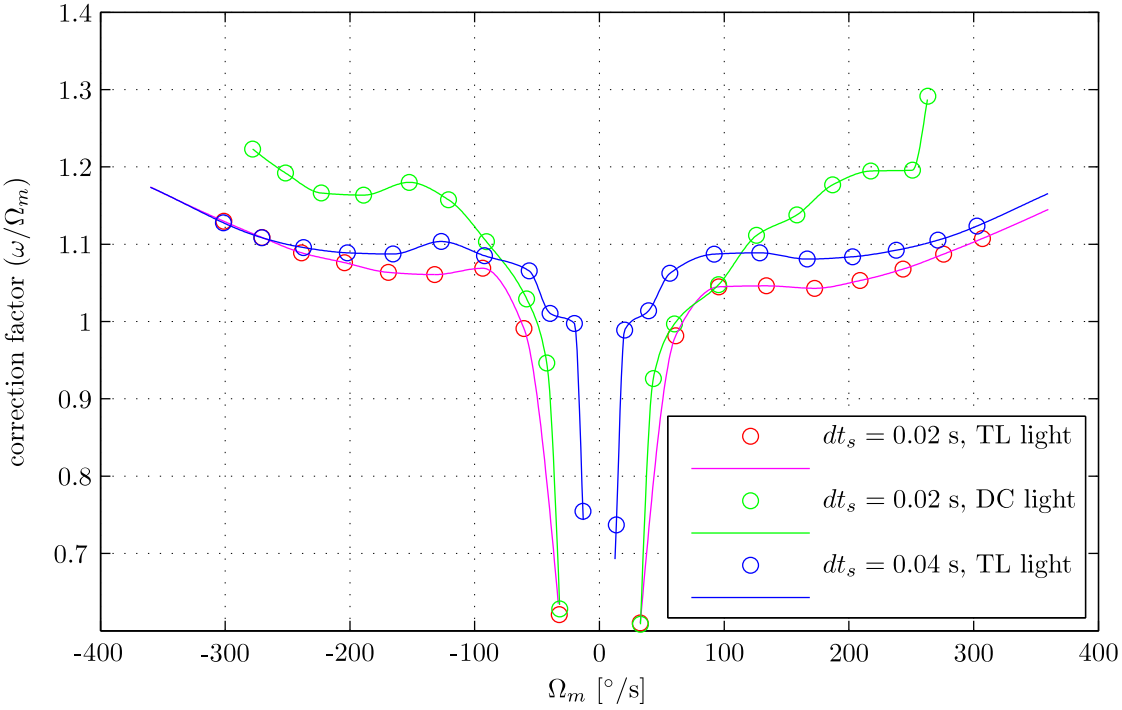


Figure 8-2: The ADNS-5030 OF calibration curve

Chapter 9

Experiment

An important goal of the thesis work is described in this chapter. The experiment was the end result of building, programming and debugging the hardware and the simulation phase preceding that.
First the experiment environment, the geometry of the motion generating pendulum and the data logging setup is described. Then the results are discussed referring to the corresponding graphs included with the appendix. Some specific conclusions are drawn based on these results, leaving the overall conclusions to Chapter 10.

9-1 Setup

The experiment's aim is to investigate the optical flow (OF) sensor concept through a hardware implementation in a realistic environment. The environment is an office room at ASTI shown in Figure 9-1. Its dimensions are approximately $3 \times 4 \times 2.6$ m, it is lit by TL tubes and it has two windows at one end. The motion pattern was chosen to be a pendulum for the following reasons:

- It has to be a well-reproducible motion with easily identifiable pattern and parameters for validation purposes. The pendulum produces a sinusoidal motion pattern with known period and decreasing amplitude. This can be easily recognised in the estimated states and it can be simulated to provide a comparison.
- The pendulum is autonomous, i.e. it requires no input after the motion has started. This means that the moments acting on the pendulum mass don't have to be measured, which simplifies the experiment and eliminates a noise source.
- The pendulum provides changing velocities and rotation in the range of a typical indoor micro aerial vehicle (MAV) with momentary zero velocity points at the extremes of the pendulum swing arc. This should provide interesting data to test the Kalman filters and check the observability.

The pendulum consists of two strings attached to the ceiling at one end and to each other at the lower end. The angle between the two strings ensures that the motion stays within one plane. A short string connects the triangle with a mass. This mass is a solid steel cylinder weighing 1.13 kg. It ensures that the pendulum keeps oscillating long enough to record a good dataset. The effective pendulum arm is 1.45 m.
The sensor board is connected to the pendulum at the string junction and also to the data cable, which runs along one of the strings to the ceiling and then to the laptop PC. Its suspension is such that it keeps its attitude with respect to the pendulum during its motion.
The camera output was monitored during test runs to align the dotted paper with the trajectory. These remained in the same place also indicating that the sensor board kept its orientation.
Before starting each run, the weight was held by an extra line to the wall, in order to get the same starting attitude.

Figure 9-1: The office room at ASTI with the pendulum experiment setup. The dotted paper has been placed in the view of the sensors to enhance the texture and thereby tracking performance.

9-2 Results

During an experiment, the data is logged on a PC using the RS232 serial interface as described in Chapter 7. In order to convert the signal to units of rotational rate it is multiplied by $\delta\Omega$, as described in Section 8-3, immediately after receiving the OF data. This factor is later fine-tuned in Matlab using the calibration graphs from the same section. Before this fine-tuning,

some simple filtering is applied to deal with sensor artefacts which are very non-white noise sources (and therefore hard to deal with in the Kalman filter). An example of the "raw" OF before these filters have been applied is shown in Figure 9-2.

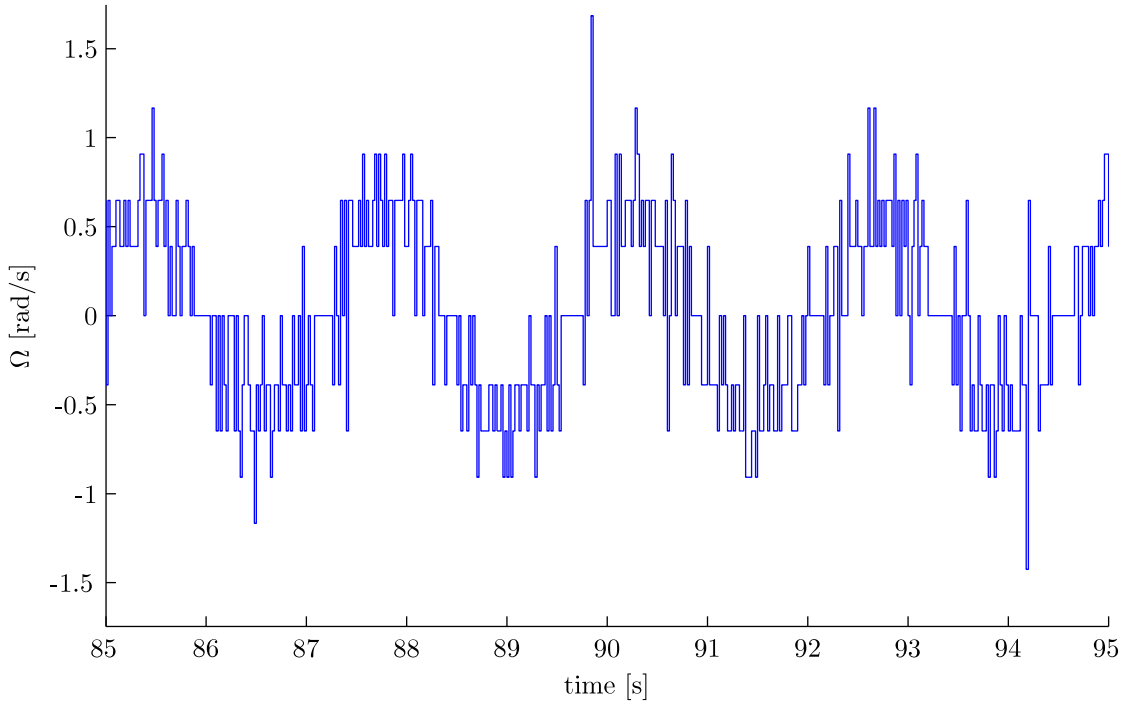


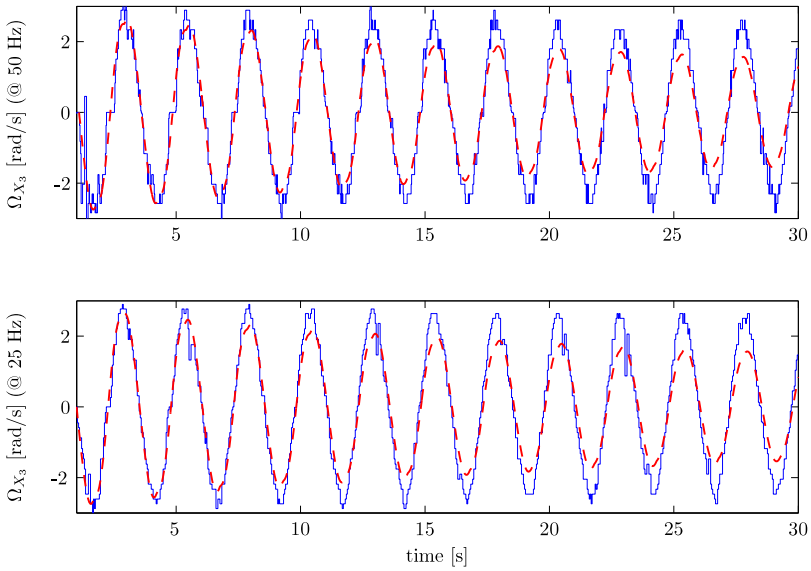
Figure 9-2: OF signal before filtering

The two artefacts mentioned above are

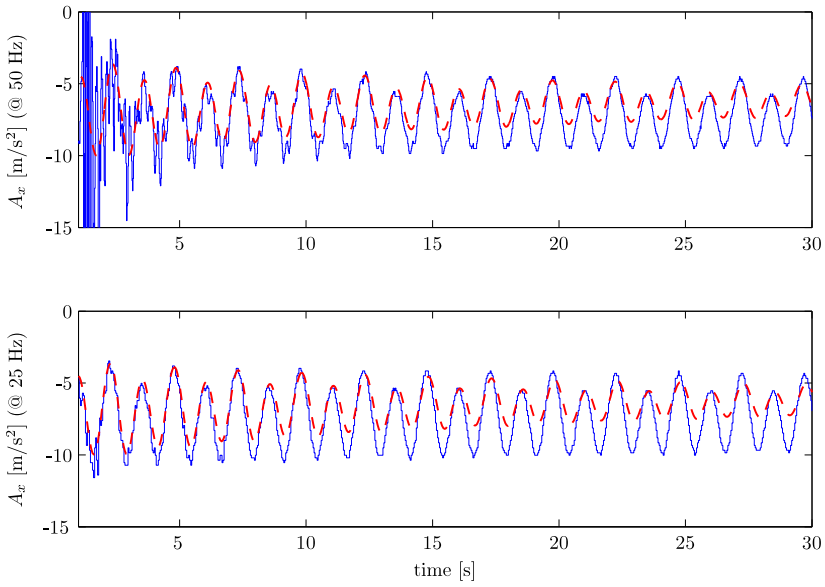
- one timestep high-valued peaks and
- many short duration zero values amid nonzero data.

The second artefact may be caused by the functioning of the register. To filter these, two filter algorithms have been applied. Firstly, the data is checked for zero values with neighbouring nonzero values. These are replaced by the mean of both their neighbouring values. Secondly, a 3-point median filter is applied. This removes any one timestep peaks.
An example of the filtered and calibrated hardware output is shown in Figure 9-3. It shows an OF component and a specific force component from two experiment runs, one sampled at 50 Hz and one sampled at 25 Hz. The hardware output is compared to data from the simulations under similar conditions. Clearly, the pendulum drag term in the simulated data has been set higher than the hardware experiment. Other than that, it is apparent that the hardware output shows more noise and spikes making it more challenging to filter. Also notice that in the 50 Hz OF graph the hardware output exhibits some "sticking" to the zero during the first few oscillations. This phenomenon originates in the ADNS-5030 digital signal processor (DSP). When insufficient movement is perceived in combination with a low number of tracked features, the sensor goes into a "rest" mode, from which it takes time to go back to normal operation mode. The ADNS-5030 has been designed to use as little

power as possible for use with wireless computer mouses. Although the data sheet (Avago, 2008) mentions a rest mode configuration register which allows the user to "force" a rest mode selection (including normal operation mode), this doesn't work in practise because the sensor decides by itself to go back to the rest mode based on the optical input. Especially the sensors which were directed at the ceiling were prone to this flaw, as the ceiling was lit only by a couple glow bulbs, producing less light intensity than the TL tubes.



(a) optical flow (OF) of sensor 3 in sensor board X-direction



(b) specific force component in the sensor board X-direction

Figure 9-3: Sample of the hardware output during a run at 50 Hz and a run at 25 Hz. The hardware signals (solid lines) are compared to simulated data (dashed).

Another observation which can be made from the graph in Figure 9-3 b is that, at first

sight, the accelerometer appears to oscillate at double frequency as compared to the OF sensor. Each minimum of $|A_x|$ coincides with a zero crossing of Ω_{X_3} and thus an extreme of the pendulum path. The maxima of $|A_x|$ coincide with the maxima of $|\Omega_{X_3}|$. This may be expected, as the accelerations will be largest when the pendulum is moving at maximum velocity which happens twice during each period. Looking at the minima of $|A_x|$ again, they show an alternating pattern where a positive Ω_{X_3} zero crossing corresponds with a low valued peak of $|A_x|$ and vice versa. This is because of the rotated attitude of the sensor board with respect to the pendulum motion. The fact that the simulated data shows a very good fit, especially with respect to the alternating peak ratio, indicates that the simulation parameters are quite close to the actual experiment.

The state estimates and standard deviation estimates resulting from application of the three types of Kalman filters (Iterated Extended Kalman filter (IEKF), Unscented Kalman filter (UKF) and Hybrid Kalman Filter (HKF)) to the two representative datasets discussed above, are included in Appendix C.
Comparing IEKF to UKF/HKF, the first observation is that the IEKF shows a much a higher noise level in its state estimates. Also, a number of states, which should be essentially equal to zero, have very significant nonzero values. This is due to the completely wrong attitude estimate of the IEKF. After an initial diverging phase, θ starts oscillating about -180° and φ has a much larger amplitude than θ although it should be zero. The large oscillations in p, which should also remain zero, are a result of the erroneous attitude. Concerning the distance estimates, the IEKF has a strong tendency to produce negative values and the filter solution would completely diverge if not for a condition that the distances can only be positive. Each time a distance value becomes arbitrarily small however, the filter innovation becomes very inaccurate because the predicted OF components are calculated by dividing velocity components by distances, producing very large corrections through the innovation. These have a destabilising effect on the solution.
Comparing the UKF and HKF, the first observation is that they produce very similar results as has been noted in the results discussion of the simulations as well. The common difference with the IEKF is the prediction step, so this part of the filter appears to have the larger influence on the state estimates. The UKF uses the Unscented Transform (UT) for the correction part of the filter as well, but this doesn't produce a big performance gain over the HKF. This result is remarkable, since the prediction is the "simple" part of the filter: it contains only a very straightforward set of general nonlinear rigid body dynamics equations which are integrated using the 4^{th} order Runge-Kutta (RK4) algorithm at a small integration timestep. The correction part of the filter, on the other hand, uses the highly nonlinear observation equations to estimate the distances. These equations are algebraic however. The major difference is the way in which the predicted covariance matrix $P(k+1\,|\,k)$ is calculated: the EKF makes one function evaluation through the RK4 algorithm and uses that in the Jacobian of the dynamics F_x to calculate $P(k+1\,|\,k)$. This ignores all higher order derivatives with respect to the state, thus linearising the relation. The UKF on the other hand evaluates the dynamics for a well defined set of perturbations of the state vector. The resulting predictions for the perturbed states form a transformed set from which $P(k+1\,|\,k)$ can be reconstructed. This method is much more computationally intensive as it requires many RK4 calls per filter timestep, but it yields a more accurate one-step-ahead prediction. This must explain the difference in filter performance as it is the only difference between the

IEKF and HKF.

The condition number $\mathcal{C}_O$ which is defined in Section 3-3-2 as the ratio between the largest and the smallest nonzero singular value of the observability matrix $\mathcal{O}^1$ has also been plotted. The best observability condition occurs when $\mathcal{C}_O = 1$ and higher means harder to observe. The peaks in the $\mathcal{C}_O$-plots of the UKF and HKF coincide with the moments where the velocity crosses zero and the pendulum reverses direction. Theoretically those points are unobservable, as explained in Chapter 3. In practise, the sensors are never sampled at the exact moment of the direction reversal. Moreover, the measurements are too noisy and inaccurate to find a value of the rank of $\mathcal{O}^1$ lower than 14. Therefore the rank condition for observability doesn't give much information in practise. Therefore the peaks in $\mathcal{C}_O$ are a good indicator to signal possible problems with observability. To be sure that the rank condition is met, the other condition number presented in Section 3-3-2, namely $\underline{\mu}\left(\mathcal{O}^1\right)$, is useful. If $\mathcal{C}_O$ would stay above a certain threshold for a longer time, the Kalman filter might well diverge. In this case, temporarily not updating the state could be a way to improve filter stability. The state would still need to be estimated in the background to monitor $\mathcal{C}_O$ though. Alternatively, the condition could be a collective low-value threshold for all OF signals as this would imply a high $\mathcal{C}_O$ as well. The advantage is that in this case, the Kalman filter could be halted.

The standard deviation estimates of the three filters have been plotted together per state. These are included as Figures C-16 - C-19 and C-35 - C-38. Due to convergence problems with the filters with a UT-based prediction part, the main diagonal entries of $P\left(k+1 \,|\, k\right)$ corresponding to the distances have been set to constant predetermined values. This yielded better filter results for the UKF and HKF. This may be partly explained by the fact that the equations of motion (EoM) contain zero derivatives for the distances, because they are unknown. No new information is available, so the predicted covariance cannot be expected to be very accurate. The update part doesn't change the distance covariances much as the corresponding standard deviations remain mostly constant in the graphs.
For a properly converging filter solution, the estimated standard deviations should show a decreasing trend towards a horizontal asymptote. The IEKF doesn't show any decreasing trends. Rather, it shows mostly constant but noisy values for $\mathbf{V}$, $\vec{\omega}$, θ and φ. At 25 Hz, φ has a lot of high peaks which correspond to the direction reversals of the pendulum. The distances show periodic divergence followed by a sudden "reset" to a small value. At 25 Hz, d_4 even completely diverges. This means both high values of d_4 and of σ_{d_4}. This is accompanied at the same time by a very low value of d_5. The distances would have diverged much faster without the absolute value constraint. These results indicate that the IEKF is not converging and is performing very poorly.
The standard deviations from the UKF and HKF algorithms do suggest some convergence in the velocities, rotations and in θ. The periodicity of the source data manifests itself in most graphs through oscillations with additionally some peaks in φ. These coincide with the three highest peaks of $\mathcal{C}_O$ in the 50 Hz HKF case. Generally, there is a strong correlation between the standard deviations and the observability condition numbers.

Another peculiar phenomenon emerging in the UKF/HKF estimates is a "beat" in the velocities, in q and in θ. This low frequency oscillation in the amplitude of these signals is of course completely absent in the true motion. Some internal feedback might interfere with the filter estimates. The pendulum has been modelled as an autonomous system, so the moment about

the Y-axis which rotates the pendulum weight is being calculated from the accelerometer measurements and the pith angle as follows:

$$\mathcal{M}_y = \frac{(A_x - g\sin\theta)\, I_y}{R} \tag{9-1}$$

This happens for each timestep of the RK4 integration using interpolated values of A_x and the value of θ from the previous timestep within the integration algorithm (which spans one filter timestep using a number of smaller timesteps to integrate the dynamics). This will make the filter less stable compared to the case when $\vec{\mathcal{M}}$ is available as input. The simulation results, discussed in Section 6-2, do not show the beat however even though they have been calculated using the same filter algorithms. Perhaps the lower quality of the hardware experiment data in combination with the lower filter stability causes the beat to appear.
Aliasing, the artificial addition of low frequency content due to sampling without proper low pass filtering, may also contribute to the beat. A combination of low filter stability and aliasing could produce a resonance effect. Although the simulation attempts to model aliasing, it may be very different from the hardware due to different noise characteristics.
The magnitudes of the signals affected by the beat are generally too large. The minima of the amplitudes in these signals are close to the simulated amplitudes of the corresponding states. Whatever is causing the beat is adding artificial energy to the solution. Perhaps the addition of a total energy term to the state in combination with a condition allowing only decrease would stabilise the filter. It would be hard to keep the filter applicable to more general motion cases, but the addition of Eq. (9-1) has made the filter specific already.
Comparing the results from the data set recorded at 50 Hz with those from the data set recorded at 25 Hz, the following may be noted: In general, the 25 Hz data yields higher errors and less stability. It aggravates the beat phenomenon. This means that, although the resolution in Ω is better at 25 Hz, the higher time resolution is preferable in this comparison. It may very well be that using two different sample rates for the accelerometer and the OF sensors may be the best compromise. The correction step in the Kalman filters would have to be rewritten and the accelerometer data would be used only in the prediction step. It would mean a drastic change of the filter algorithms and of the software of the microcontroller (μC). This was not possible within the scope of this thesis work.

Chapter 10

Conclusions & Recommendations

From the literature study on insect motion perception and flight control in the introduction, it can be concluded that these species use optical flow (OF) extensively to successfully navigate complex indoor environments. This thesis proposes an OF-based approach for the flight control and obstacle avoidance of indoor MAVs. The implementation of OF sensors in the proposed concept is driven by the factors availability, low cost, and light weight. It should be noted that other approaches, e.g. using wide angle cameras, are also possible. This would require more processing power and make the sensor package more expensive, larger and heavier and therefore be less practical.

It has been the goal of this M.Sc. work to investigate a design which should be relatively easy to build and test using real hardware. In this way, the theory and simulation results have been compared to real world performance. This should give valuable information about the validity of concepts like observability and simulation results. That is perhaps a more important goal than the design of a particular sensor concept itself.

Observability analysis indicates that the concept with 6 orthogonally mounted OF-sensors and 3 accelerometers should provide sufficient output to observe the platform motion (u, v, w, p, q, r), the sensor-obstacle distances ($d_1 - d_6$) and two attitude angles (θ, φ) as long as the platform is moving relative to a stationary environment. The accelerometer biases are not observable though, which has been alleviated by calibrating the accelerometers prior to each set of experiment runs. Figure 9-3 b shows that the calibration is working quite well as the hardware output fits well with the simulation data (which had zero bias).

The result of the simulation clearly indicates that the use of the Unscented Transform (UT) in a Kalman filter yields better estimates for this problem. The largest effect is achieved in the prediction step. This can be concluded, because the Unscented Kalman filter (UKF) and Hybrid Kalman Filter (HKF) algorithms share the same prediction part based on the UT and they have very comparable performance, while the Iterated Extended Kalman filter (IEKF) uses a linearised prediction step and it produces much larger errors. The use of the UT in the update step, as is done in the UKF, doesn't provide clear performance benefits. This is an unexpected finding as the nonlinear nature of the observation equations, including coordinate transformation and velocities and distances only appearing as ratios, would suggest that an algorithm with higher order probability distribution capture would produce better results.

The HKF, using iterations with UT-based predicted state and covariance, appears to yield the best results.
The convergence zone of the IEKF appears to be smaller than that of the other algorithms, so the initial conditions have to be chosen closer to the true values to get results which don't diverge. The cost of the improved accuracy of the UT-based filters is about 6 times longer computation time.
Theory indicates that the problem is observable. However, many factors affect the quality of the solution. Observability does not imply a useful result or even a converging one. That depends on the requirements of the application, the quality of the sensors, many conditions of the environment, such as lighting, texture on obstacles, the scale of the room, and it depends on the filter algorithm. The real world data suffers from many deteriorating factors, such as noise, limited resolution, sample rate and manufacturing inaccuracies. These factors cause the problem and its solution to be stochastic in nature, meaning that it is never equal to the true values, and can be close at best. Therefore the theoretical statement that an estimation problem is observable only provides a starting point. In principle, the solution can be reconstructed from the observations, given perfect signals and an exact process model. It says nothing about the achievable quality of the data.
A useful quantity which can be derived from the concept of observability, is the observability condition number $\mathcal{C}_O$. Both the simulation and the hardware experiment results clearly show that $\mathcal{C}_O$ peaks (indicating a hard to observe motion state) when the velocity approaches zero. The error of the pendulum velocity estimate may be driven by the observability condition. The standard deviations show a clear correlation with $\mathcal{C}_O$, indicating that filter performance is impacted by the observability condition.
The simulation and hardware experiment show that even with an extremely low cost sensor package, the stated problem may yield converging results. The errors in these results do get larger when the quality of the measurement data deteriorates. Given the fact that the sensors needed a lot of light and very large contrast in the texture to work, it is fair to say that a practical application onboard of an micro aerial vehicle (MAV) would require much better quality sensors to work in generic indoor environments.
Also, the processing power requirements for the Kalman filter are significant. The UKF/HKF have run about realtime on an intel core 2 duo machine. This was in Simulink's "Rapid Accelerator" mode, which does compile the code to run it quicker. However, an implementation of the algorithm in C++ may produce better results. Still, to fit sufficient computation power onboard of a small, lightweight vehicle with very limited power available is expected to be very hard at best. A more realistic scenario would be a datalink and off-board processing of the filter together with a controller. The control commands could then be send back to the vehicle. This scenario has been successfully demonstrated by (Berkelaar & Oonk, 2009), who have used off-board image processing to extract the attitude of their quadrotor from images containing laser dots.
For application in an indoor MAV, this sensor concept requires the aerodynamic moments applied to the vehicle as inputs. The confined space usually means that an indoor MAV is essentially in a hover state most of the time. Then these moments can be directly derived from the control inputs, with negligible dependence on the motion state. If the moments do depend strongly on the motion of the vehicle, they would have to be estimated in the Kalman filter using the control commands as inputs. This would alter the problem and would require more tests. Addition of gyroscopes to the sensor board may in that case be an option, although these have not been included originally because the rotational rates can be tracked

very well using the OF sensors. Gyroscopes also introduce additional states because their bias errors must be estimated as well.
Better quality sensors and an addition to the configuration may improve the filter performance to useful levels. In principle, the UT based Kalman filter has shown to be capable of estimating the motion state of an indoor MAV using OF sensors and accelerometers. The example of flying insects indicates that a lot is possible. Their completely different data processing structure with highly parallel neuron network would be an interesting research topic. However, insects don't know their absolute velocities and that is where this work has diverged from the path that nature has chosen. It turned out to be difficult to estimate the velocity and distances as individual states. A promising research path would be to look at control concepts which can handle translational optical flow directly.

Bibliography

Anderson, N. A. (1998). Instrumentation for Process Measurement and Control. In (3rd ed., chap. 2). CRC Press.

Aubépart, F. & Franceschini, N. (2007). Bio-inspired Optic Flow Sensors based on FPGA: Application to Micro-Air-Vehicles. *Microprocessors and Microsystems*, *31*, 408 – 419.

Avago. (2006, August). *Adns-2620 datasheet* (5989-3098EN ed.; microcontroller manual). Avago Technologies Ltd. (`http://www.avagotech.com/docs/AV02-1115EN`)

Avago. (2008, September). *Adns-5030 datasheet* (AV02-0113EN ed.; microcontroller manual). Avago Technologies Ltd. (`http://www.avagotech.com/docs/AV02-0113EN`)

Baird, E., Srinivasan, M. V., Zhang, S., Lamont, R. & Cowling, A. (2006). Visual Control of Flight Speed and Height in the Honeybee. *Lecture Notes in Computer Science LNAI*, *4095*, 40 – 51.

Barber, D. B., Griffiths, S. R., McLain, T. W. & Beard, R. W. (2007, May). Autonomous Landing of Miniature Aerial Vehicles. *Journal of Aerospace Computing, Information and Communication*, *4*(5), 770 – 784.

Barbour, N. M. (2009, May). Inertial Navigation Sensors. In *RTO-EN-SET-116.* NATO Research and Technology Organisation.

Barron, A. & Srinivasan, M. V. (2006, March). Visual Regulation of Ground Speed and Headwind Compensation in Freely Flying Honey Bees (Apis mellifera L.). *Journal of Experimental Biology*, *209*(5), 978 – 984.

Berkelaar, W. & Oonk, A. (2009). *Design of a Laser Attitude and Altitude System for Quadrotor UAV Control* (Master of Science Thesis). Delft University of Technology.

Breakwell, J. V. (1967, November). *Estimation with Slight Non-Linearity.* (Unpublished communication, TWR Systems)

Brown, R. G. & Hwang, P. Y. C. (1997). *Introduction to Random Signals Analysis and Applied Kalman Filtering* (3rd ed.). Wiley.

Chen, Z. (1991, December). Local Observability and its Application to Multiple Measurement Estimation. *IEEE Transactions on Industrial Electronics*, *38*(6), 491 – 496.

Chowdhary, G. & Jategaonkar, R. (2006, August). Aerodynamic Parameter Estimation from Flight Data Applying Extended and Unscented Kalman Filter. *AIAA Atmospheric Flight Mechanics Conference and Exhibit*.

Chu, Q. P. (2006). *Course Notes: Modern Flight Test Technologies and System Identification.* (Faculty Aerospace Engineering, Delft University of Technology)

David, C. T. (1982, December). Compensation for Height in the Control of Groundspeed by Drosophila in a New, 'Barber's Pole' Wind Tunnel. *Journal of Comparative Physiology*, *147*(4), 485 – 493.

Denham, W. F. & Pines, S. (1966). Sequential Estimation when Measurement Function Nonlinearity is Comparable to Measurement Error. *AIAA J.*, *4*(6), 1071 – 1076.

Driels, M. R. & Pathre, U. S. (1990, April). Significance of Observation Strategy on the Design of Robot Calibration Experiments. *Journal of Robotic Systems*, *7*(2), 197 – 223.

Fleet, D. & Weiss, Y. (2005). Mathematical Models in Computer Vision: The Handbook. In (chap. 15: Optical Flow Estimation). Springer. (pp. 239 – 258)

Franceschini, N., Ruffier, F. & Serres, J. (2007, February). A Bio-Inspired Flying Robot Sheds Light on Insect Piloting Abilities. *Current Biology*, *17*, 329 – 335.

Franz, M. O., Chahl, J. S. & Krapp, H. G. (2004). Insect-Inspired Estimation of Egomotion. *Neural Computation*, *16*, 2245 – 2260.

Golub, G. H. & Loan, C. F. V. (1996). *Matrix Computations* (3rd ed.). Baltimore, MD: The Johns Hopkins Univ. Press.

Grewal, M. S. & Andrews, A. P. (1993). *Kalman Filtering; Theory and Practice.* Englewood Cliffs: Prentice Hall.

Hedrick, J. K. & Girard, A. (2005). Control of Nonlinear Dynamic Systems: Theory and Applications. In (chap. 6). University of California at Berkeley.

Hermann, R. & Krener, A. (1977, October). Nonlinear Controllability and Observability. *IEEE Transactions on Automatic Control*, *AC-22*(5), 728 – 740.

Hong, S., Chun, H. H., Kwon, S. H. & Lee, M. H. (2008, January). Observability Measures and their Application to GPS/INS. *IEEE Transactions on Vehicular Technology*, *57*(1), 97 – 106.

IARC. (2009). International Aerial Robotics Competition 2009 Website.. `http://iarc.angel-strike.com/`.

Ibbotson, M. R. (1991). A Motion-Sensitive Visual Descending Neuron in Apis Mellifera Monitoring Translatory Flow-Fields in the Horizontal Plane. *Journal of Experimental Biology*, *157*, 573 – 577.

Jazwinski, A. H. (1970). *Stochastic Processes and Filter Theory* (Vol. 64). Academic Press. (Mathematics in Science and Engineering series)

Julier, S. & Uhlmann, J. K. (2001). Handbook of Multisensor Data Fusion. In (chap. 13: Data Fusion in Nonlinear Systems). CRC Press LLC.

Julier, S. J. & Uhlmann, J. K. (1997). A New Extension of the Kalman Filter to Nonlinear Systems. *Proceedings of SPIE*, *3068*, 182 – 193.

Julier, S. J. & Uhlmann, J. K. (2004, March). Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, *92*(3), 401 – 422.

Kailath, T. (1968). An Innovation Approach to Least-Squares Estimation. In *Transactions on Automatic Control* (pp. 16(6), 646 – 660).

Kailath, T., Sayed, A. H. & Hassibi, B. (2000). *Linear Estimation*. Upper Saddle River, New Jersey: Prentice-Hall.

Kalman, R. E. (1960, March). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35 – 45.

Krapp, H. G. & Hengstenberg, R. (1996, December). Estimation of Self-motion by Optic Flow Processing in Single Visual Interneurons. *Nature*, *384*(6608), 463 – 466.

Kushner, H. J. (1967, April). Dynamical Equations for Optimal Nonlinear Filtering. *Journal of Differential Equations*, *3*(2), 179 – 190.

Lefebvre, T., Bruyninckx, H. & De Schutter, J. (2004, May). Kalman Filters for Non-Linear Systems: a Comparison of Performance. *International Journal of Control*, *77*(7), 639 – 653.

Lis302dl datasheet [Computer software manual]. (2007). `http:\www.st.com`.

Mulder, J. A., Staveren, W. H. J. J. van & Vaart, J. C. van der. (2000, August). *Flight Dynamics*. Lecture Notes, Delft University of Technology.

Neumann, T. R. & Bulthoff, H. H. (2001). *Insect Inspired Visual Control of Translatory Flight*.

Springer-Verlag Berlin Heidelberg.

Olsder, G. J. & Woude, J. W. van der. (2005). *Mathematical Systems Theory* (3rd ed.). Leeghwaterstraat 42, Delft, the Netherlands: VSSD.

Preiss, R. & Kramer, E. (1984). Localization and Orientation in Biology and Engineering. In (chap. Control of Flight Speed by Minimization of the Apparent Ground Pattern Movement). Springer Verlag. (pp. 140 – 142)

Ray, S. F. (2002). *Applied Photographic Optics* (3rd ed.). Oxford: Focal Press.

Ruffier, F. & Franceschini, N. (2005). Optic Flow Regulation: the Key to Aircraft Automatic Guidance. *Robotics and Autonomous Systems*, *50*, 177 – 194.

Schenato, L., Wu, W. C. & Sastry, S. (2004, February). Attitude Control for a Micromechanical Flying Insect via Sensor Output Feedback. *IEEE Transactions on Robotics and Automation*, *20*(1), 93 – 106.

Schmidt, G. T. (2009, May). INS/GPS Technology Trends. In *RTO-EN-SET-116.* NATO Research and Technology Organisation.

Serres, J., Ruffier, F. & Franceschini, N. (2006). Two Optic Flow Regulators for Speed Control and Obstacle Avoidance. *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, *2006*(1639180), 750 – 757.

Srinivasan, M., Zhang, S., Chahl, J., Barth, E. & Venkatesh, S. (2000). How Honeybees make Grazing Landings on Flat Surfaces. *Biological Cybernetics*, *83*(3), 171 – 183.

Srinivasan, M. V. (2006). Small Brains, Smart Computations: Vision and Navigation in Honeybees, and Applications to Robotics. *International Congress Series*, *1291*, 30 – 37.

Verhaegen, M. & Verdult, V. (2007). *Filtering and System Identification* (first ed.). New York: Cambridge University Press.

Walcott, B. L., Corless, M. J. & Żak, S. H. (1987). Comparative Study of Non-Linear State-Observation Techniques. *International Journal of Control*, *45*(6), 2109 – 2132.

Welch, G. & Bishop, G. (2001). *An Introduction to the Kalman Filter.* `http://www.cs.unc.edu/~welch/publications.html`. (University of North Carolina at Chapel Hill)

Zufferey, J. C. & Floreano, D. (2005, April). Toward 30-gram Autonomous Indoor Aircraft: Vision-based Obstacle Avoidance and Altitude Control. In *Proceedings of 2005 IEEE International Conference on Robotics and Automation* (p. 2594 – 2599). Barcelona, Spain.

Part IV

Appendices

Appendix A

Helix Simulation Graphs

This appendix section contains the results from one simulation run using the helical path trajectory.
First, the simulation output is shown, including the moment vector $\vec{\mathcal{M}}$ used as input, the accelerometer output $\vec{A}$, the optical flow signals Ω and the vehicle states $\vec{V}$, $\vec{\omega}$, θ, φ and $d_1 - d_6$.
This is followed by the four sample cases as described in Section 6-1. In order to compare filter performance, the resulting state estimates from the three Kalman filter algorithms have been plotted together as errors with respect to the true states.
The discussion of these results is included with Section 6-2 on page 64.

A-1 Input, Measurements and the State

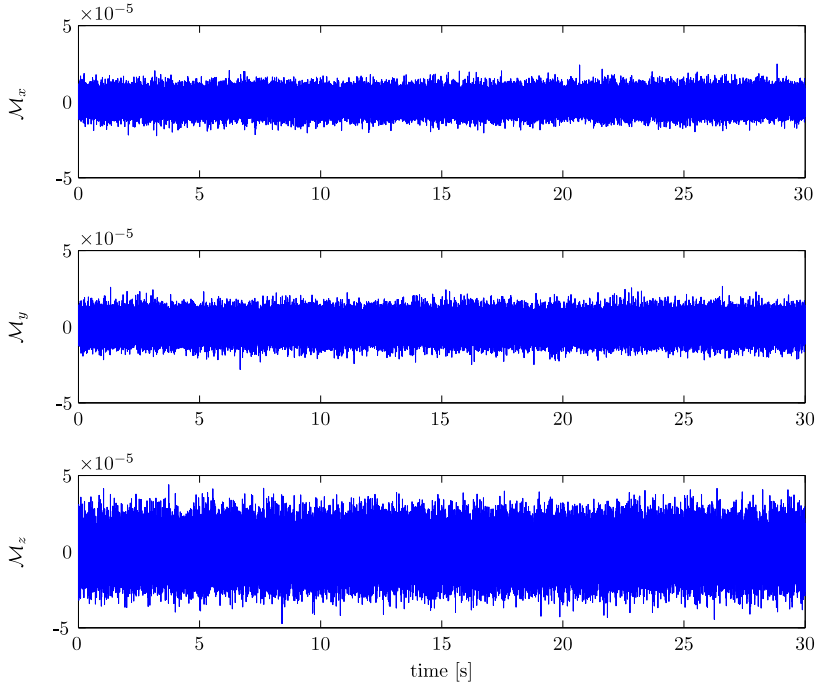


Figure A-1: $\vec{\mathcal{M}}$

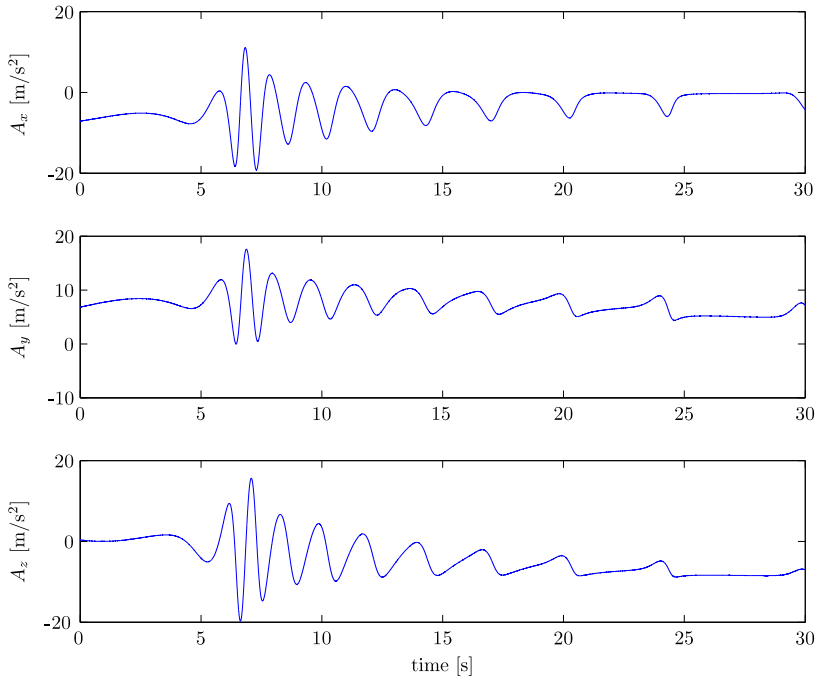


Figure A-2: $\vec{A}$

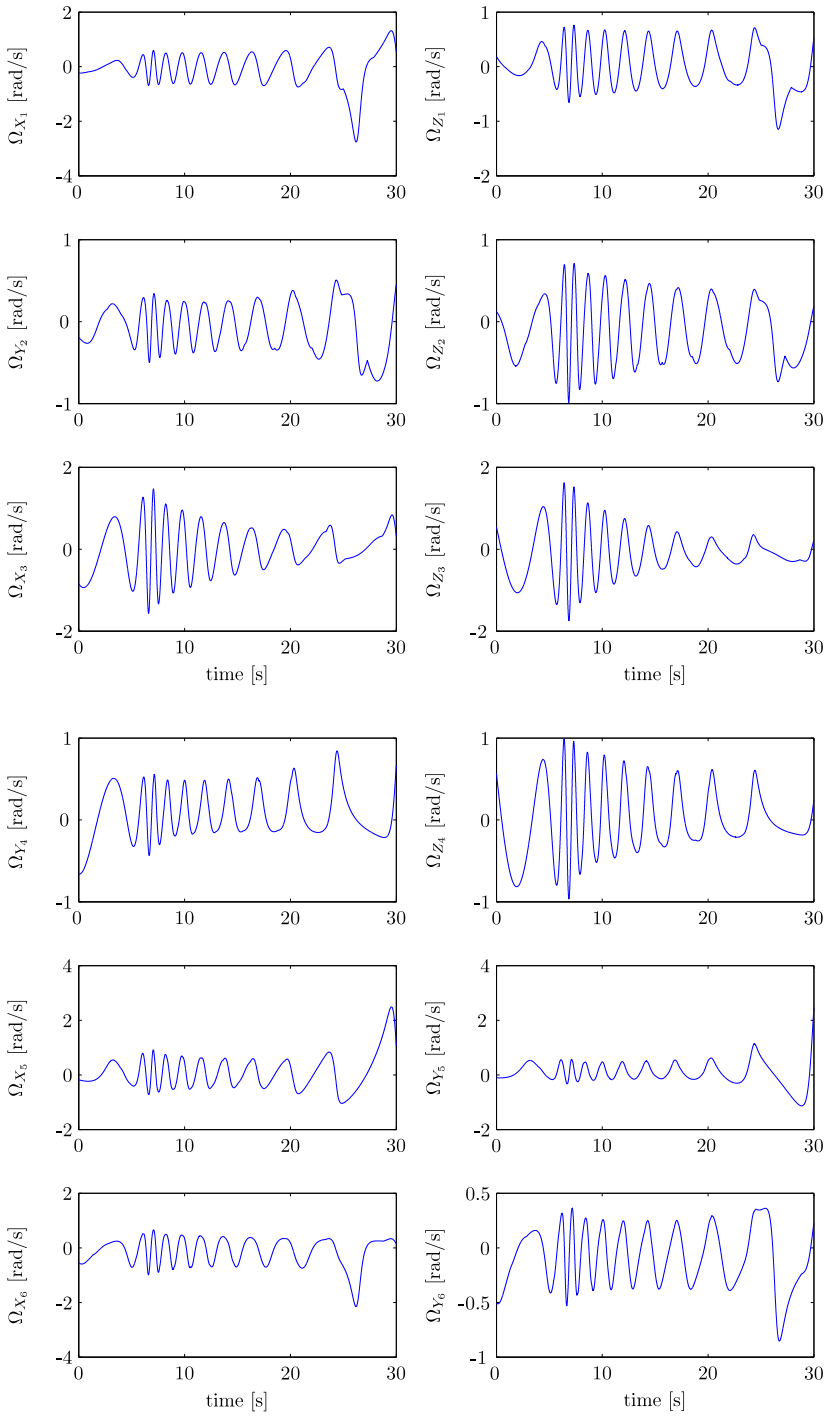


Figure A-3: Ω

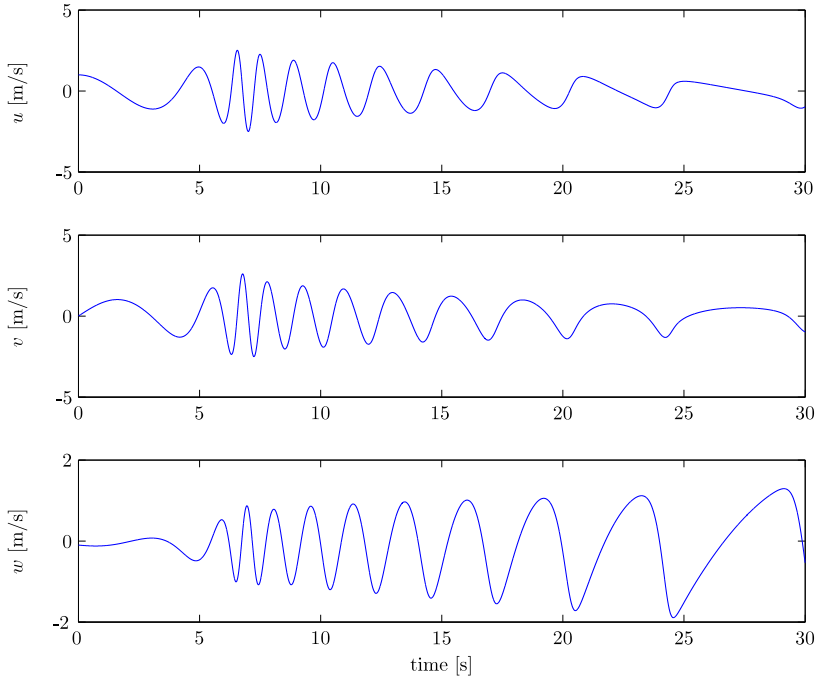


Figure A-4: $\vec{V}$

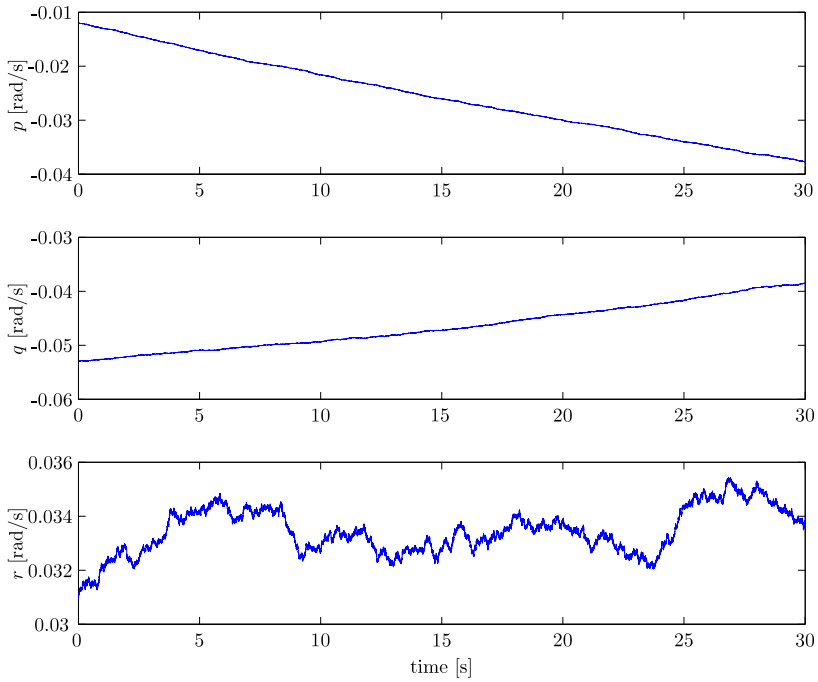


Figure A-5: $\vec{\omega}$

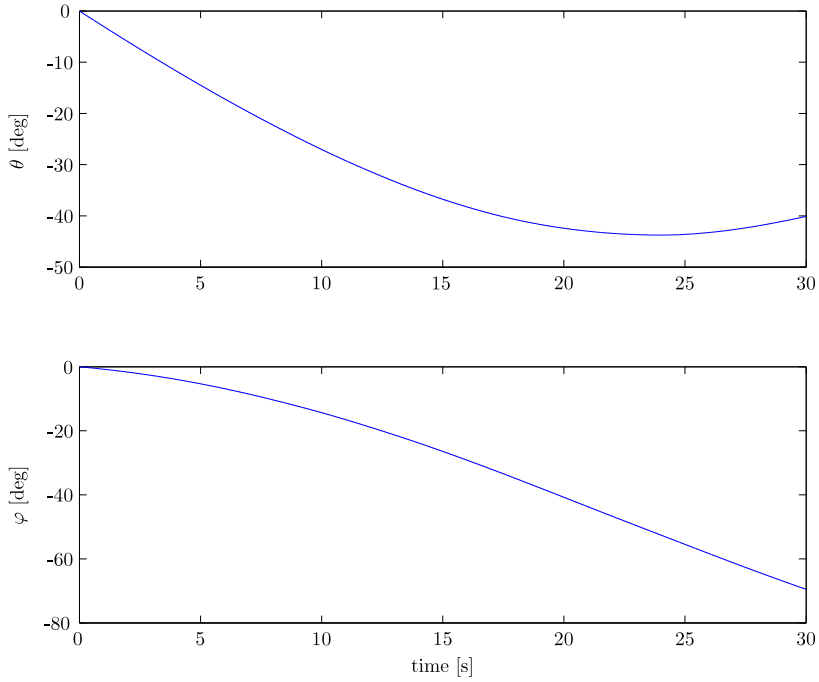


Figure A-6: θ & φ

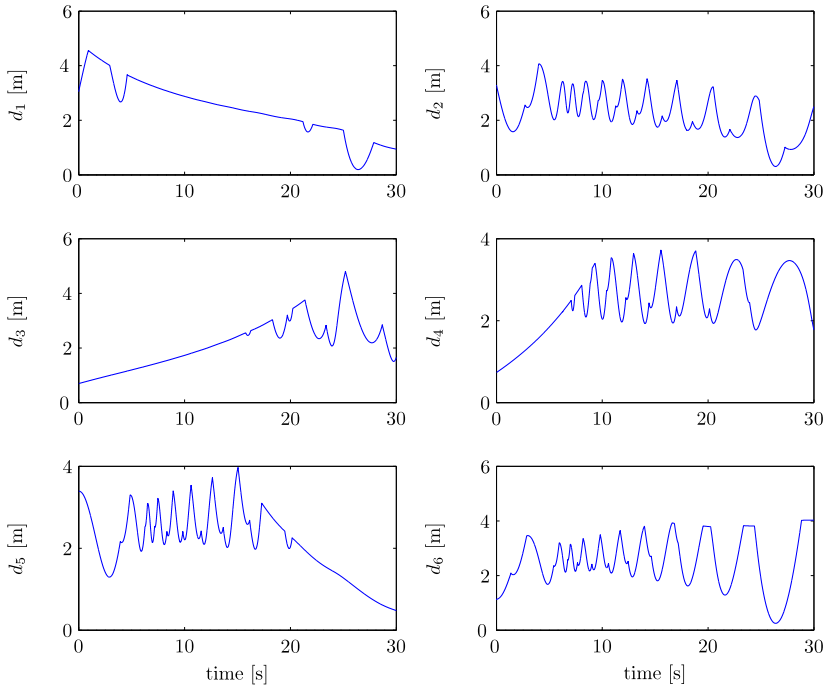


Figure A-7: d_1 – d_6

A-2 Sample Case A

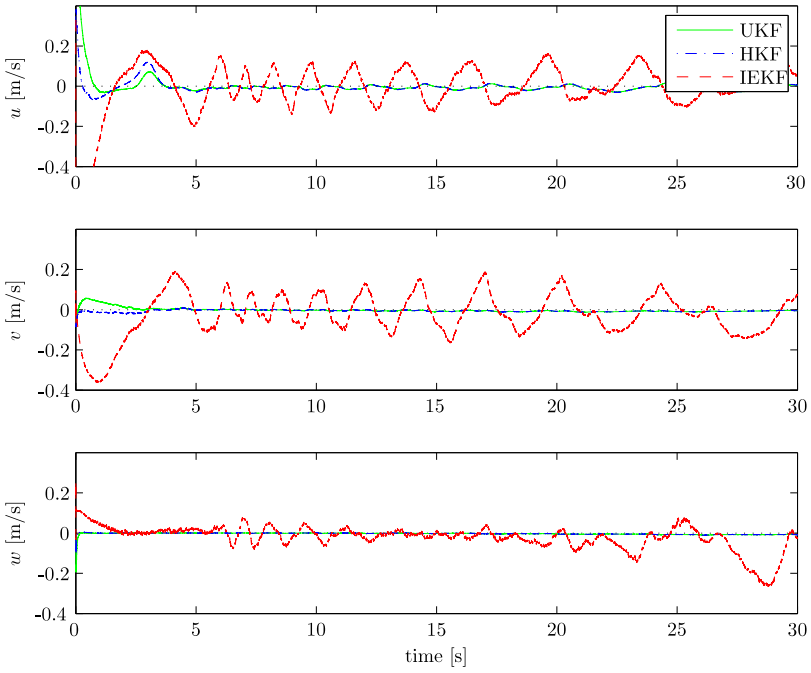


Figure A-8: $\vec{V}$ estimation errors

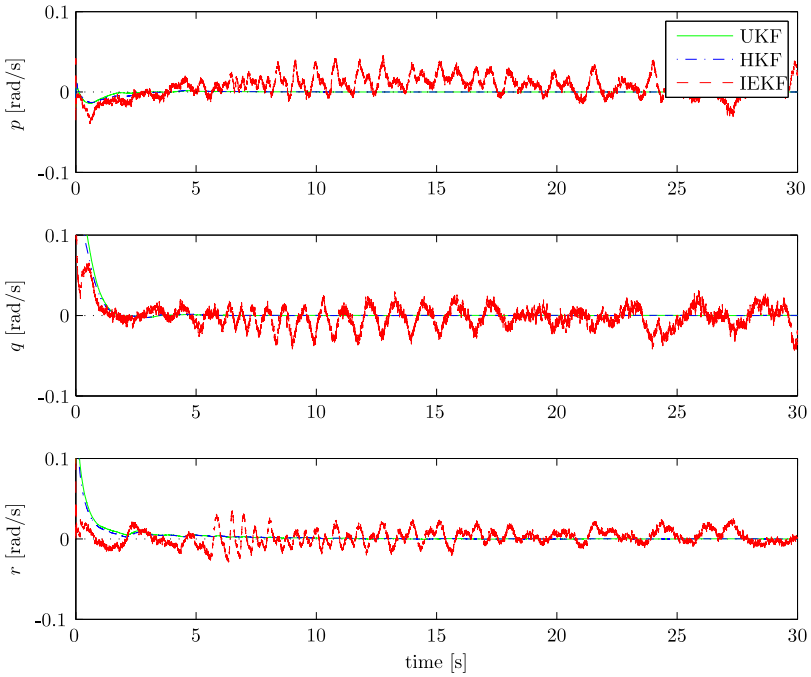


Figure A-9: $\vec{\omega}$ estimation errors

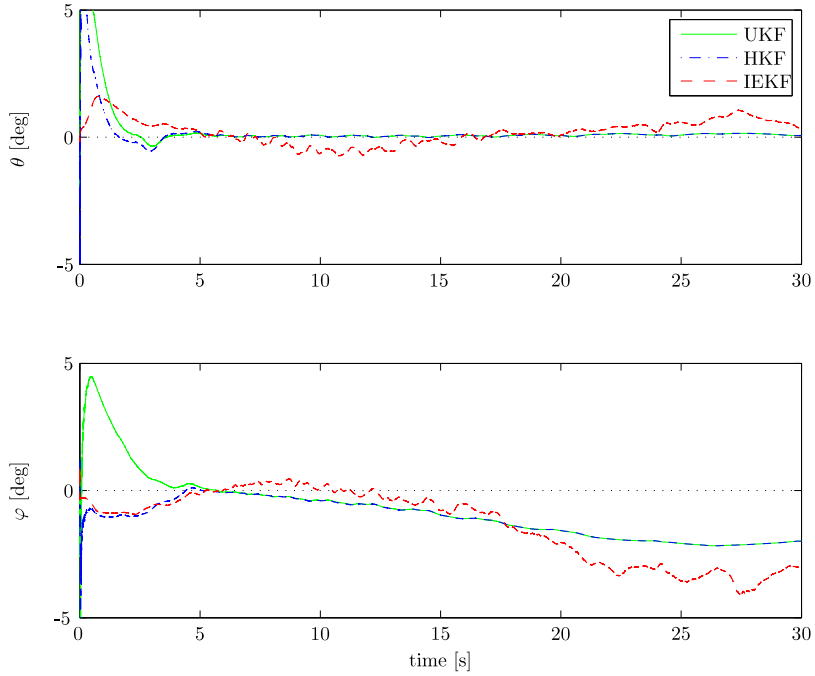


Figure A-10: θ & φ estimation errors

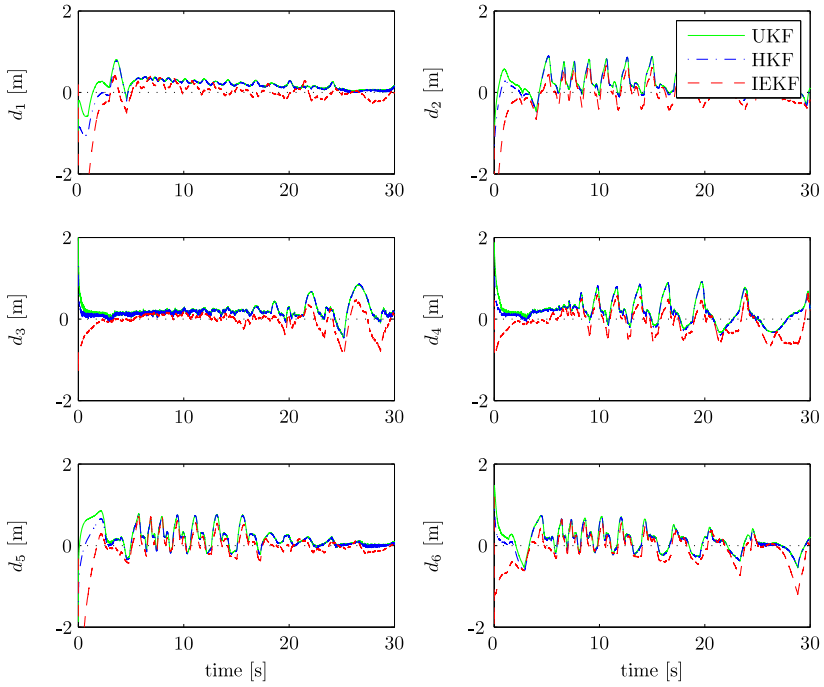


Figure A-11: d_1 – d_6 estimation errors

A-3 Sample Case B

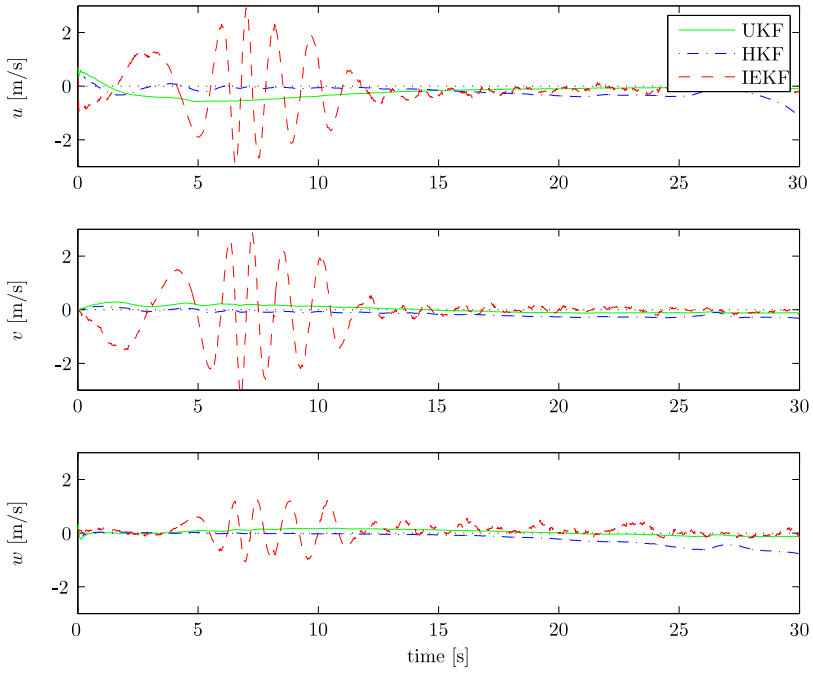


Figure A-12: $\vec{V}$ estimation errors

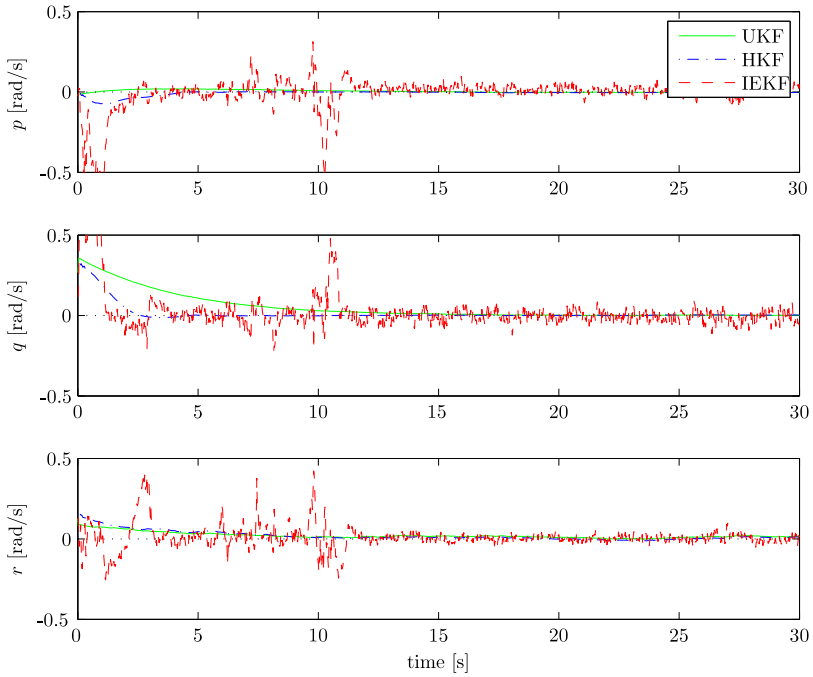


Figure A-13: $\vec{\omega}$ estimation errors

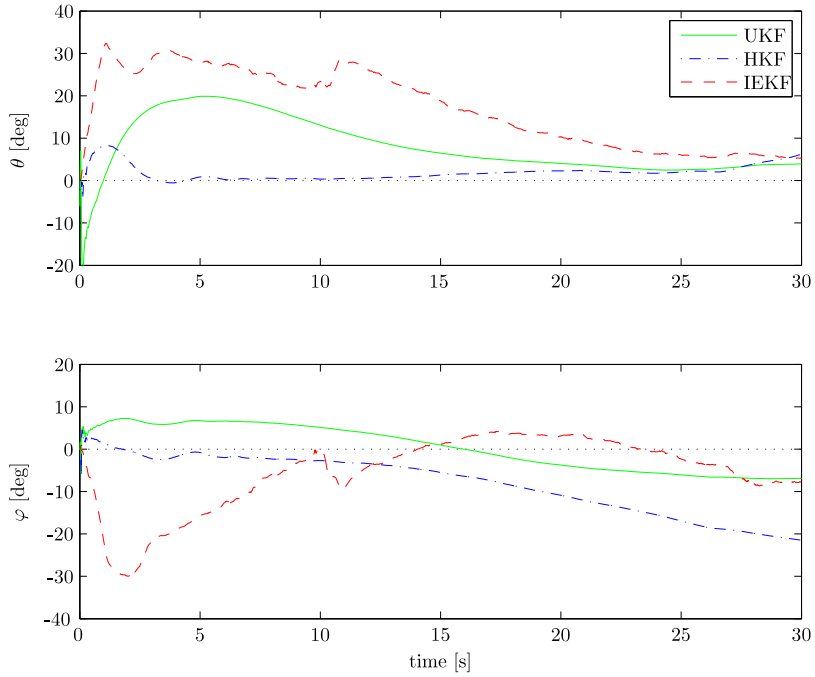


Figure A-14: θ & φ estimation errors

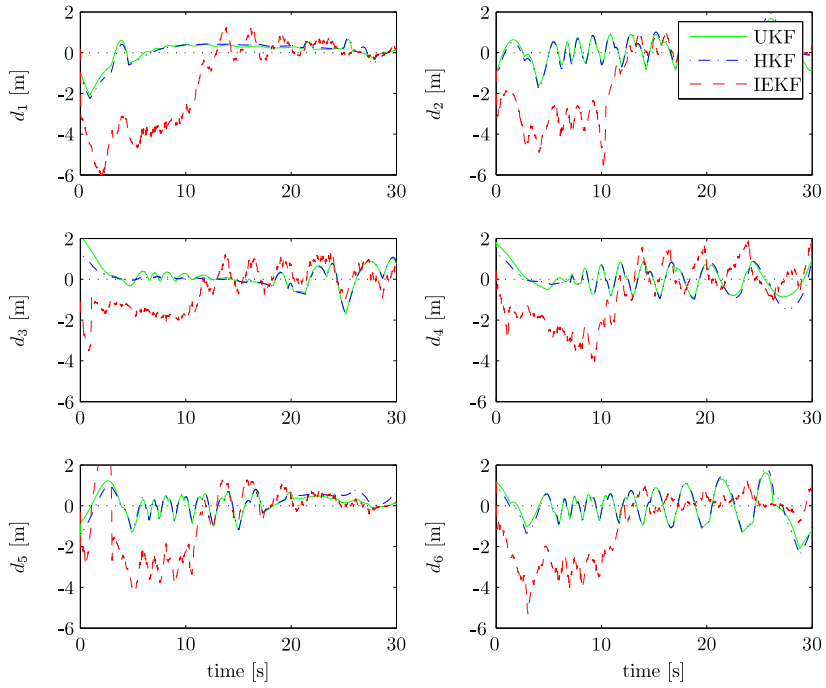


Figure A-15: d_1 – d_6 estimation errors

A-4 Sample Case C

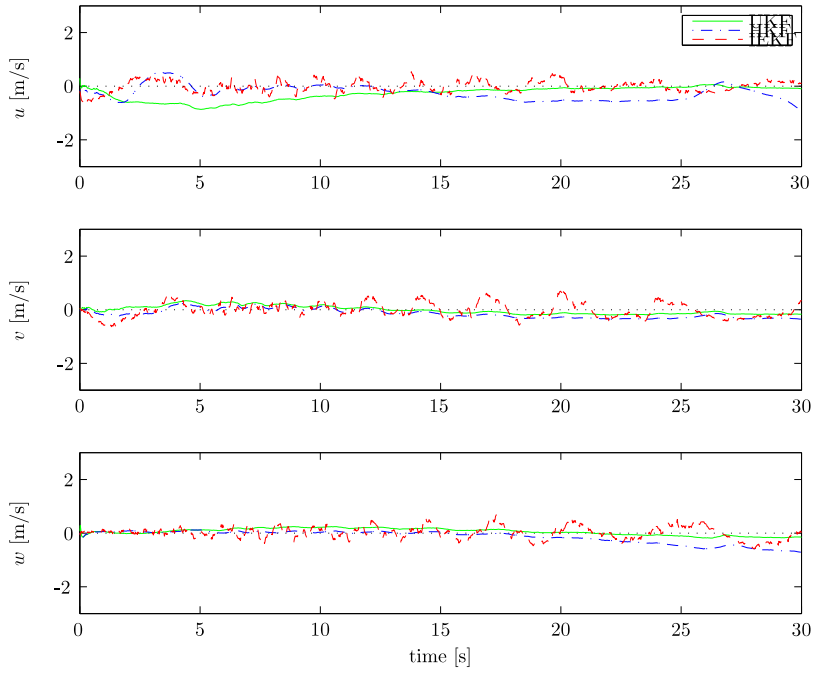


Figure A-16: $\vec{V}$ estimation errors

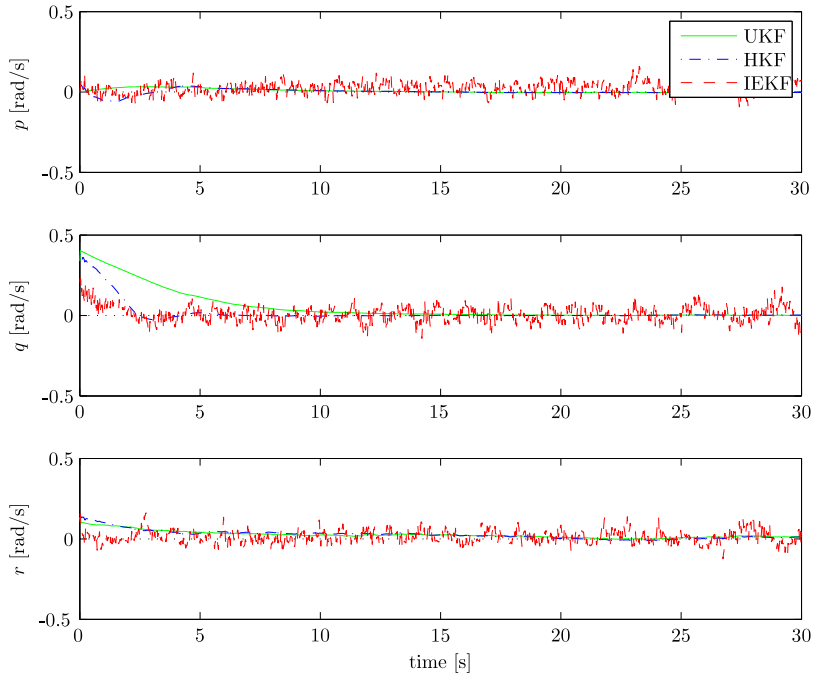


Figure A-17: $\vec{\omega}$ estimation errors

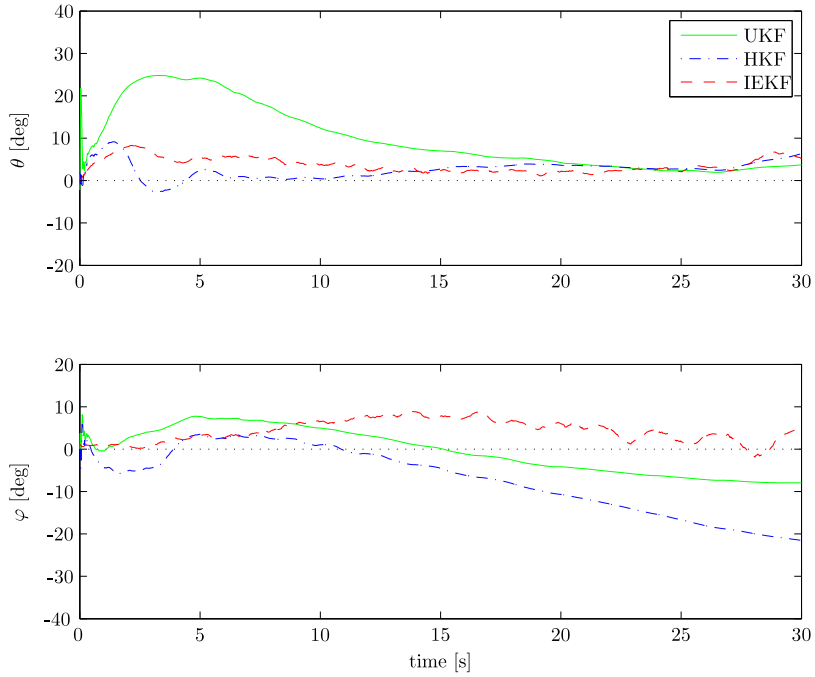


Figure A-18: θ & φ estimation errors

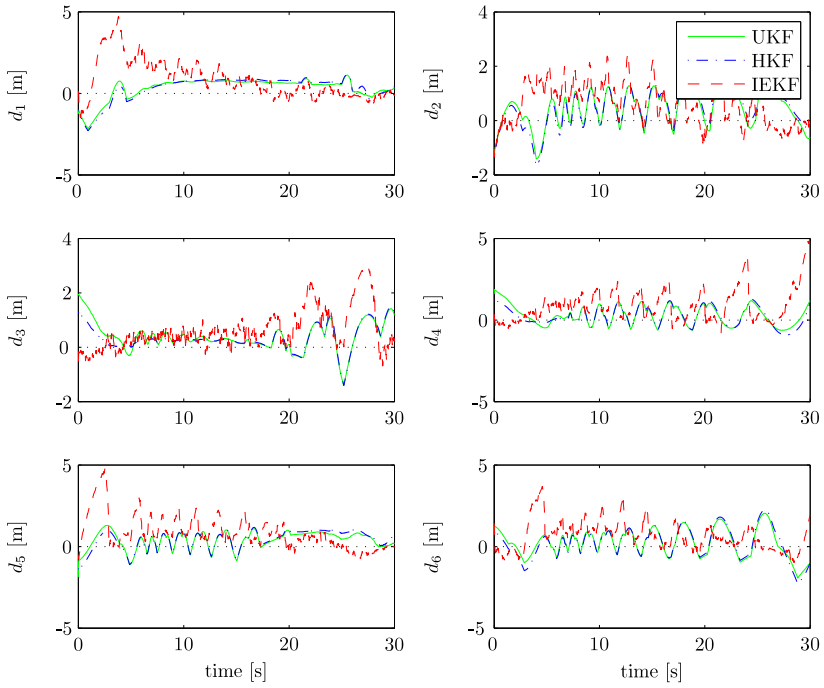


Figure A-19: d_1 – d_6 estimation errors

A-5 Sample Case D

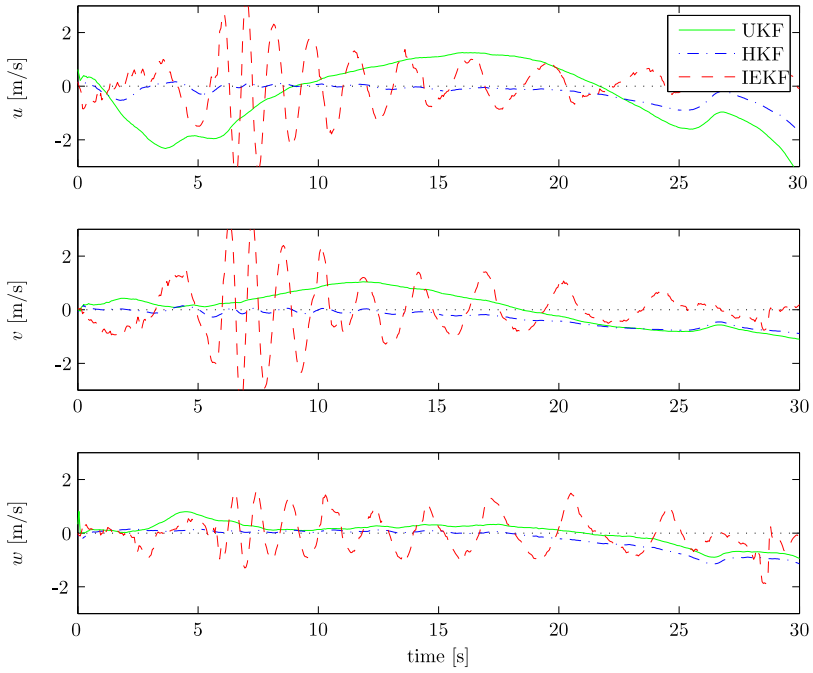


Figure A-20: $\vec{V}$ estimation errors

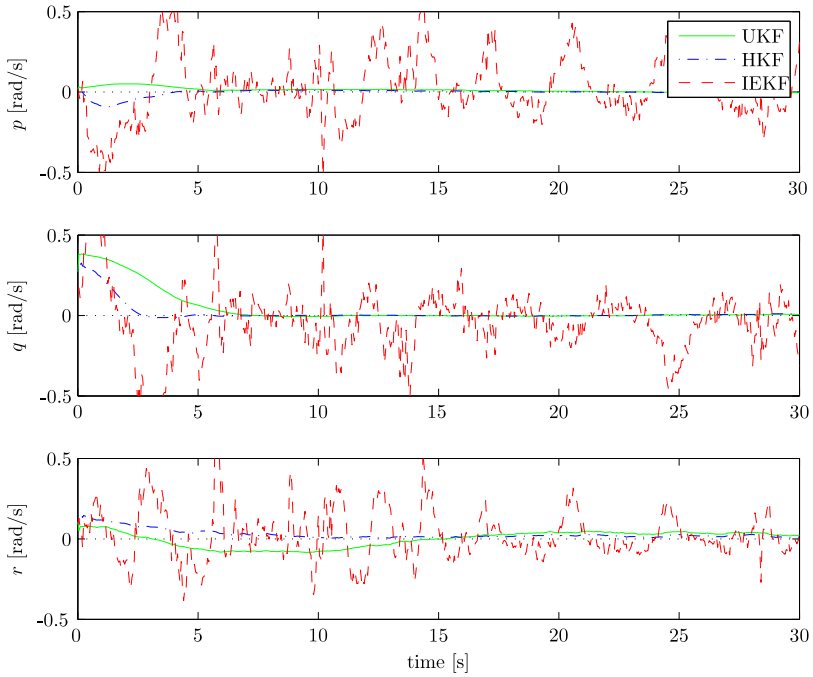


Figure A-21: $\vec{\omega}$ estimation errors

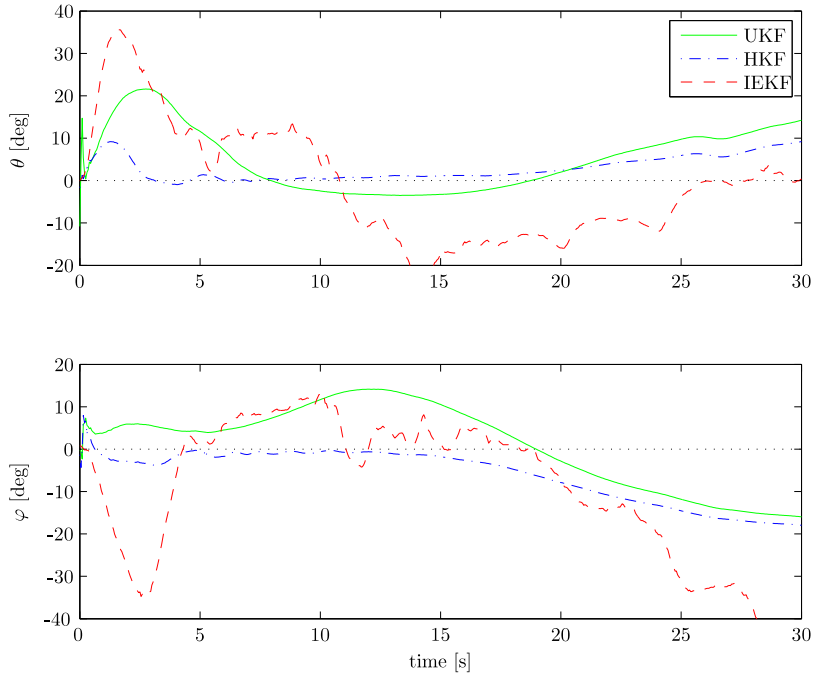


Figure A-22: θ & φ estimation errors

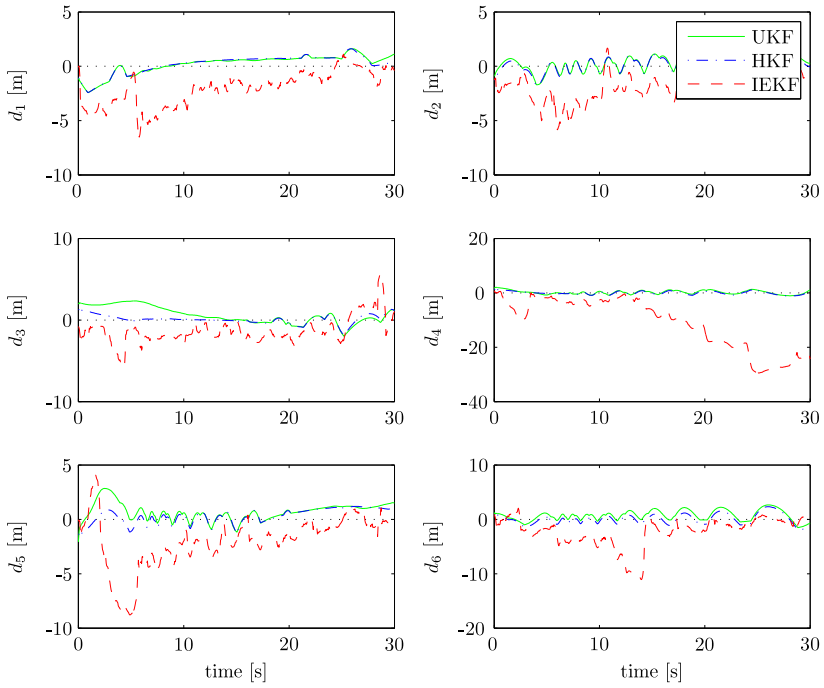


Figure A-23: d_1 – d_6 estimation errors

Appendix B

Pendulum Simulation Graphs

This appendix section contains the results from one simulation run using the pendulum model. First, the simulation output is shown, including the accelerometer output $\vec{A}$, the optical flow signals Ω and the vehicle states $\vec{V}$, $\vec{\omega}$, θ, φ and $d_1 - d_6$. As the pendulum is an autonomous system, no moments are available as input to the filters.
This is followed by the four sample cases as described in Section 6-1. In order to compare filter performance, the resulting state estimates from the three Kalman filter algorithms have been plotted together as errors with respect to the true states.
The discussion of these results is included with Section 6-2 on page 64.

B-1 Input, Measurements and the State

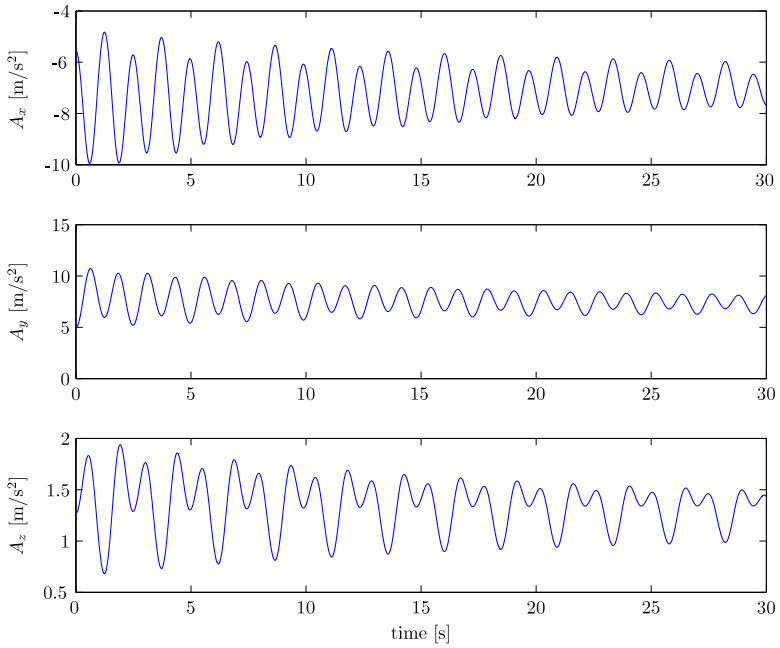


Figure B-1: $\vec{A}$

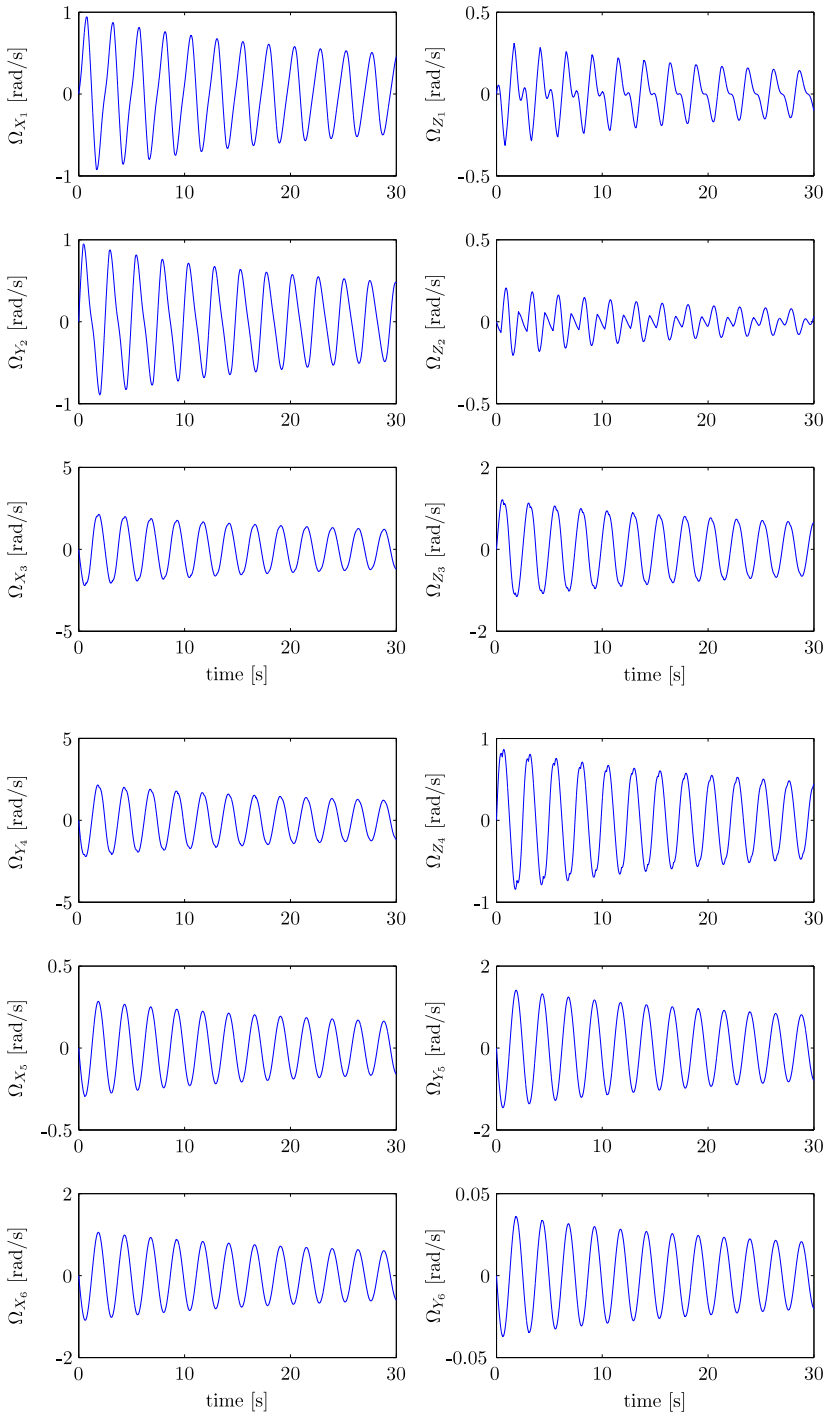


Figure B-2: $\vec{\Omega}$

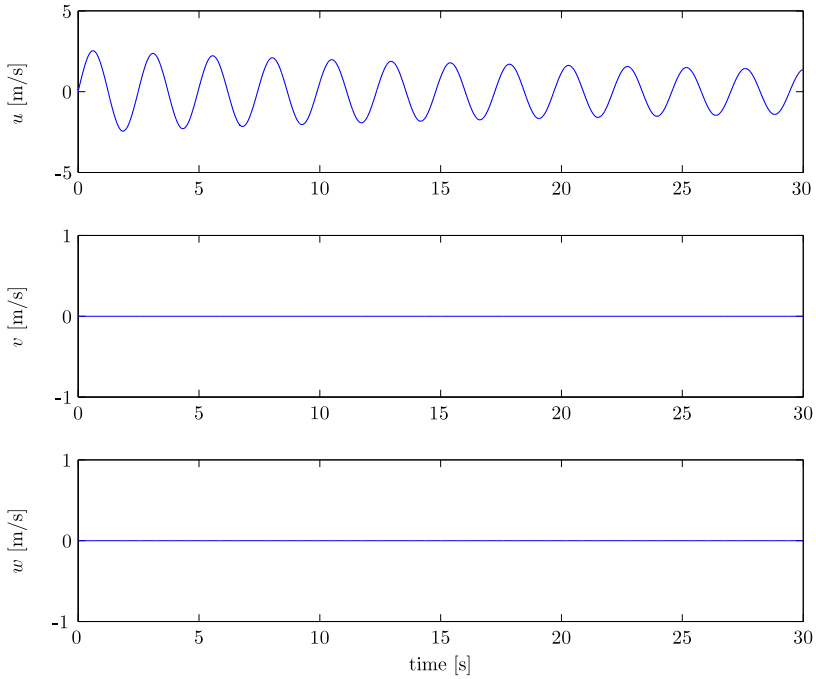


Figure B-3: $\vec{V}$

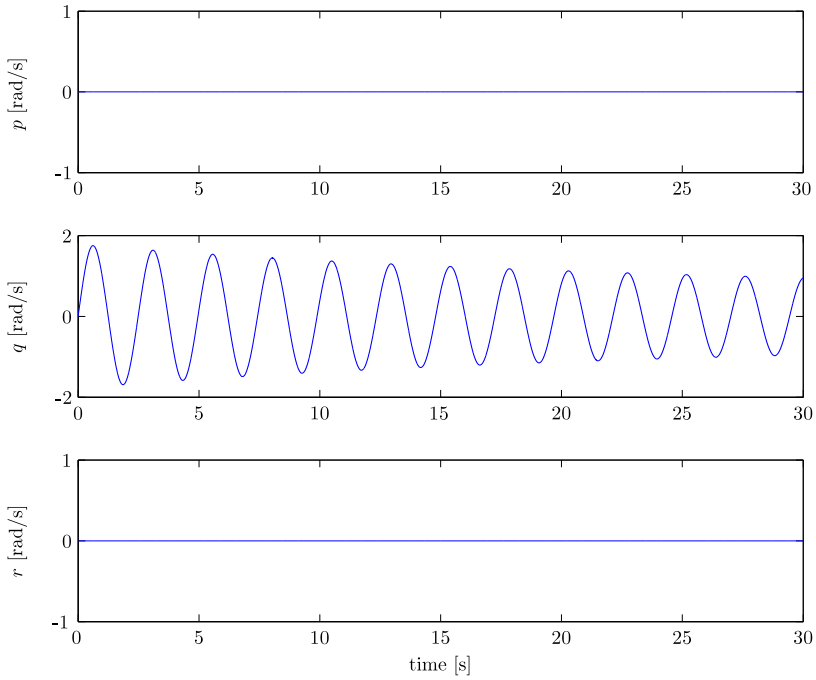


Figure B-4: $\vec{\omega}$

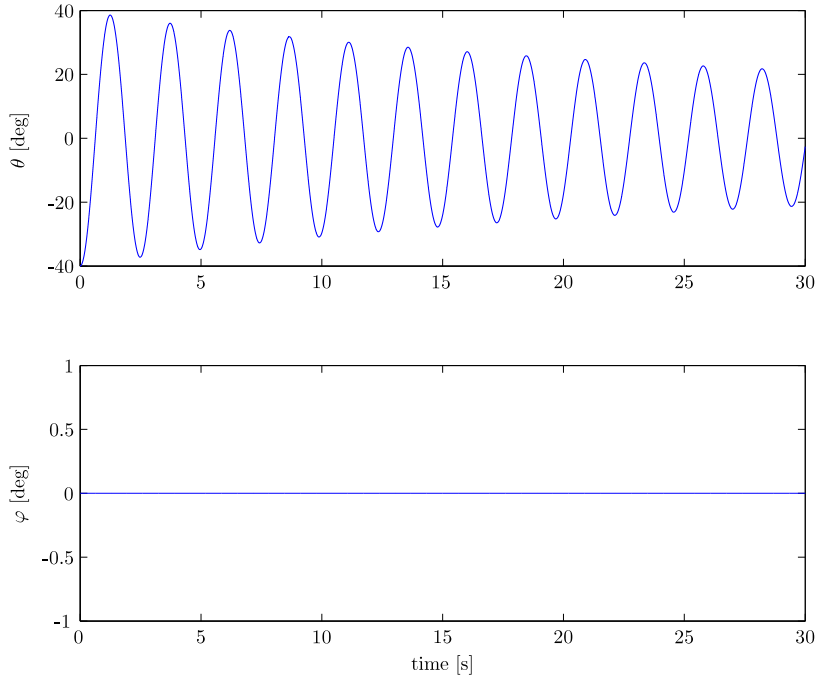


Figure B-5: θ & φ

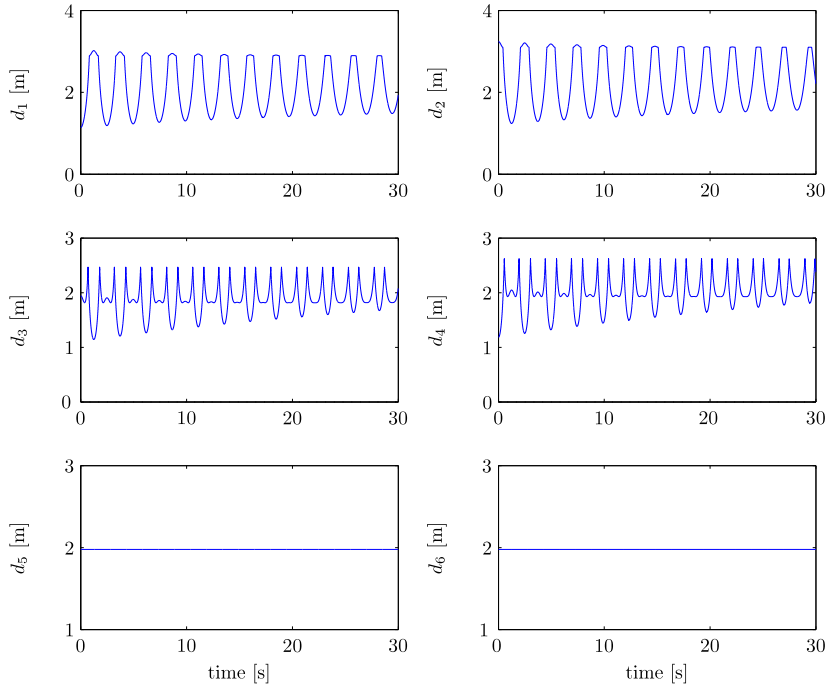


Figure B-6: d_1 – d_6

B-2 Sample Case A

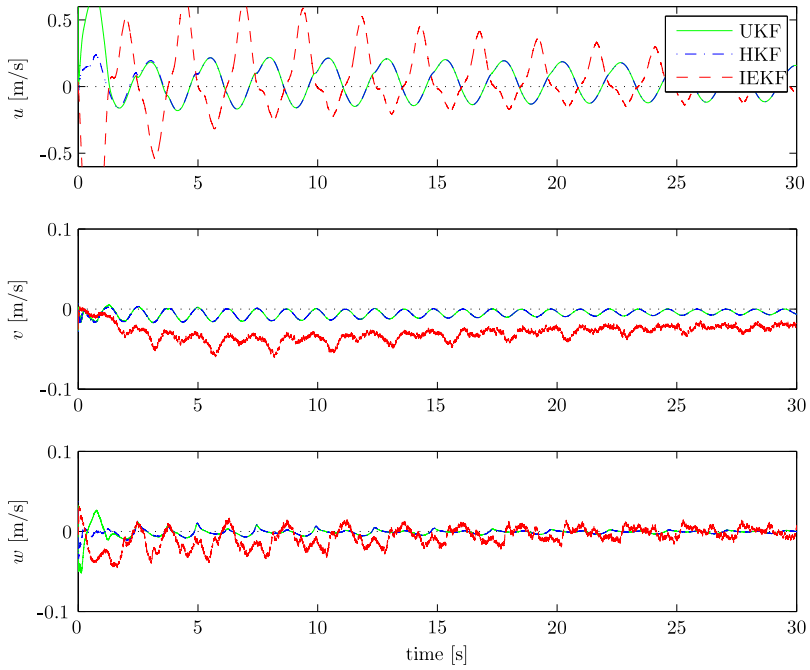


Figure B-7: $\vec{V}$ estimation errors

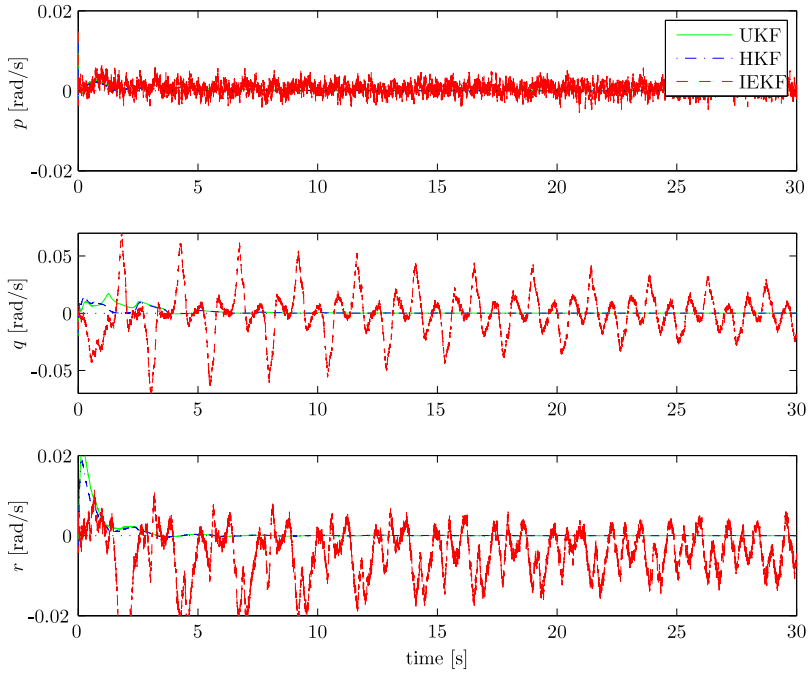


Figure B-8: $\vec{\omega}$ estimation errors

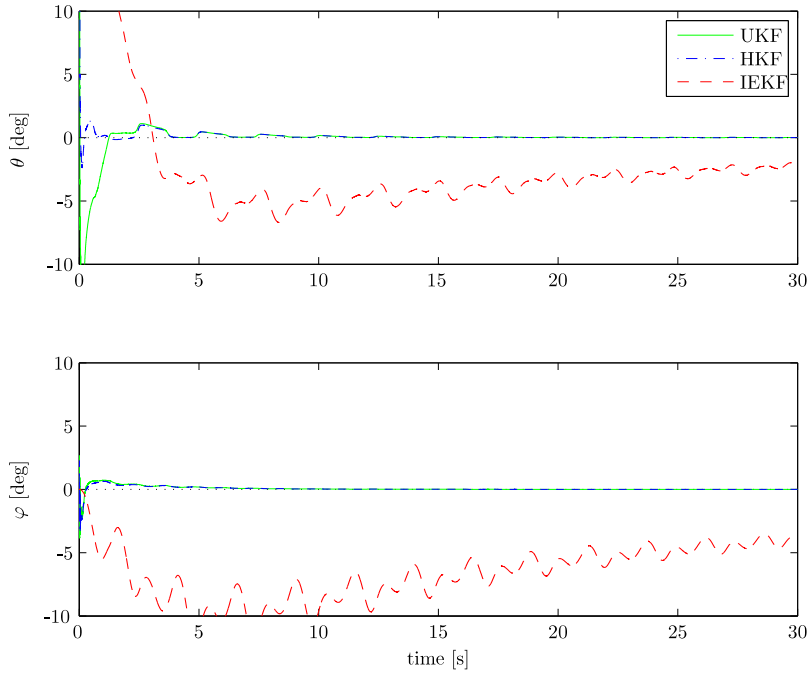


Figure B-9: θ & φ estimation errors

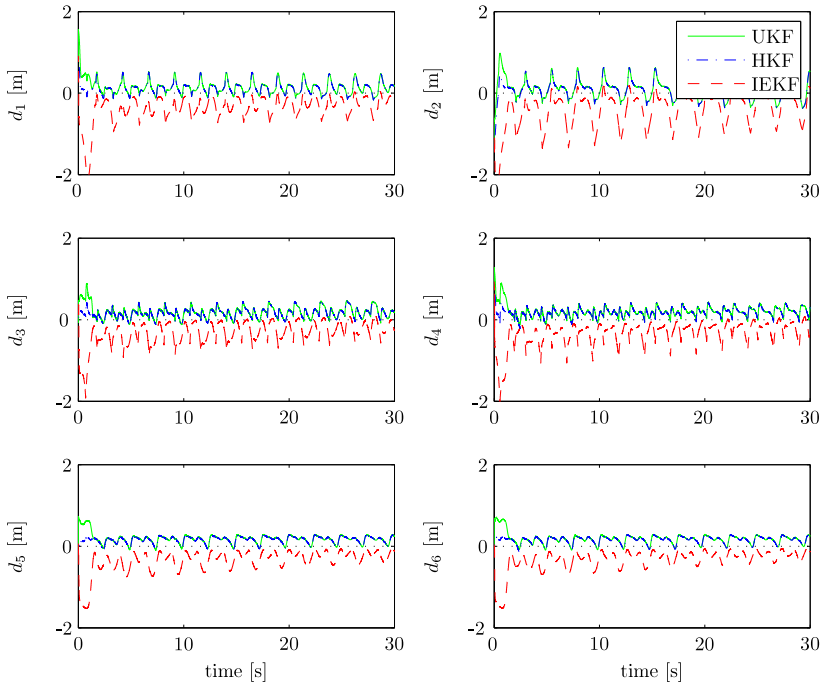


Figure B-10: d_1 – d_6 estimation errors

B-3 Sample Case B

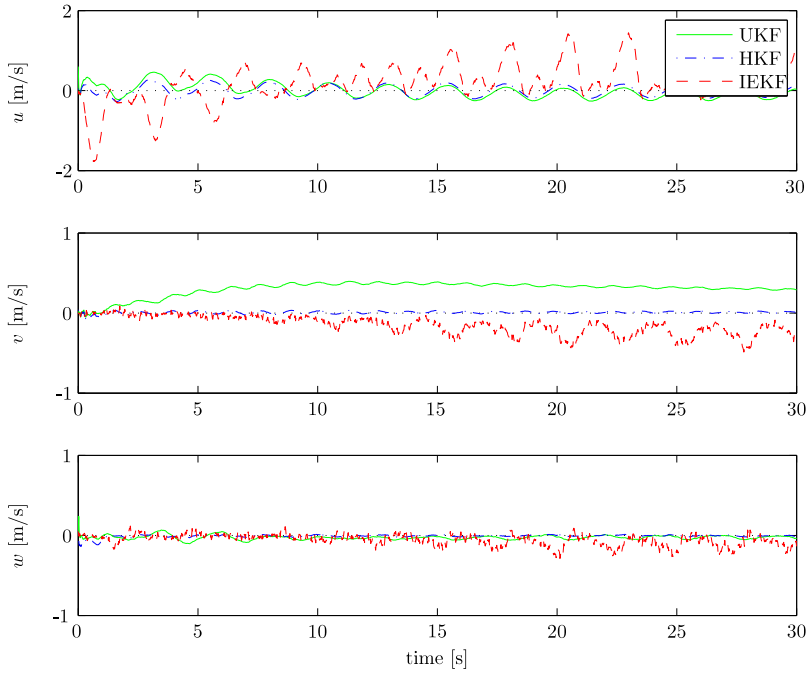


Figure B-11: $\vec{V}$ estimation errors

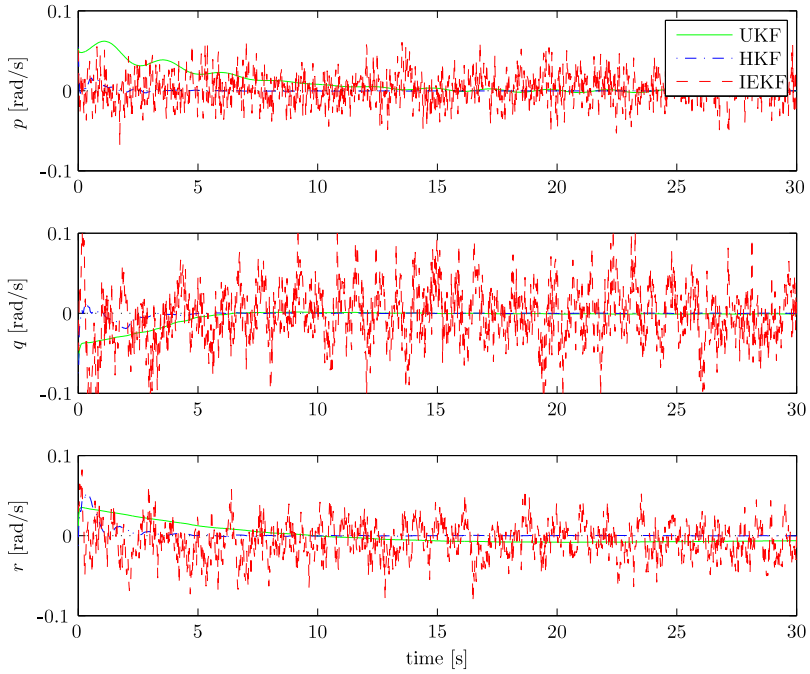


Figure B-12: $\vec{\omega}$ estimation errors

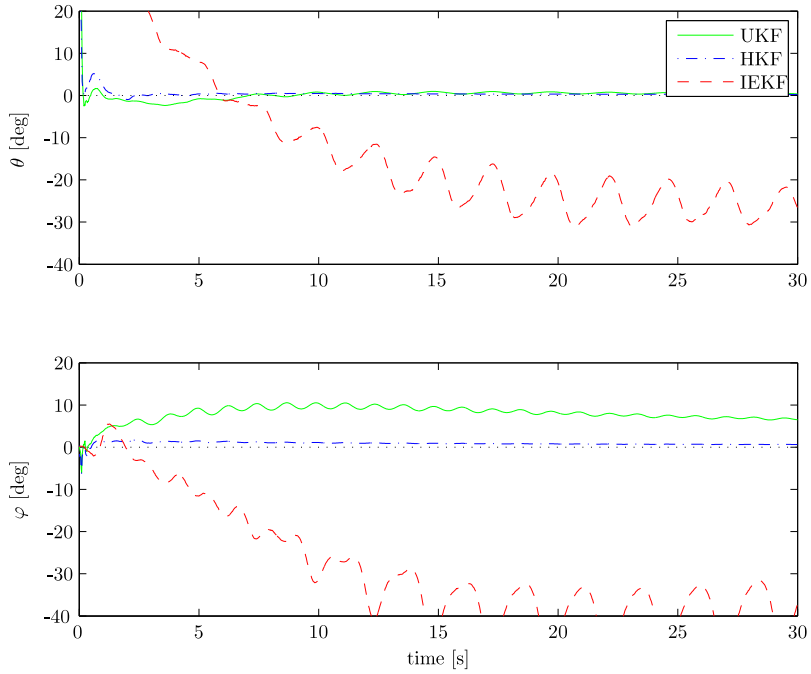


Figure B-13: θ & φ estimation errors

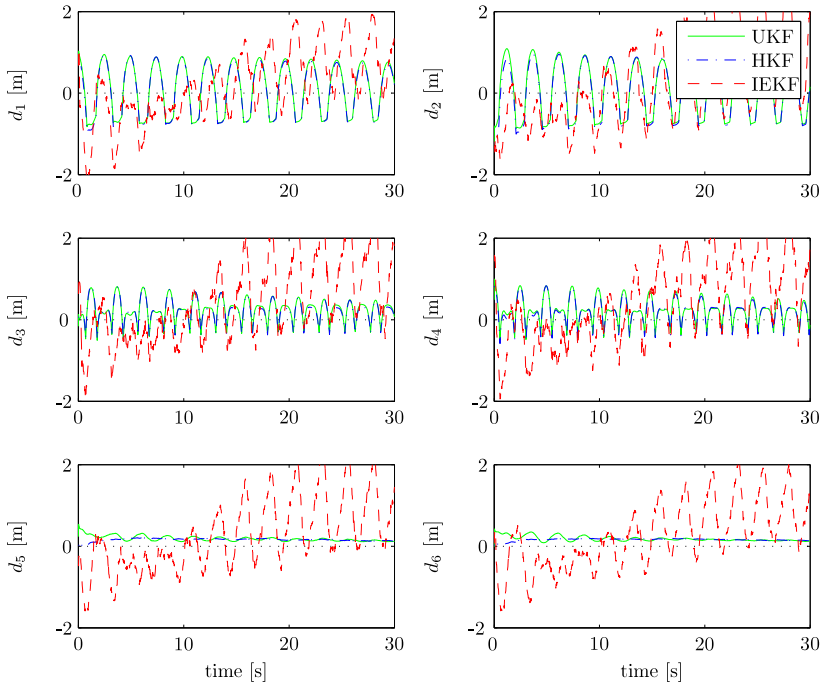


Figure B-14: d_1 – d_6 estimation errors

B-4 Sample Case C

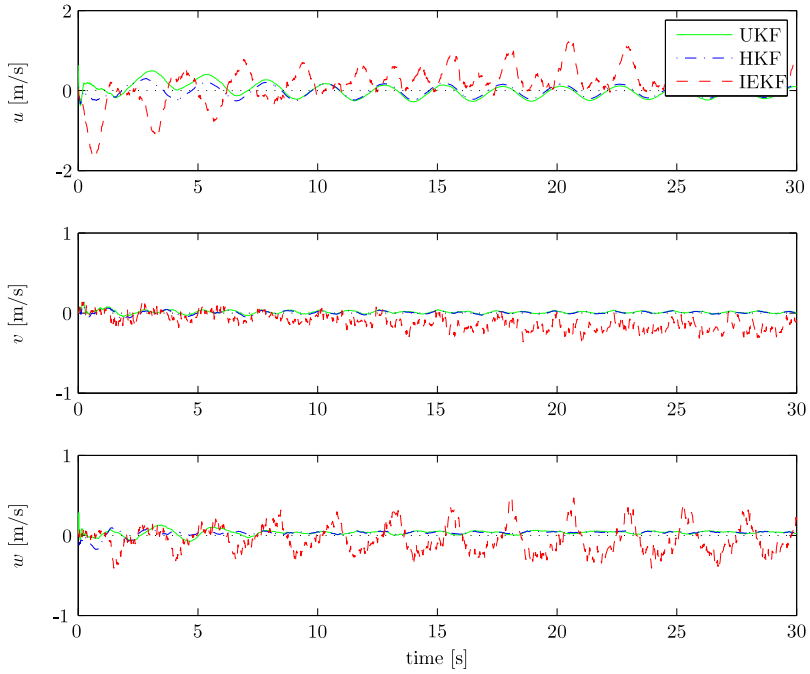


Figure B-15: $\vec{V}$ estimation errors

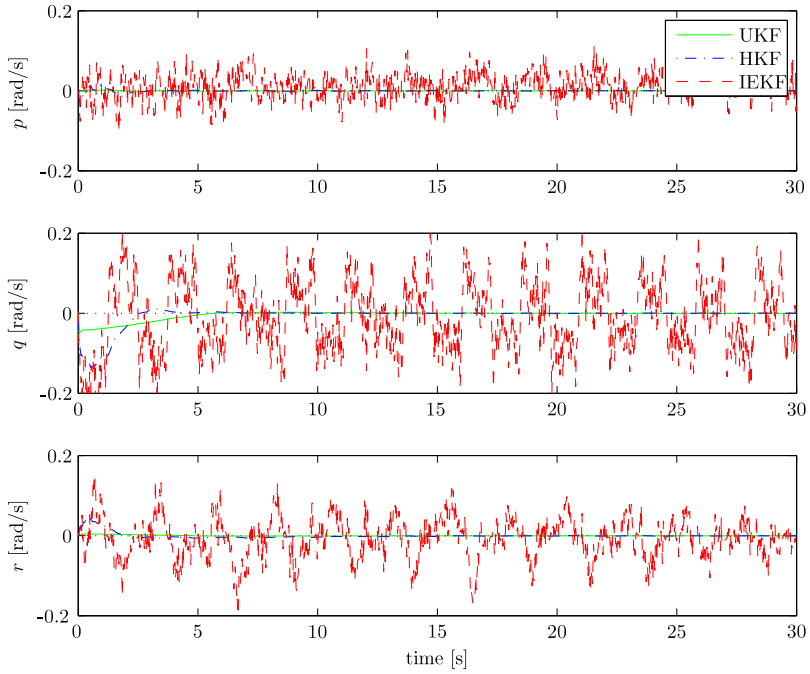


Figure B-16: $\vec{\omega}$ estimation errors

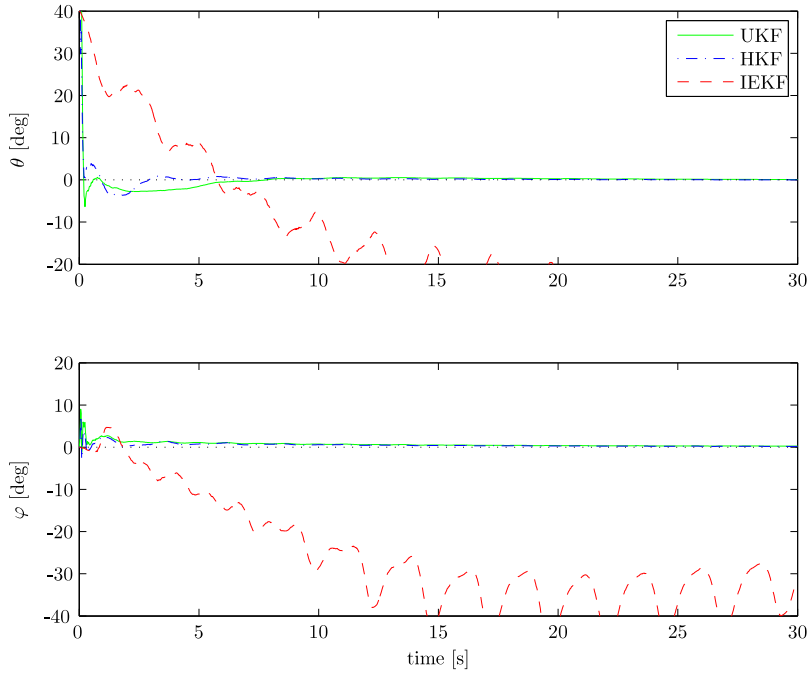


Figure B-17: θ & φ estimation errors

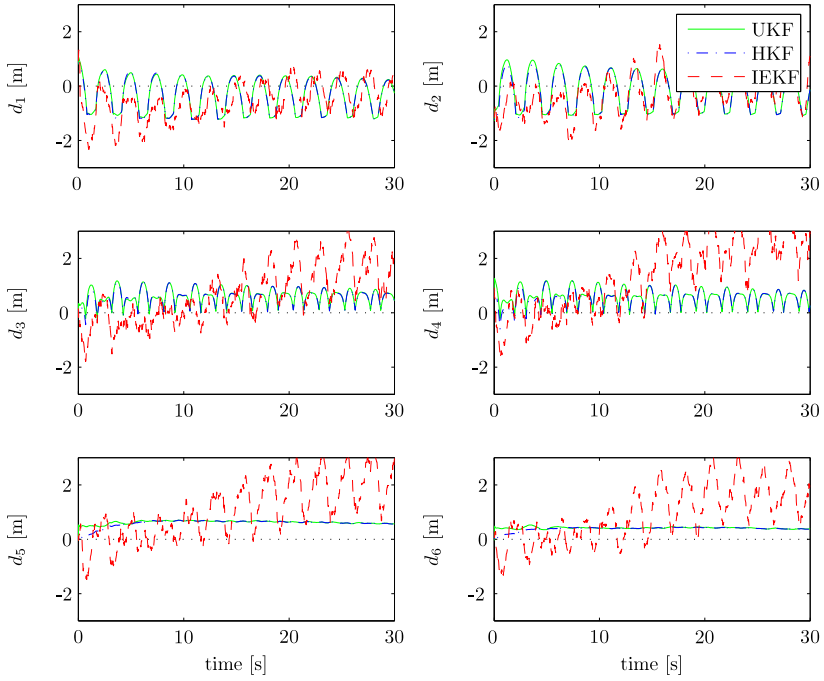


Figure B-18: d_1 – d_6 estimation errors

B-5 Sample Case D

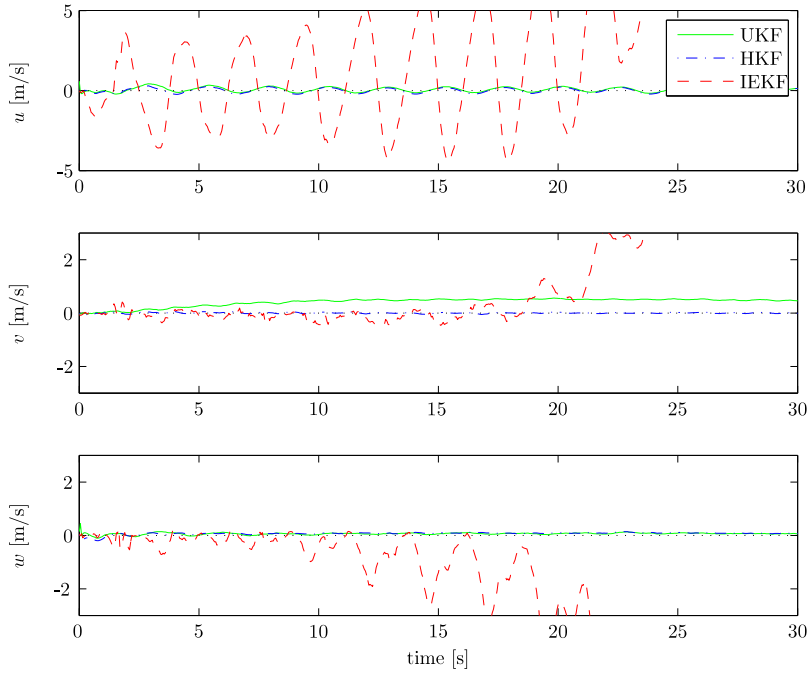


Figure B-19: $\vec{V}$ estimation errors

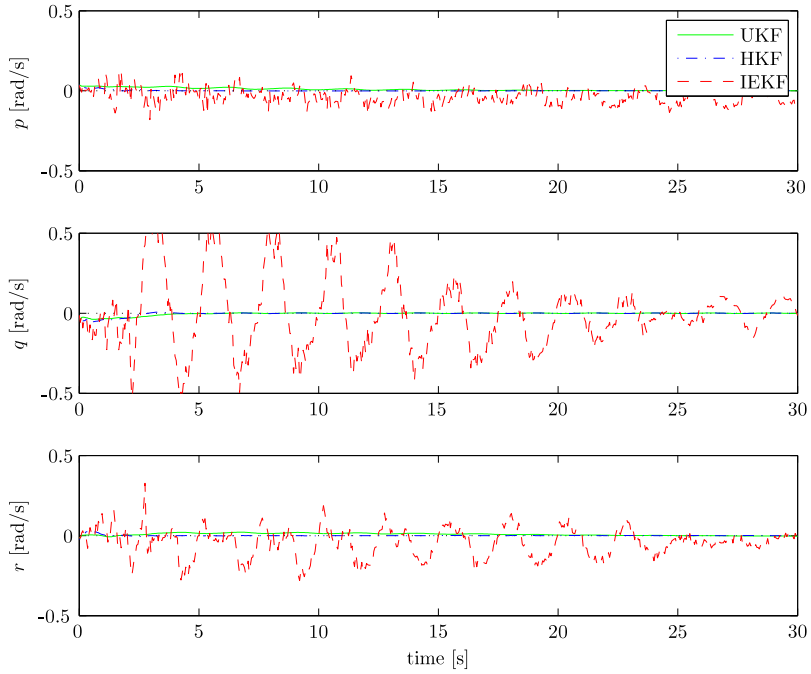


Figure B-20: $\vec{\omega}$ estimation errors

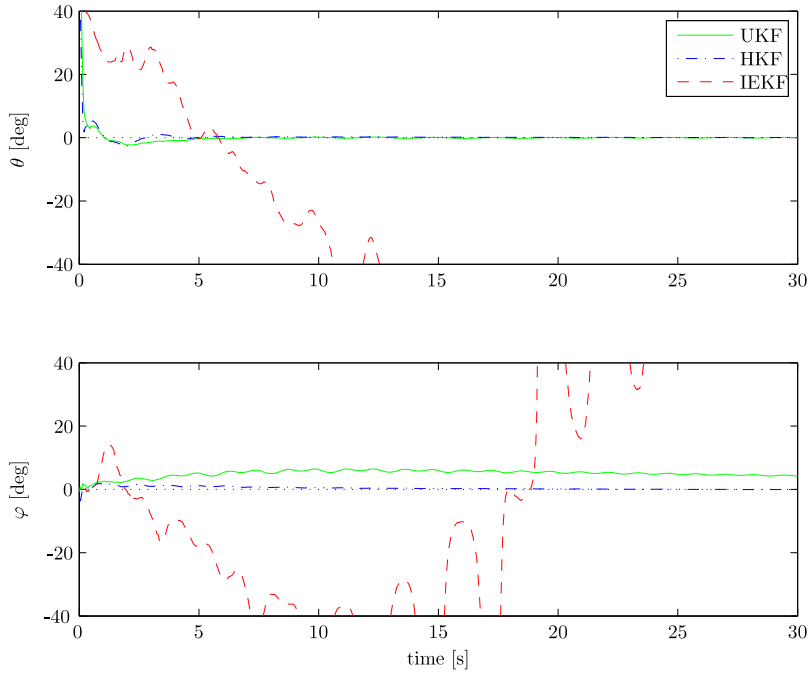


Figure B-21: θ & φ estimation errors

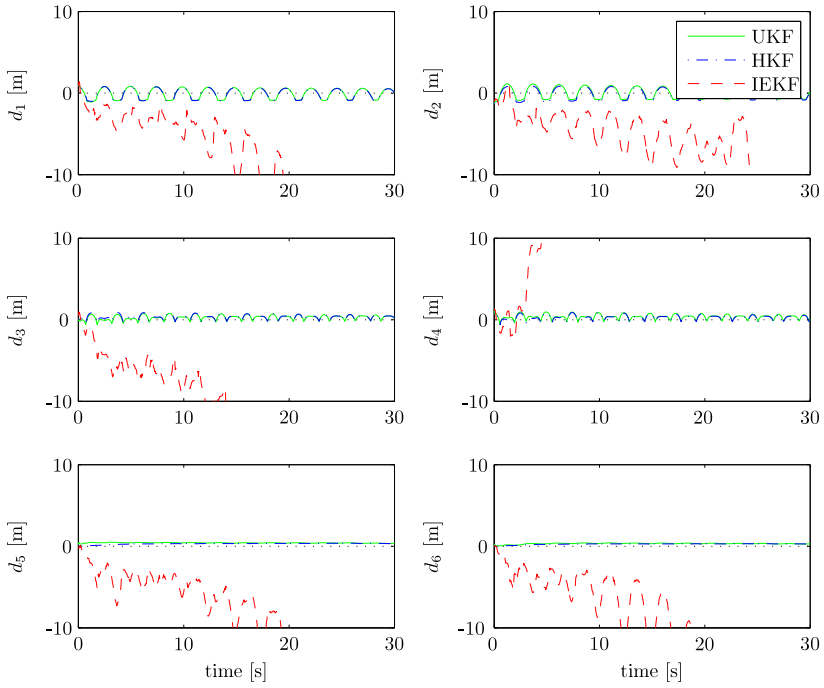


Figure B-22: d_1 – d_6 estimation errors

Appendix C

Hardware Experiment Graphs

These graphs show the state estimates from the three filter types for a run at 50 Hz and a run at 25 Hz. Unlike with the simulation results, no "truth" data is available for the experiment. Therefore, the plots do not show the error, but simply the signals themselves. However, the motion has certain known properties, such as period and amplitude of a damped sinusoid. For reference, the simulated states for the pendulum are plotted in Figures B-3 - B-6. The estimated standard deviations have been plotted as well. This gives an indication of the convergence behaviour of the Kalman filter. The corresponding discussion is included with Section 9-2 on page 86.

C-1 State Estimates, Condition Numbers and Standard Deviations from a Data Recording at 50 Hz

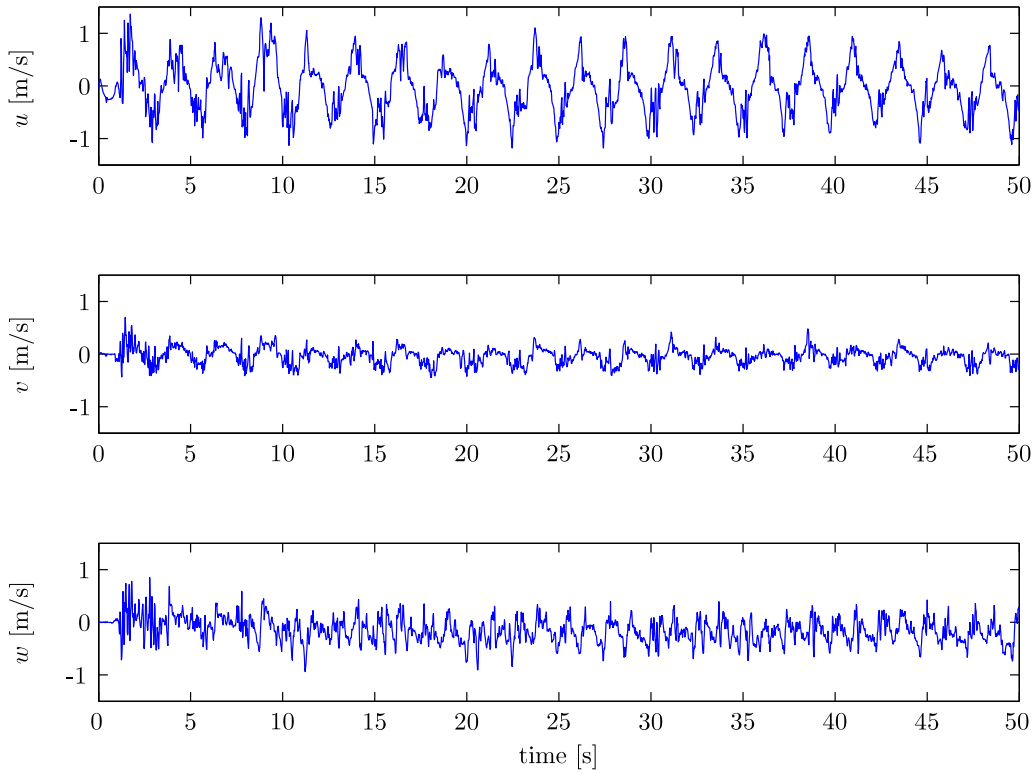


Figure C-1: IEKF velocity estimates @ 50 Hz

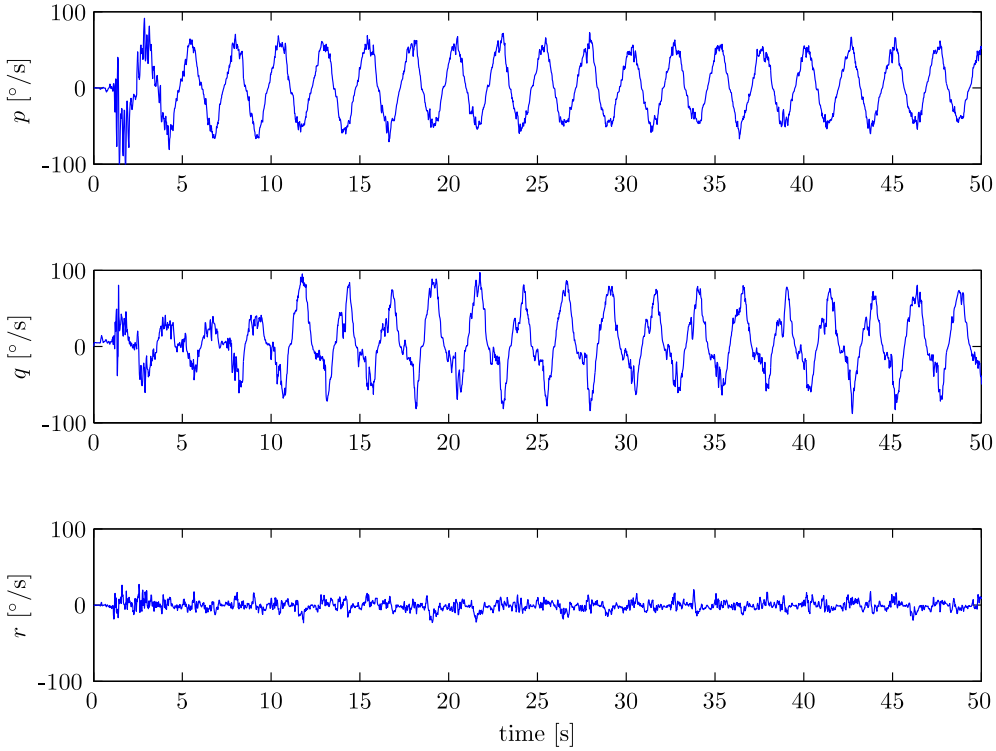


Figure C-2: IEKF rotation estimates @ 50 Hz

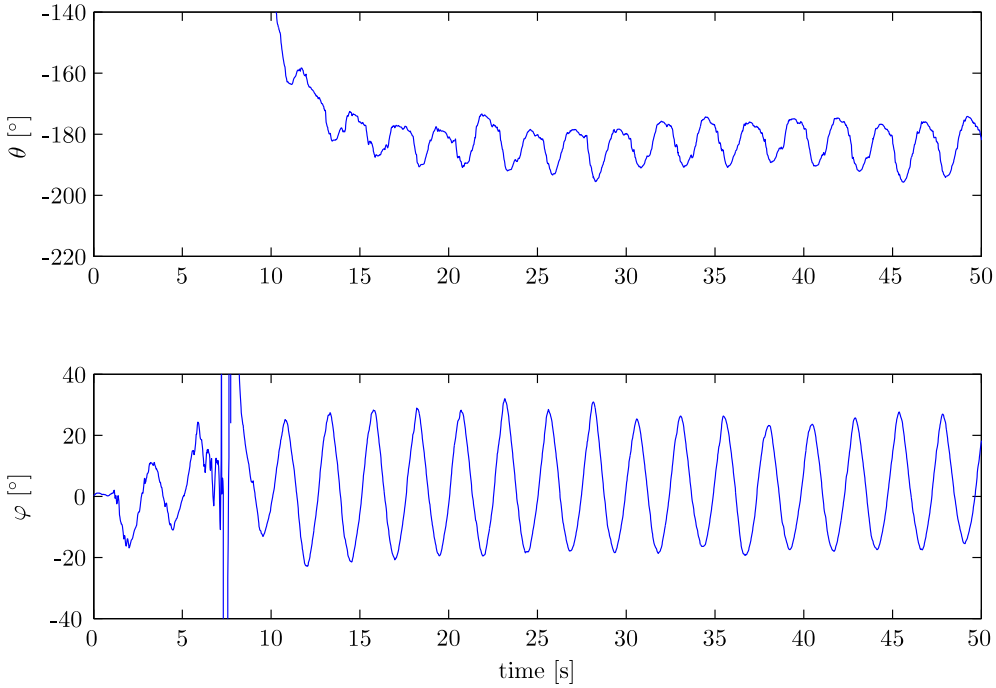


Figure C-3: IEKF attitude estimates @ 50 Hz

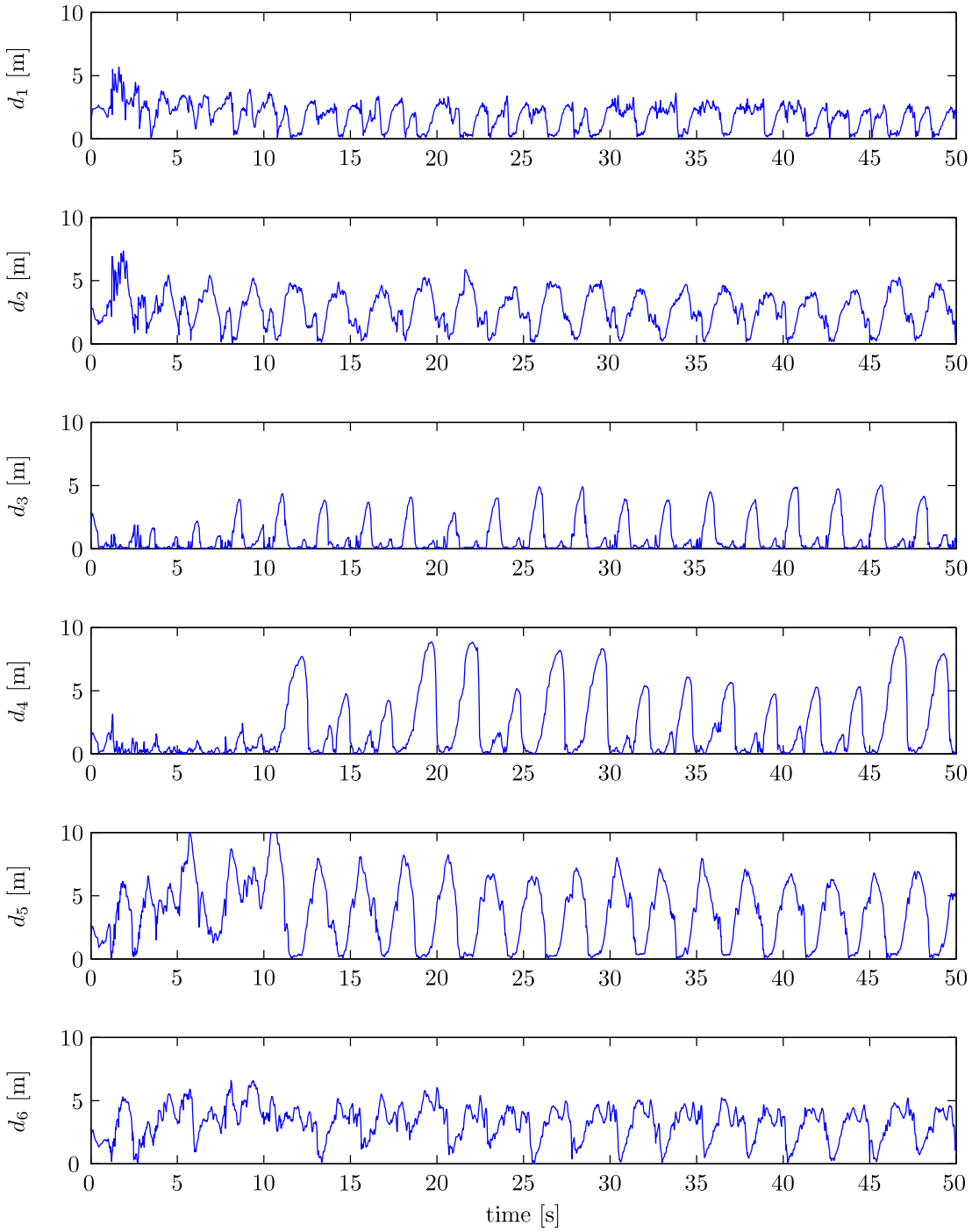


Figure C-4: IEKF distance estimates @ 50 Hz

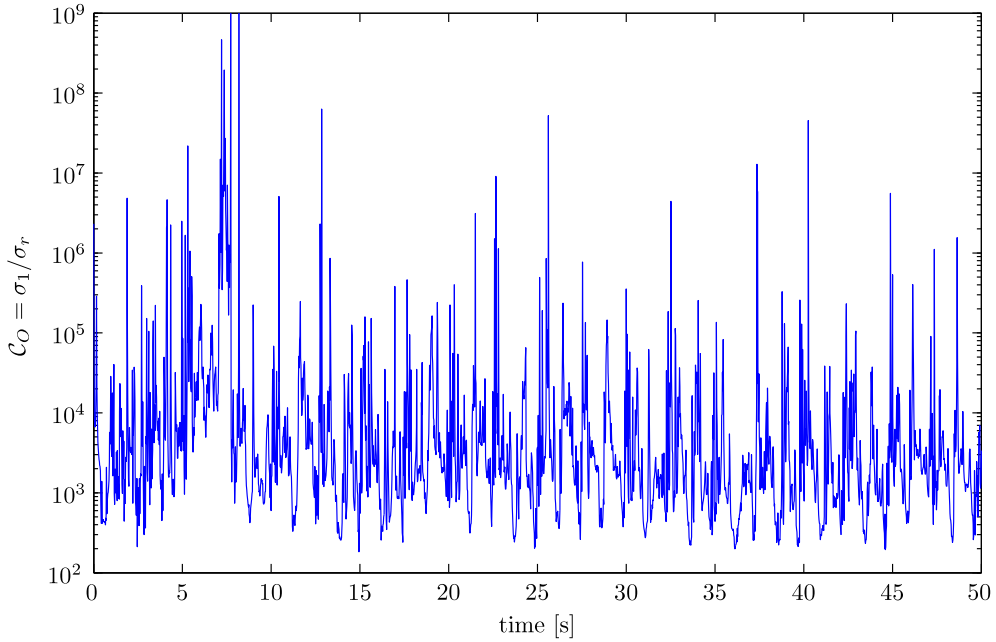


Figure C-5: IEKF condition number @ 50 Hz

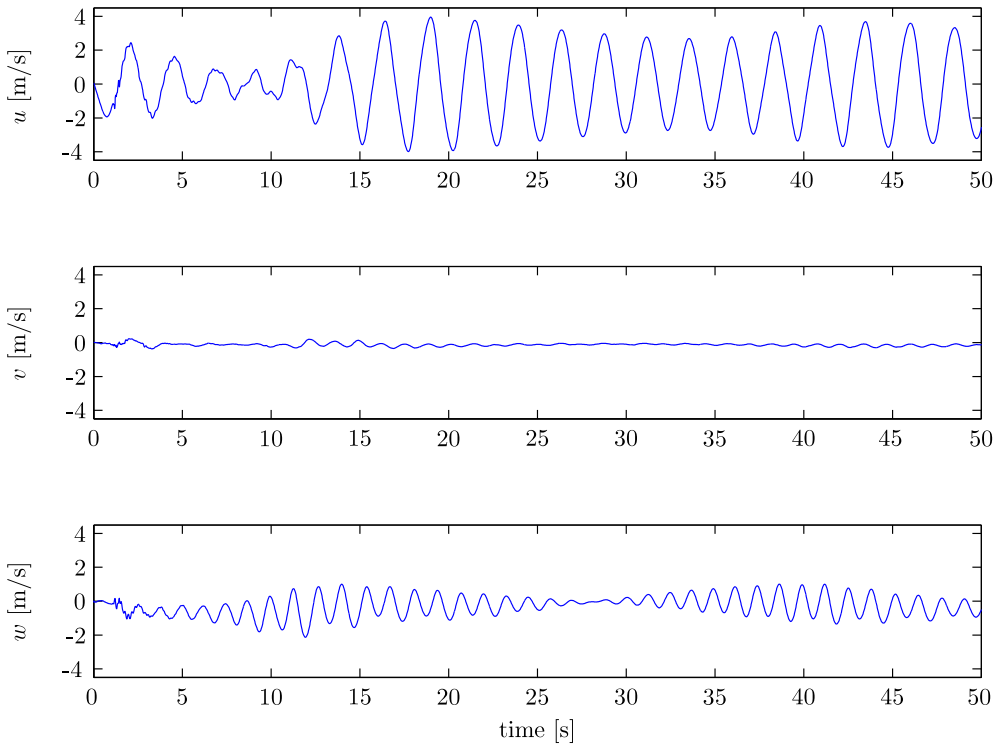


Figure C-6: UKF velocity estimates @ 50 Hz

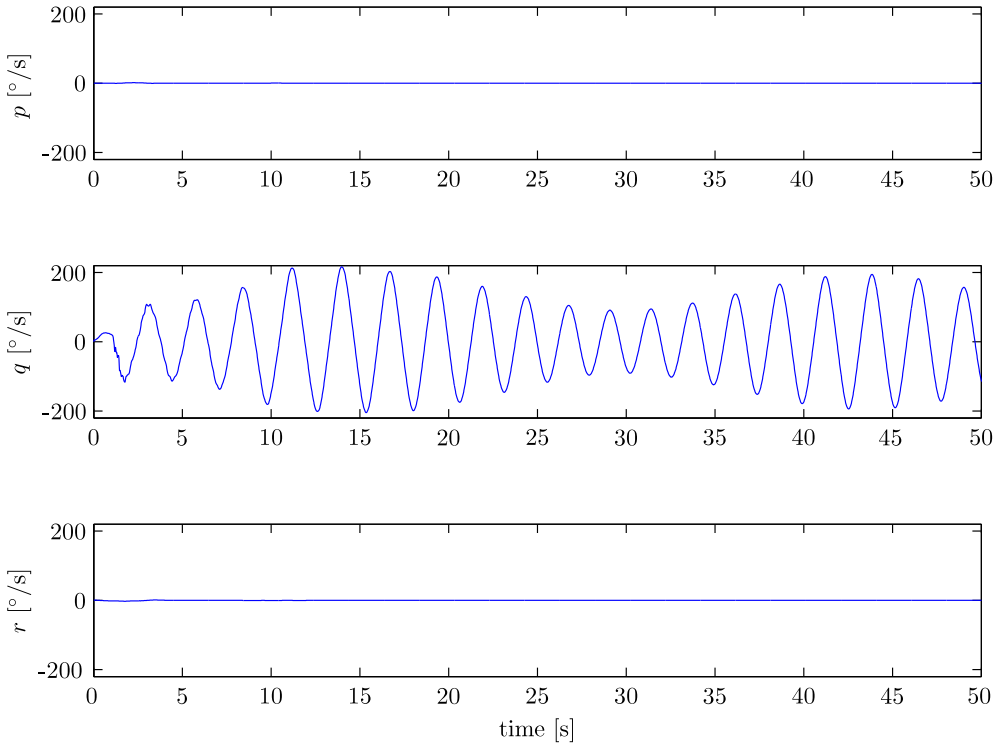


Figure C-7: UKF rotation estimates @ 50 Hz

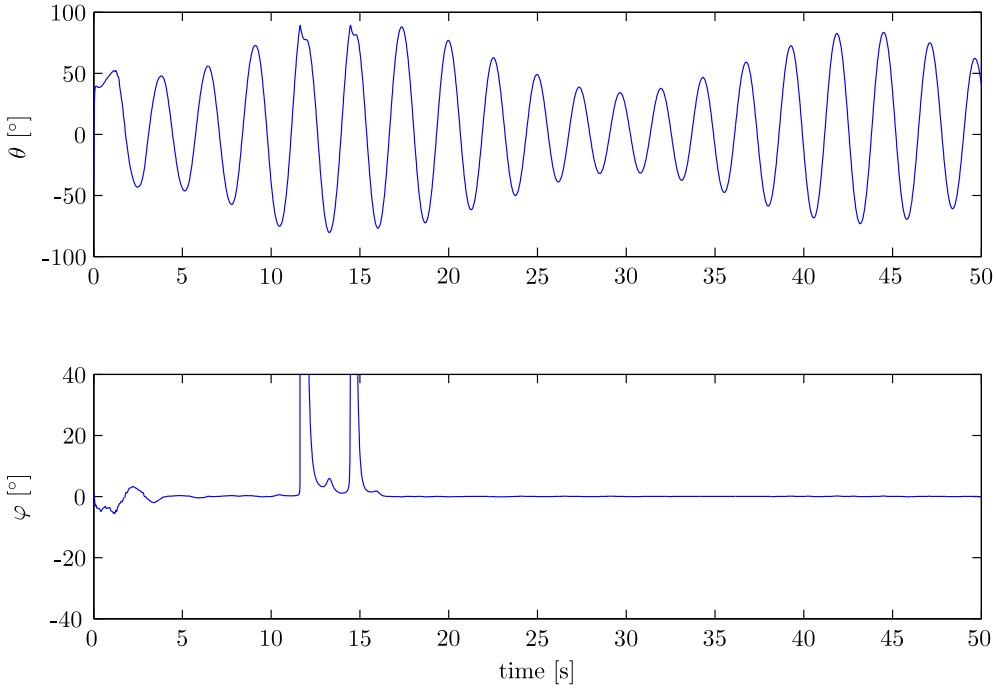


Figure C-8: UKF attitude estimates @ 50 Hz

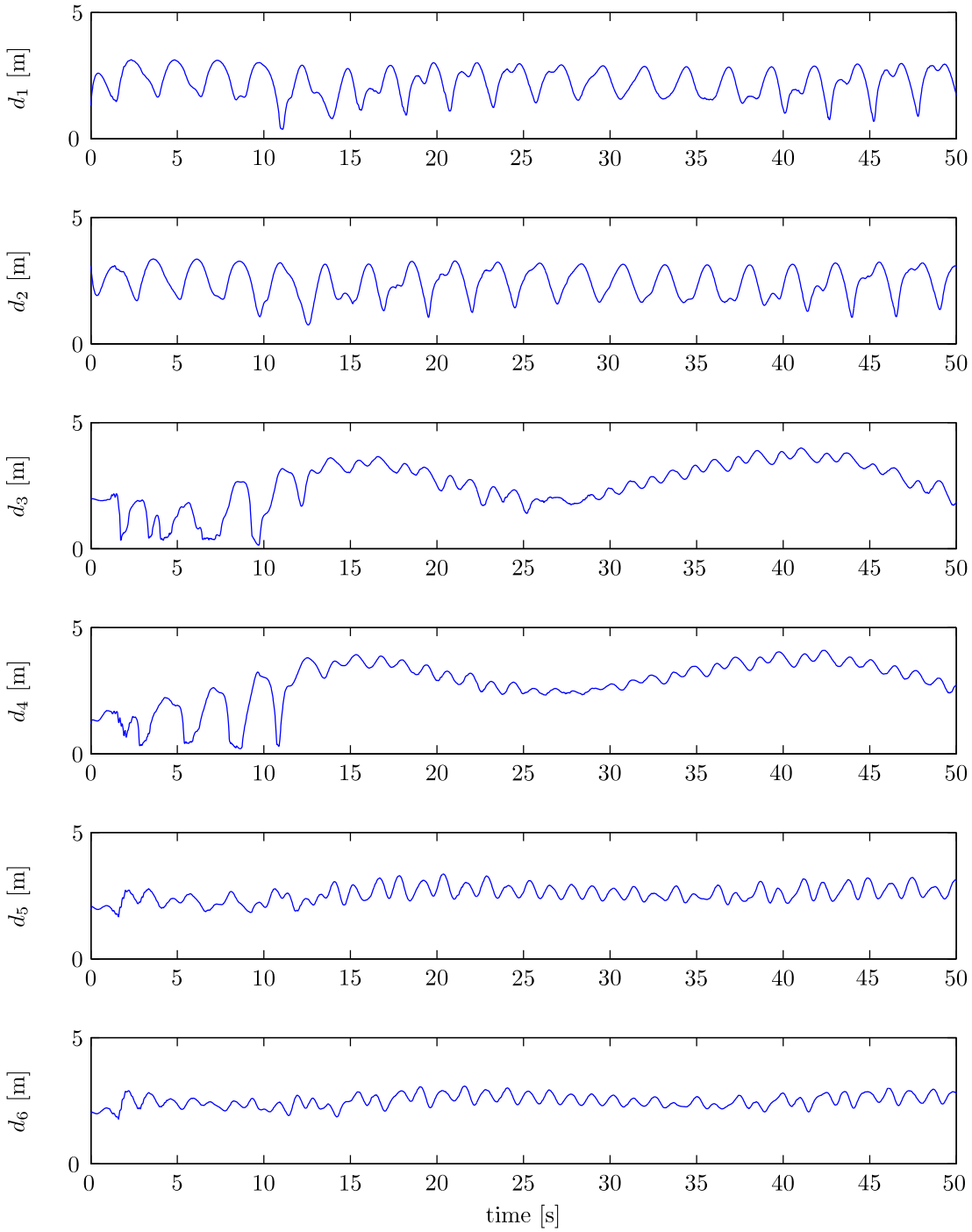


Figure C-9: UKF distance estimates @ 50 Hz

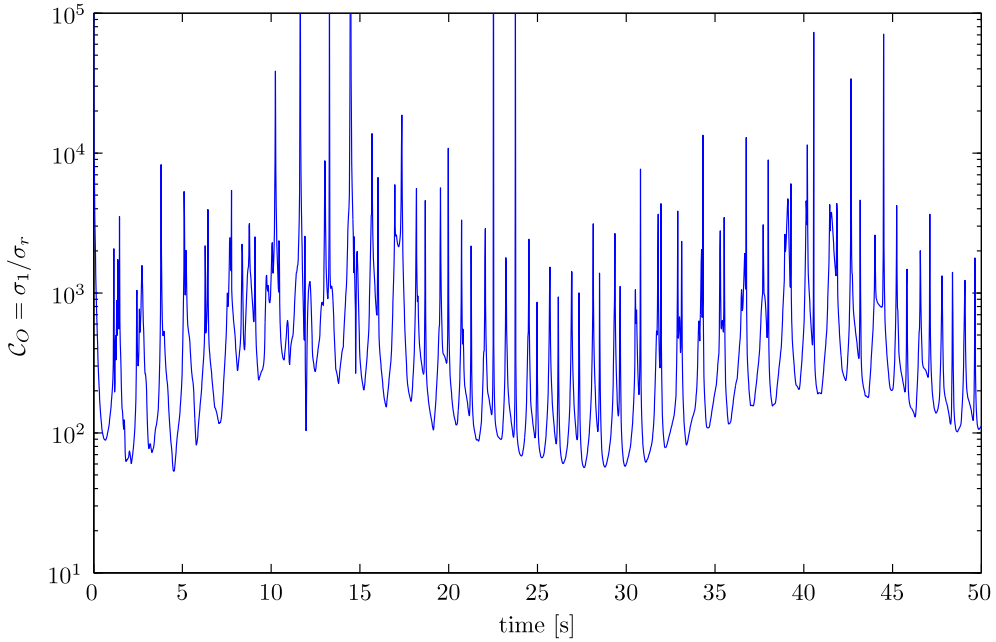


Figure C-10: UKF condition number @ 50 Hz

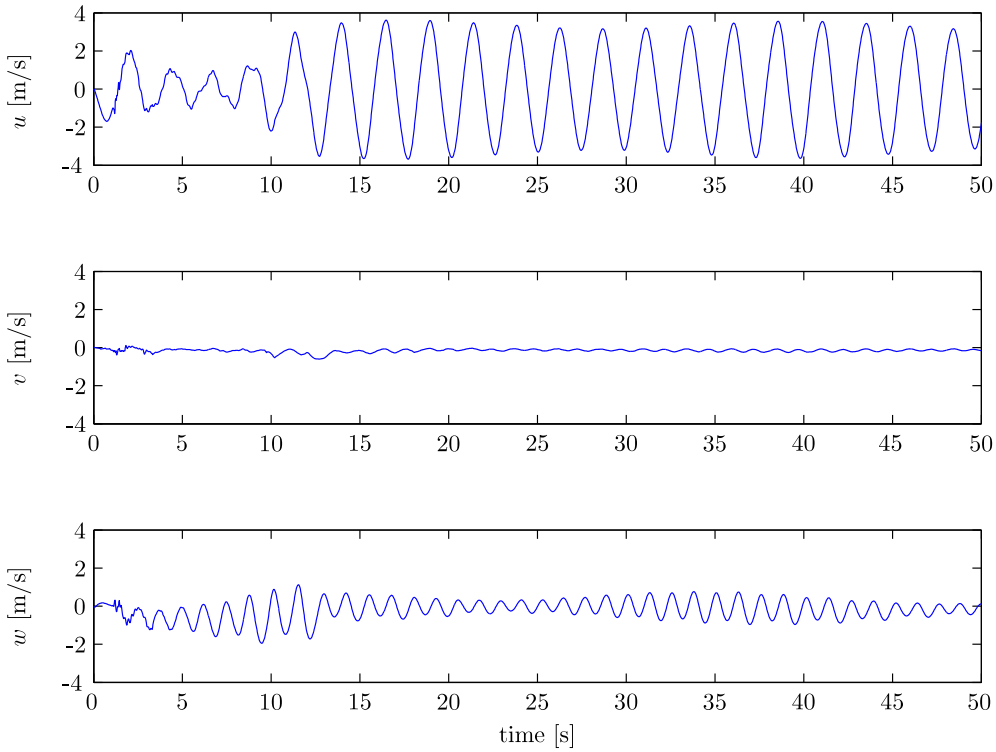


Figure C-11: HKF velocity estimates @ 50 Hz

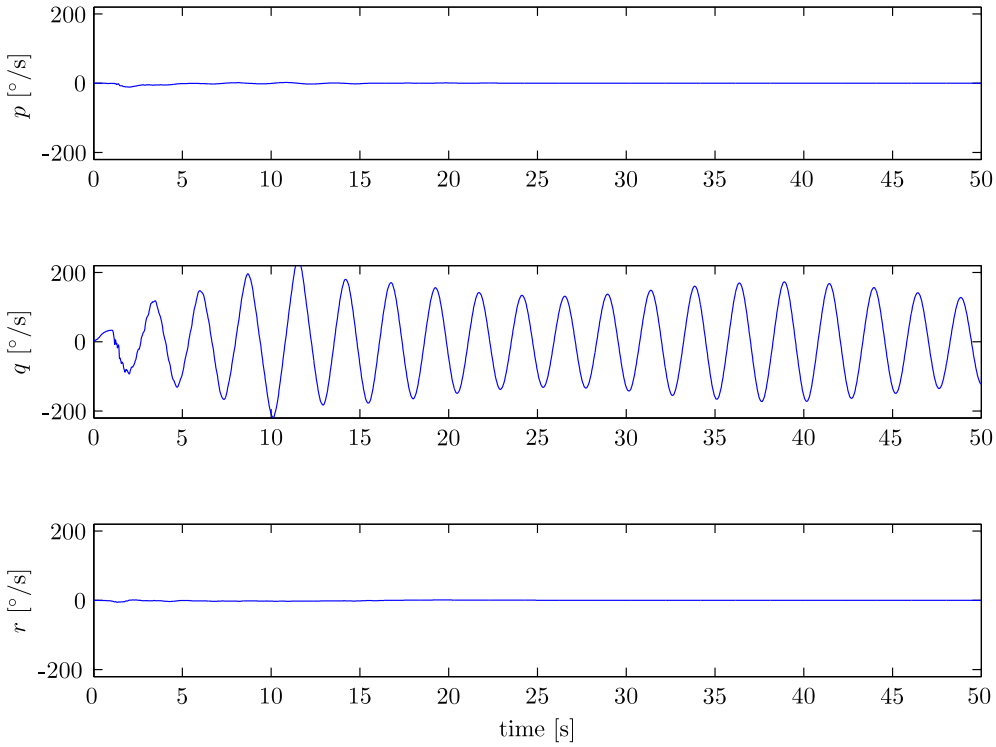


Figure C-12: HKF rotation estimates @ 50 Hz

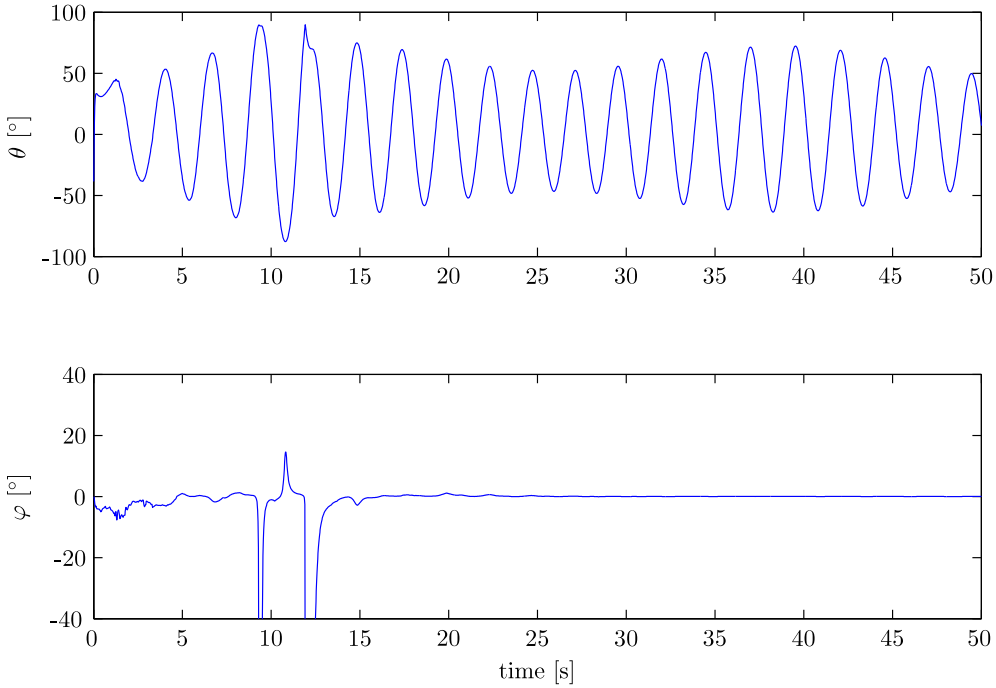


Figure C-13: HKF attitude estimates @ 50 Hz

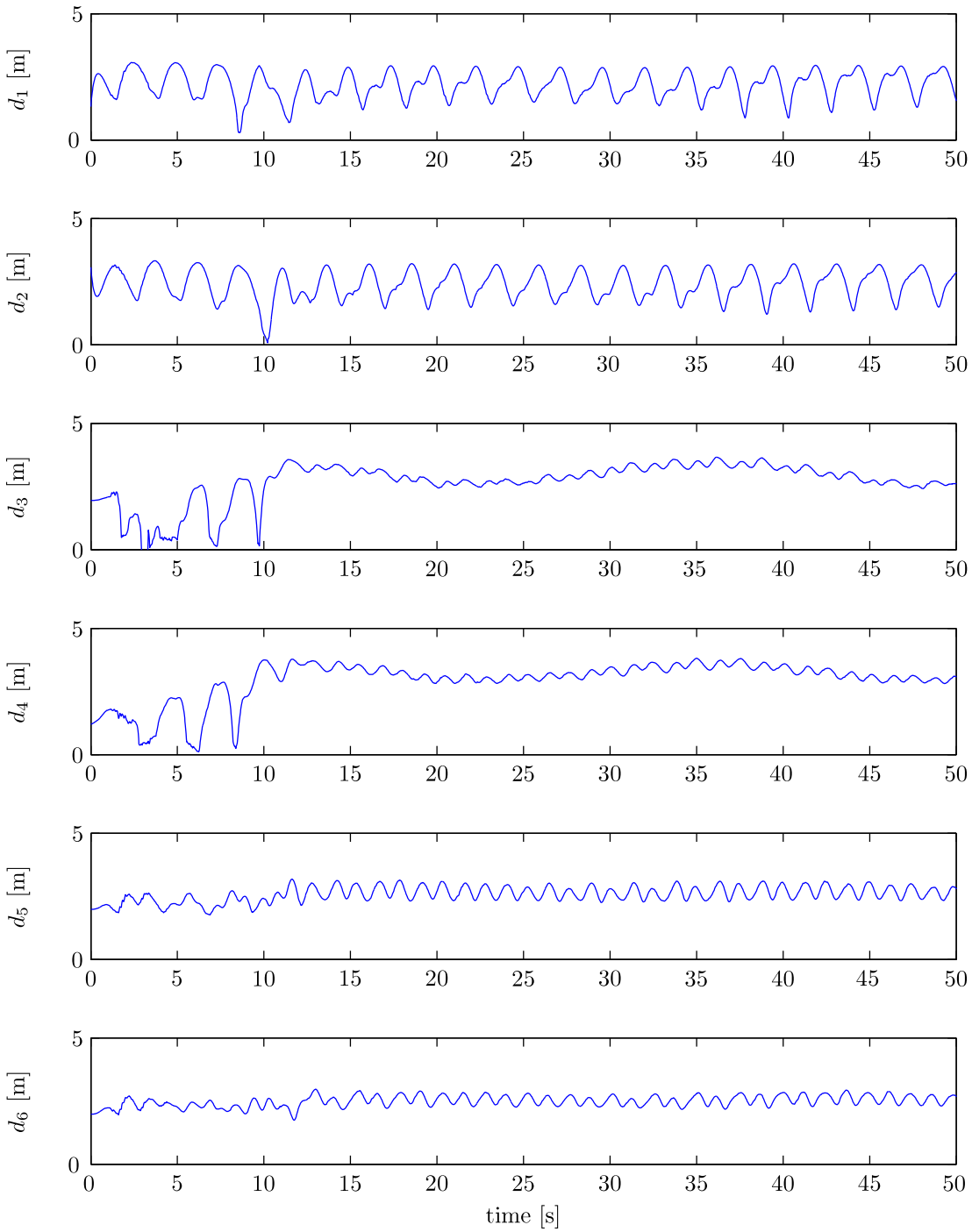


Figure C-14: HKF distance estimates @ 50 Hz

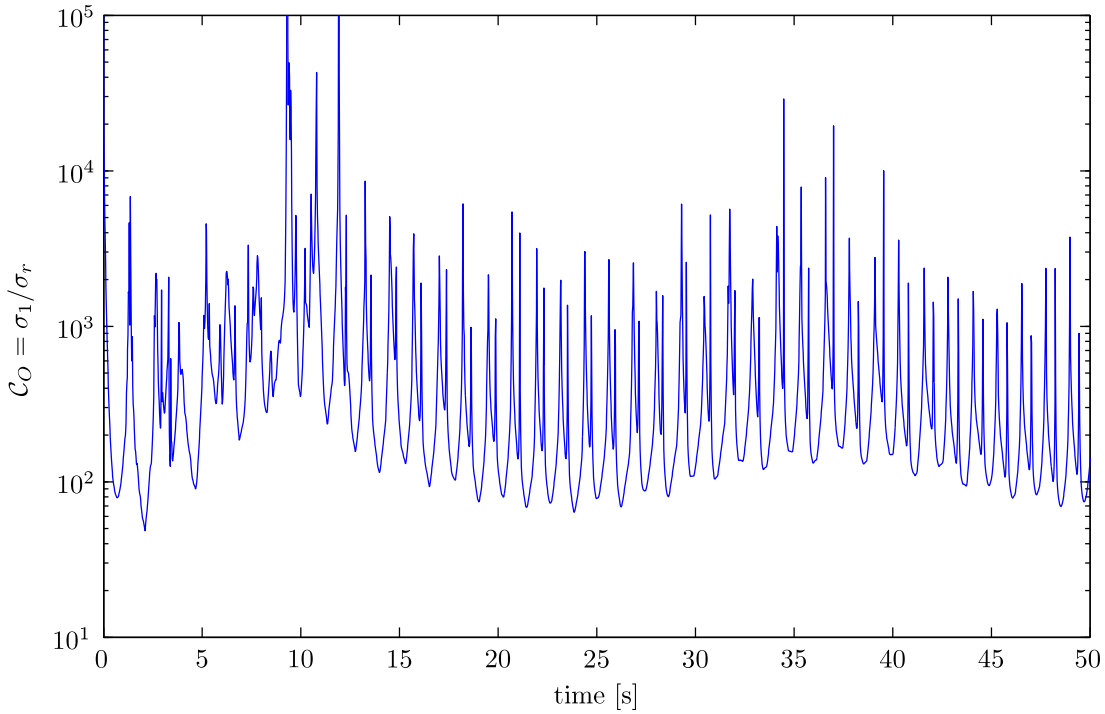


Figure C-15: HKF condition number @ 50 Hz

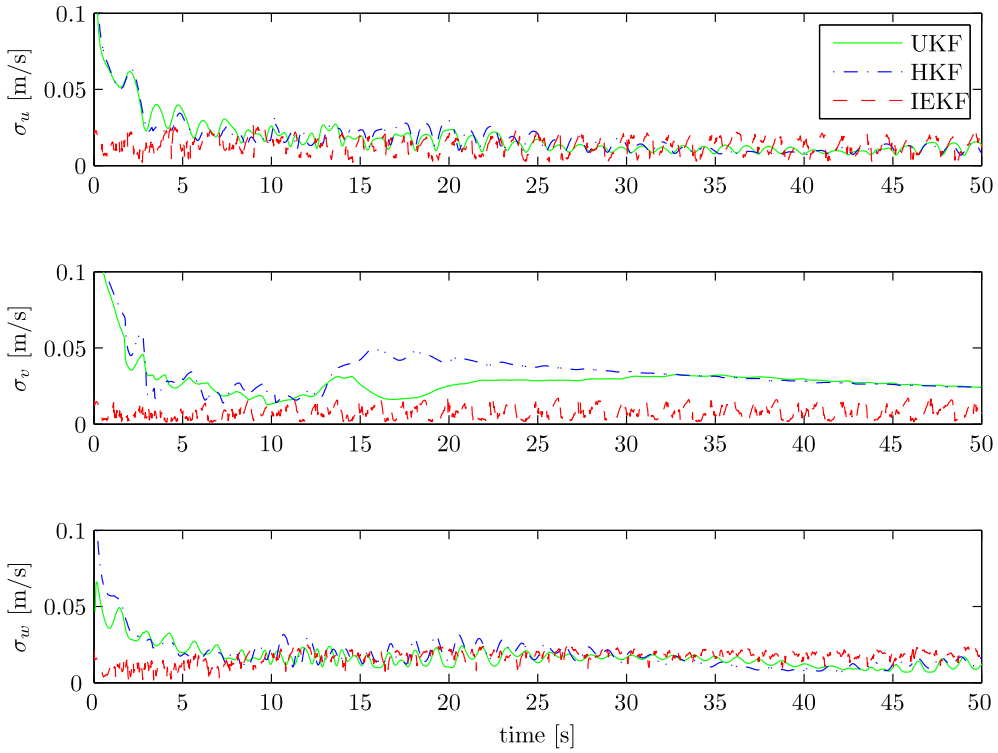


Figure C-16: Standard deviation estimates of the velocities @ 50 Hz

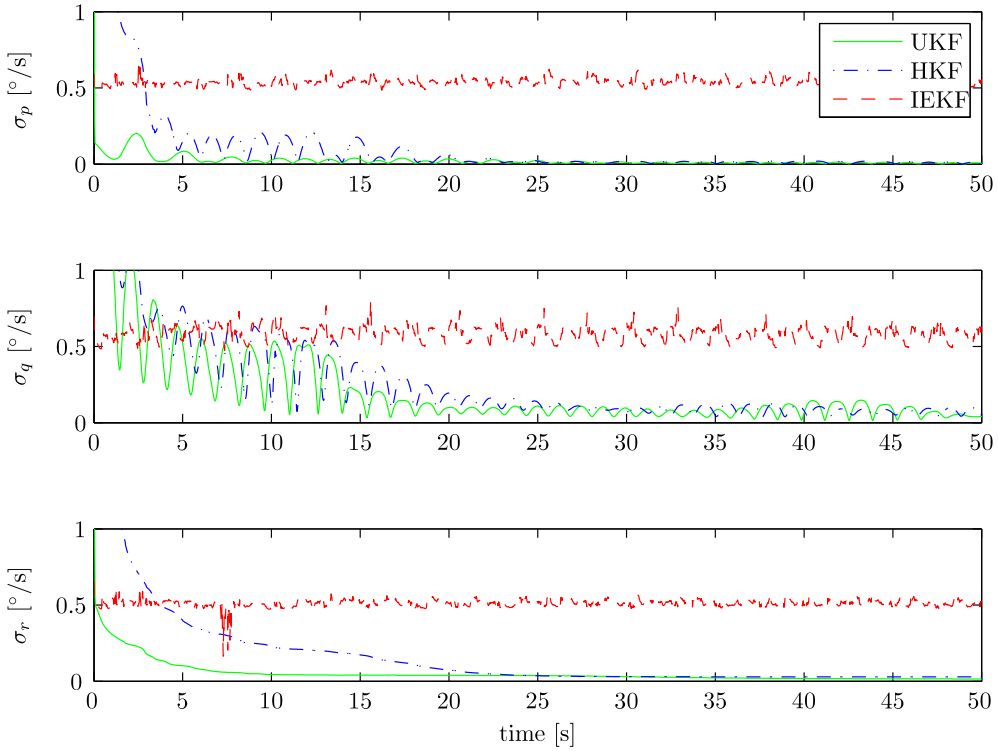


Figure C-17: Standard deviation estimates of the rotations @ 50 Hz

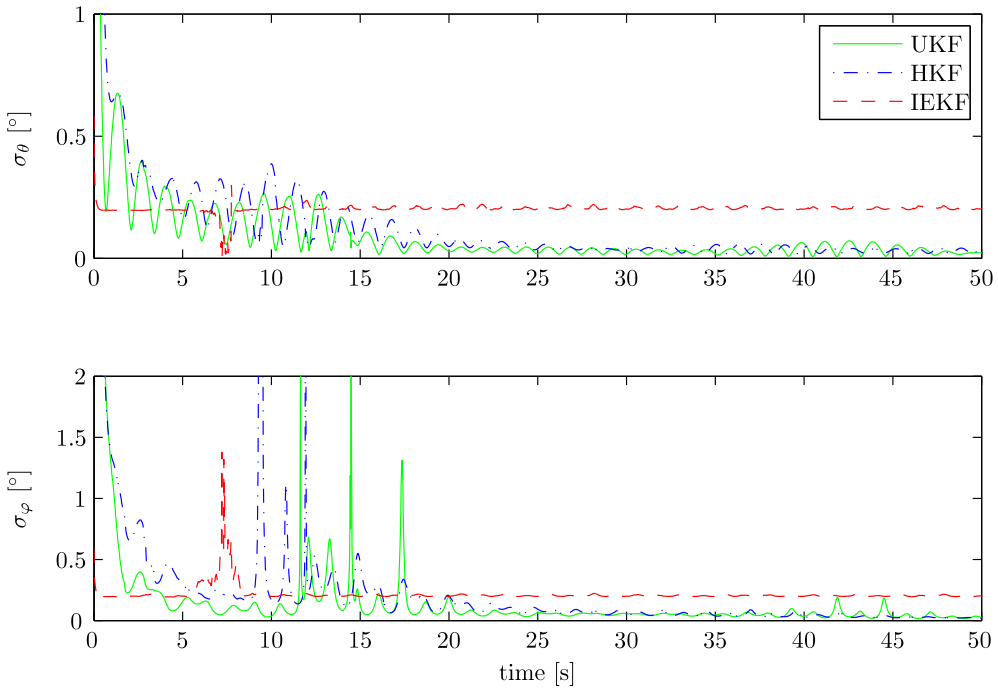


Figure C-18: Standard deviation estimates of the attitude @ 50 Hz

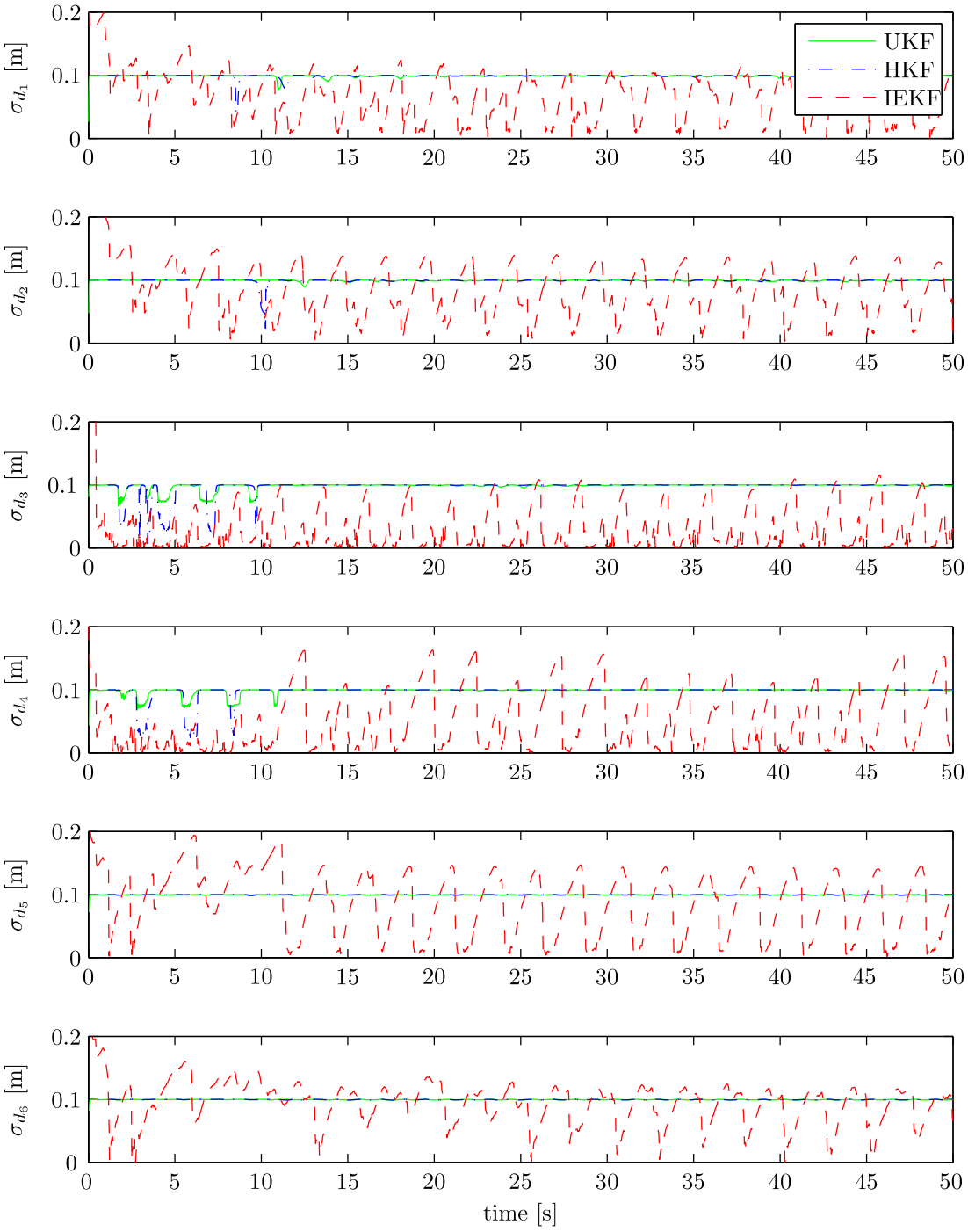


Figure C-19: Standard deviation estimates of the distances @ 50 Hz

C-2 State Estimates, Condition Numbers and Standard Deviations from a Data Recording at 25 Hz

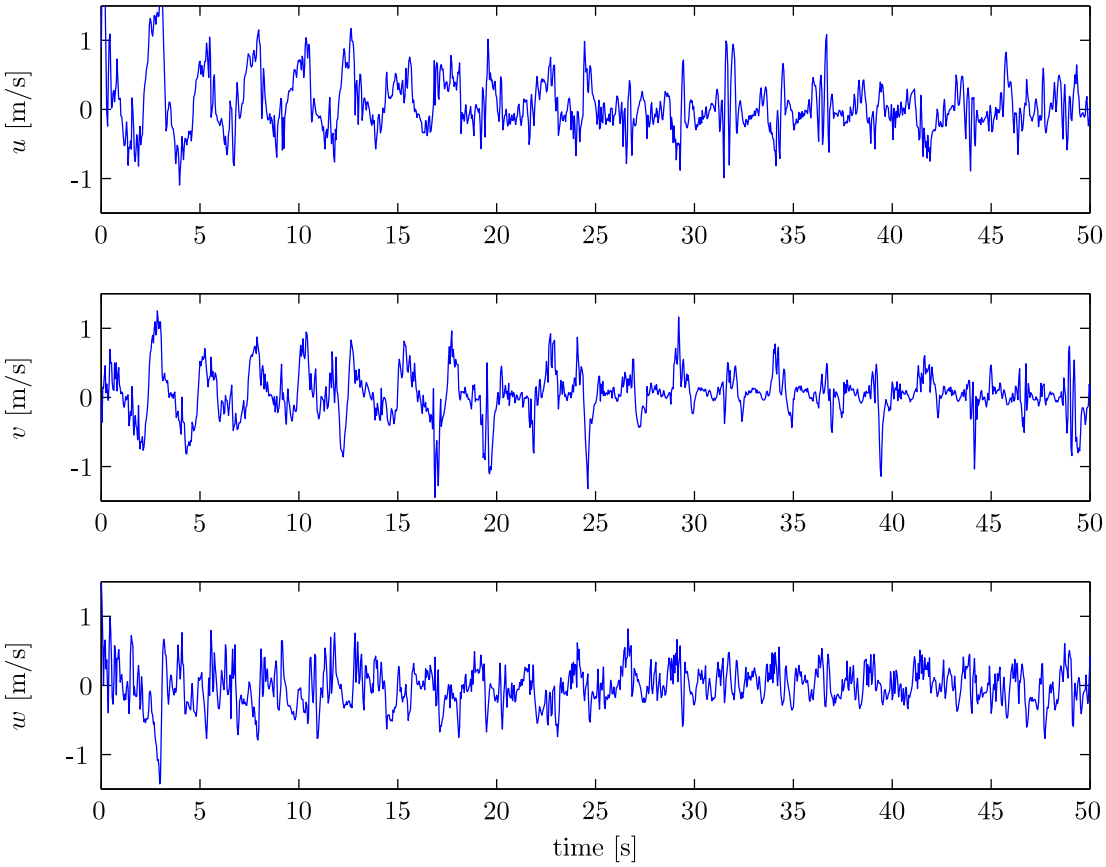


Figure C-20: IEKF velocity estimates @ 25 Hz

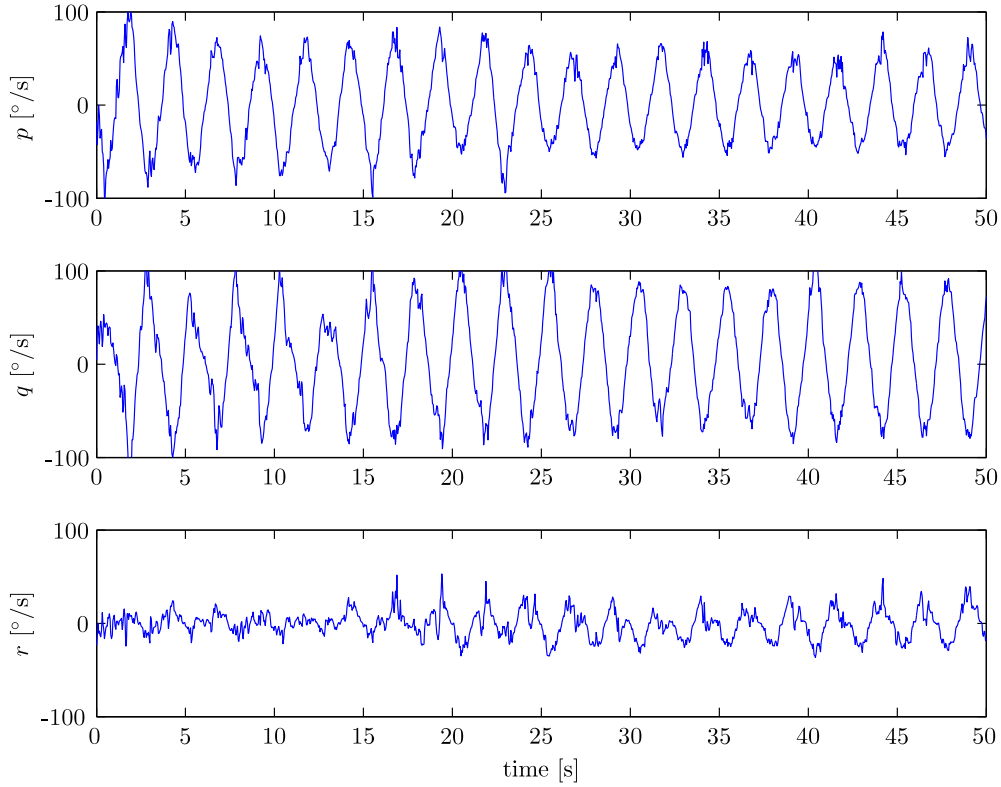


Figure C-21: IEKF rotation estimates @ 25 Hz

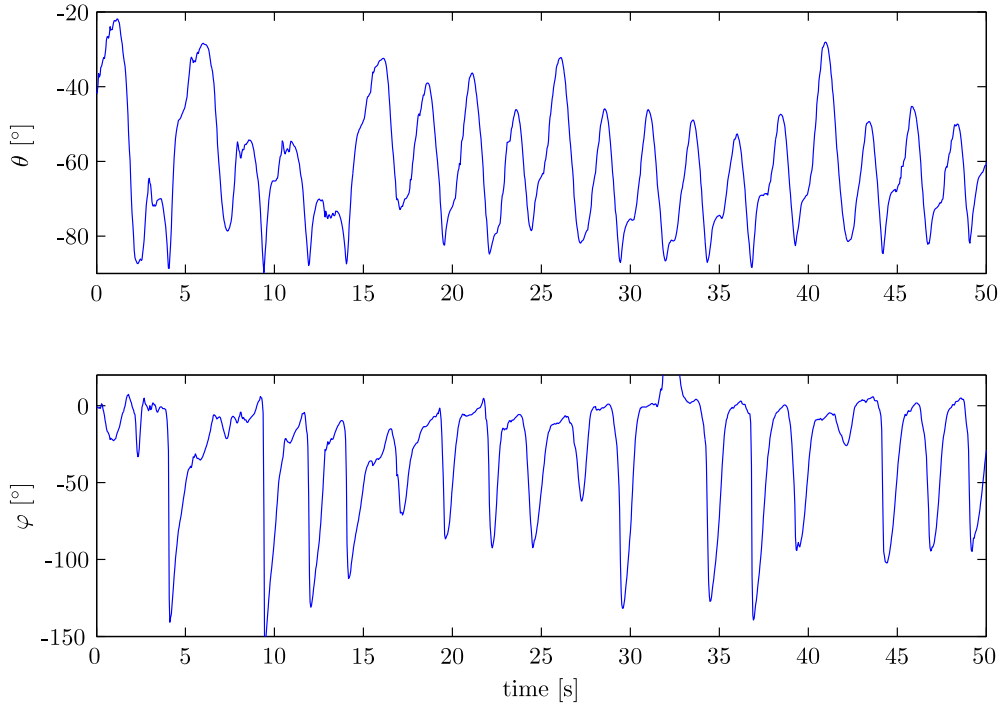


Figure C-22: IEKF attitude estimates @ 25 Hz

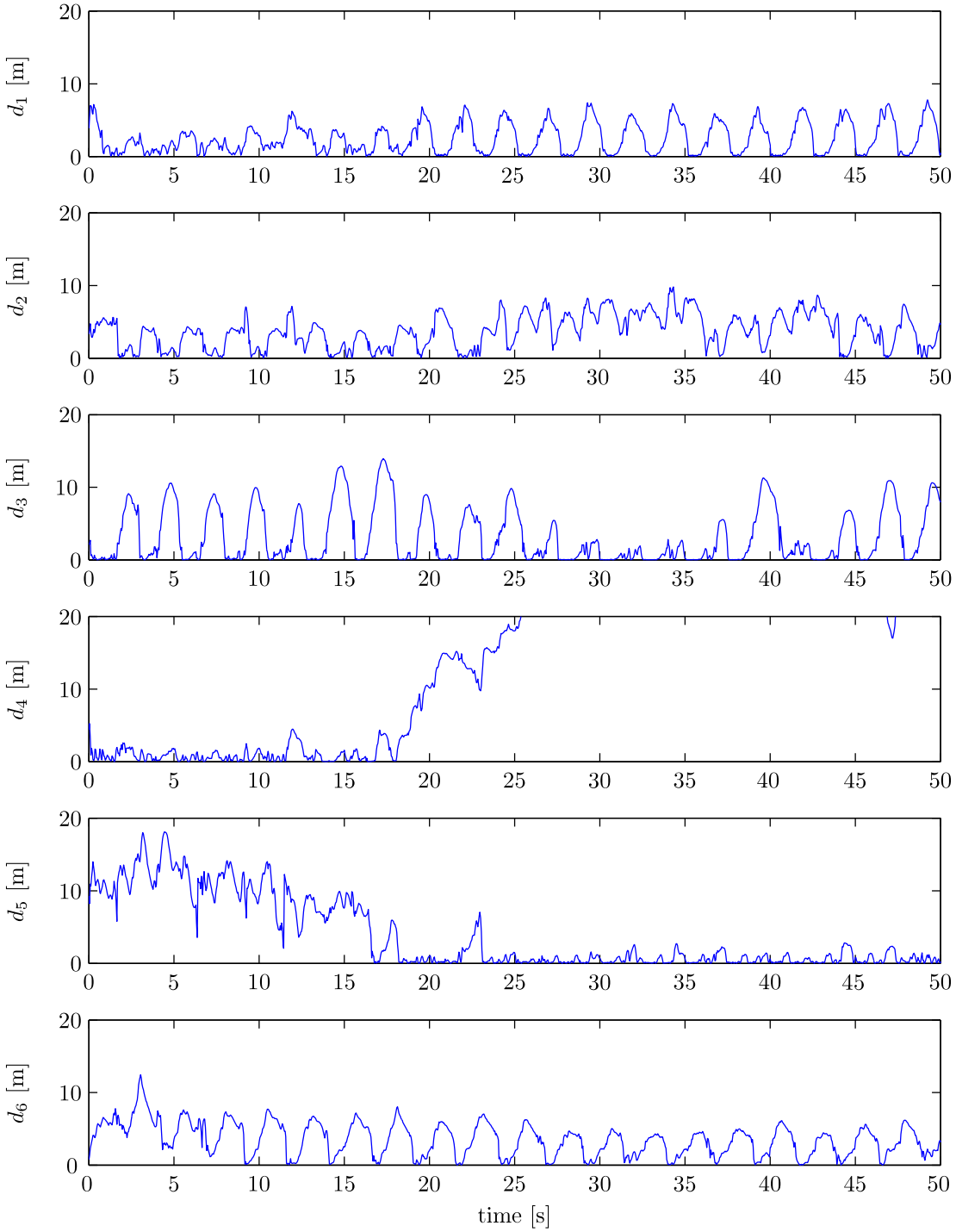


Figure C-23: IEKF distance estimates @ 25 Hz

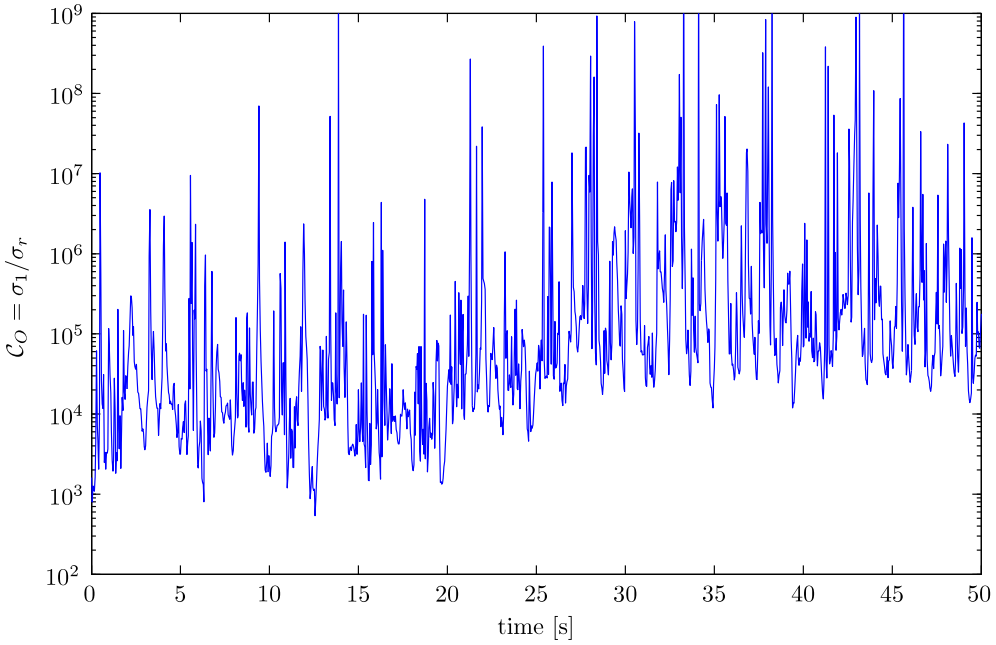


Figure C-24: IEKF condition number @ 25 Hz

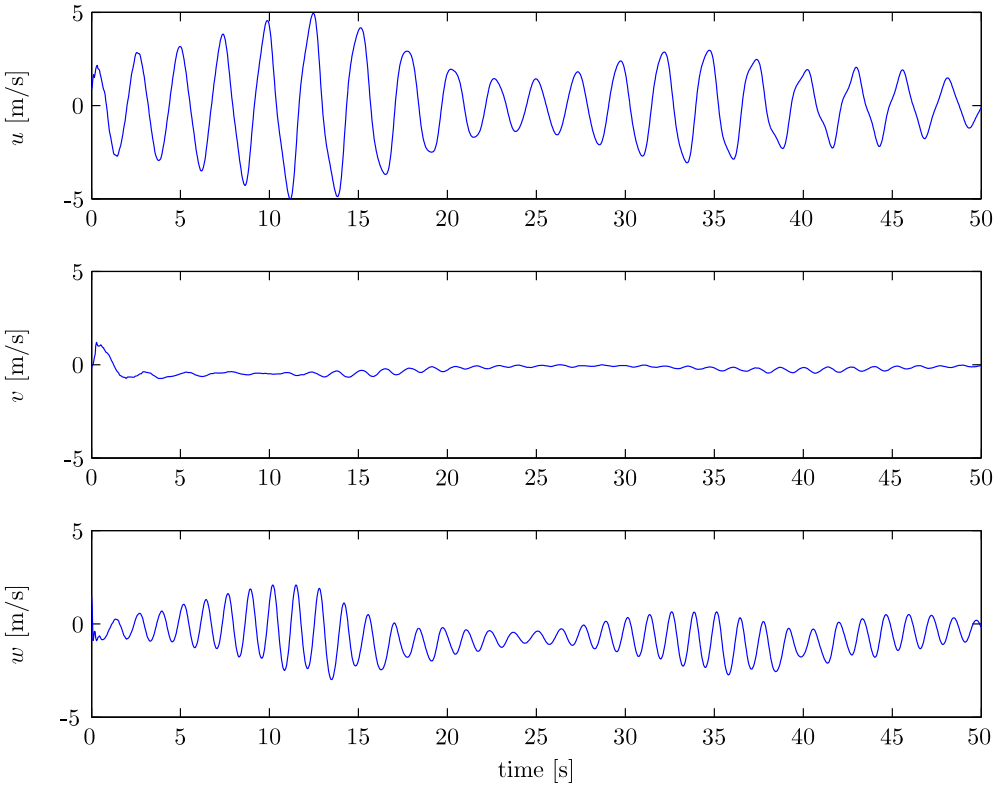


Figure C-25: UKF velocity estimates @ 25 Hz

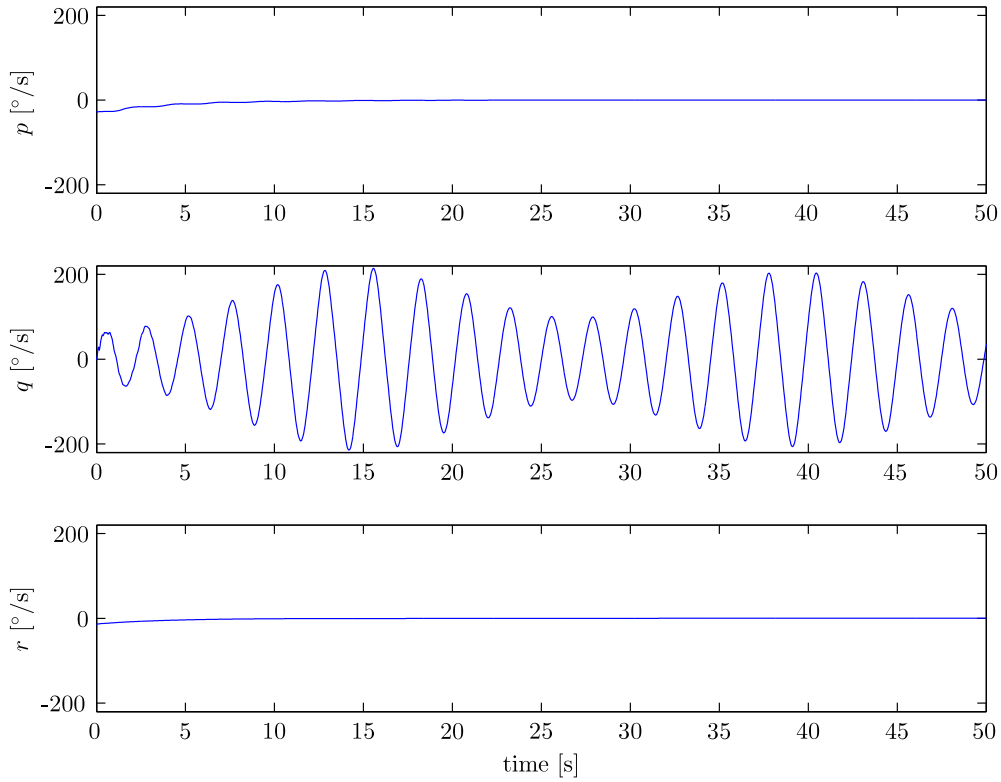


Figure C-26: UKF rotation estimates @ 25 Hz

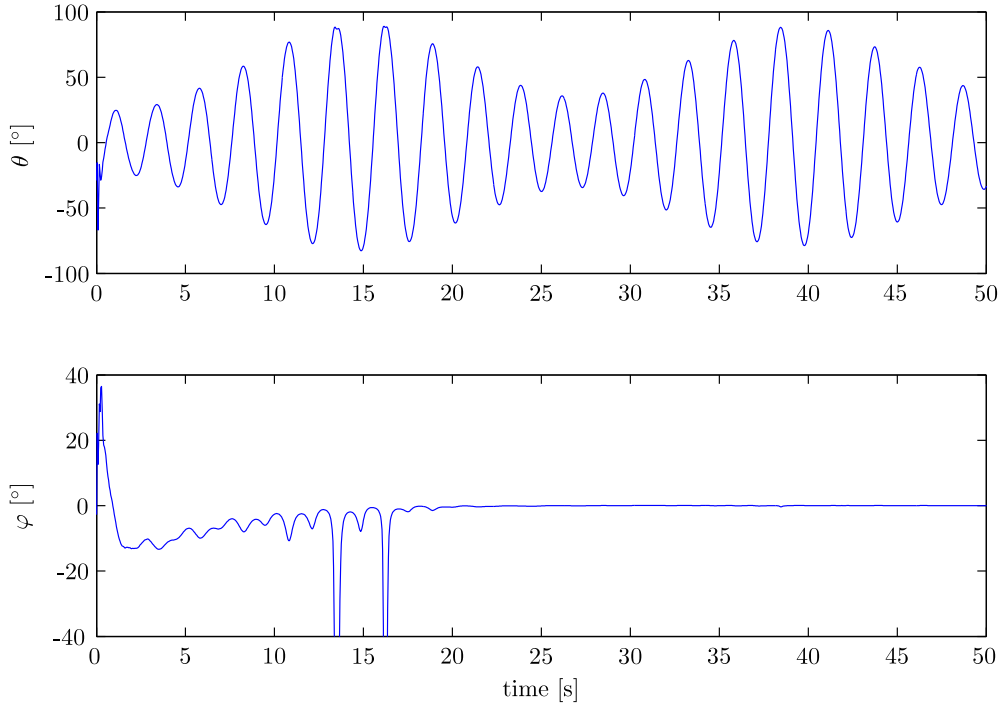


Figure C-27: UKF attitude estimates @ 25 Hz

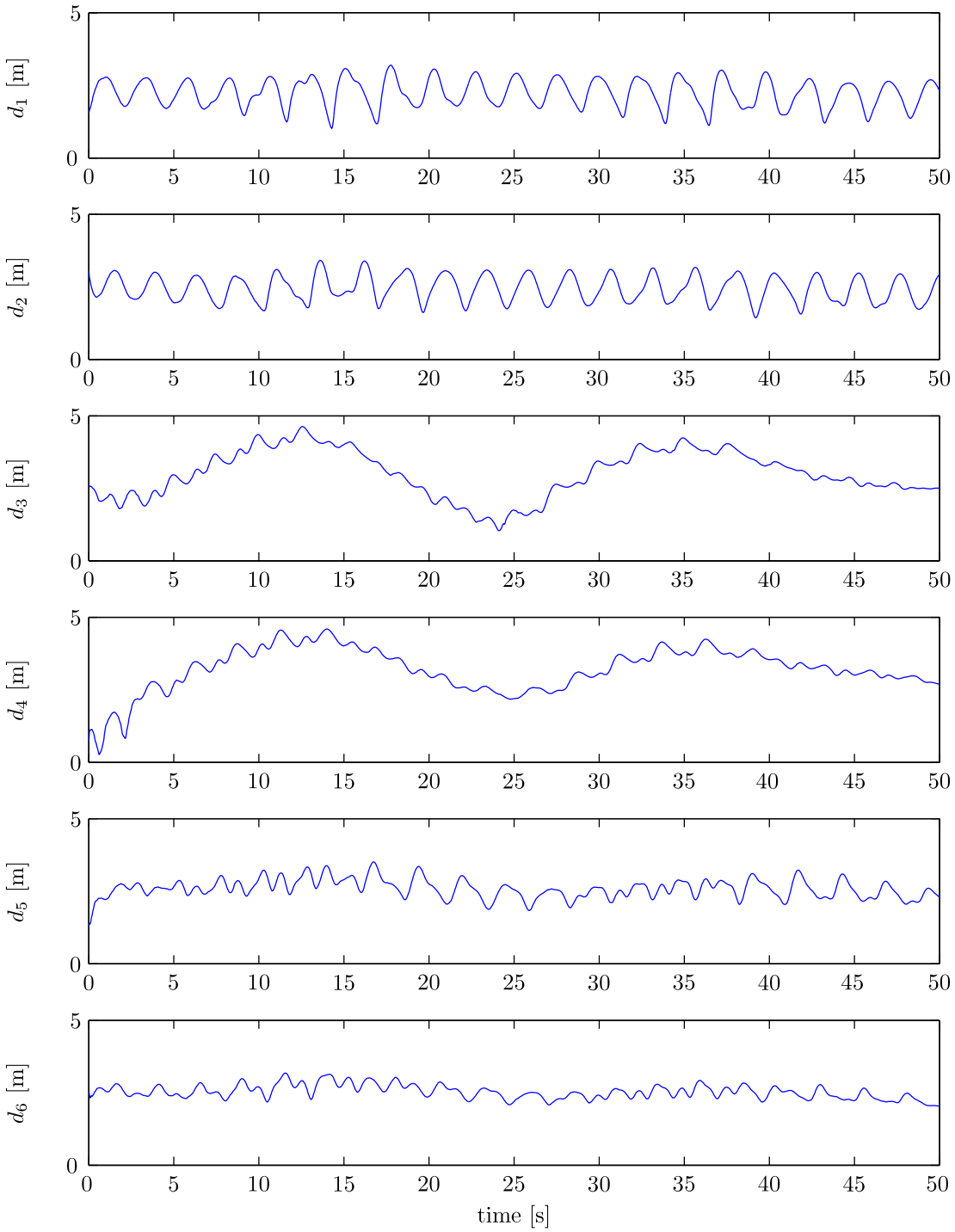


Figure C-28: UKF distance estimates @ 25 Hz

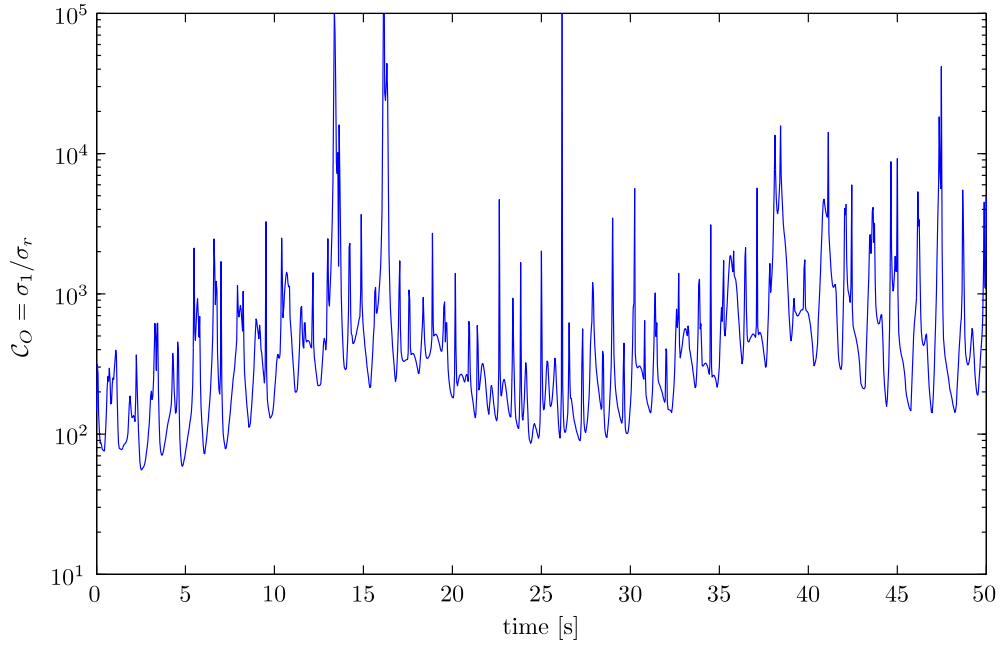


Figure C-29: UKF condition number @ 25 Hz

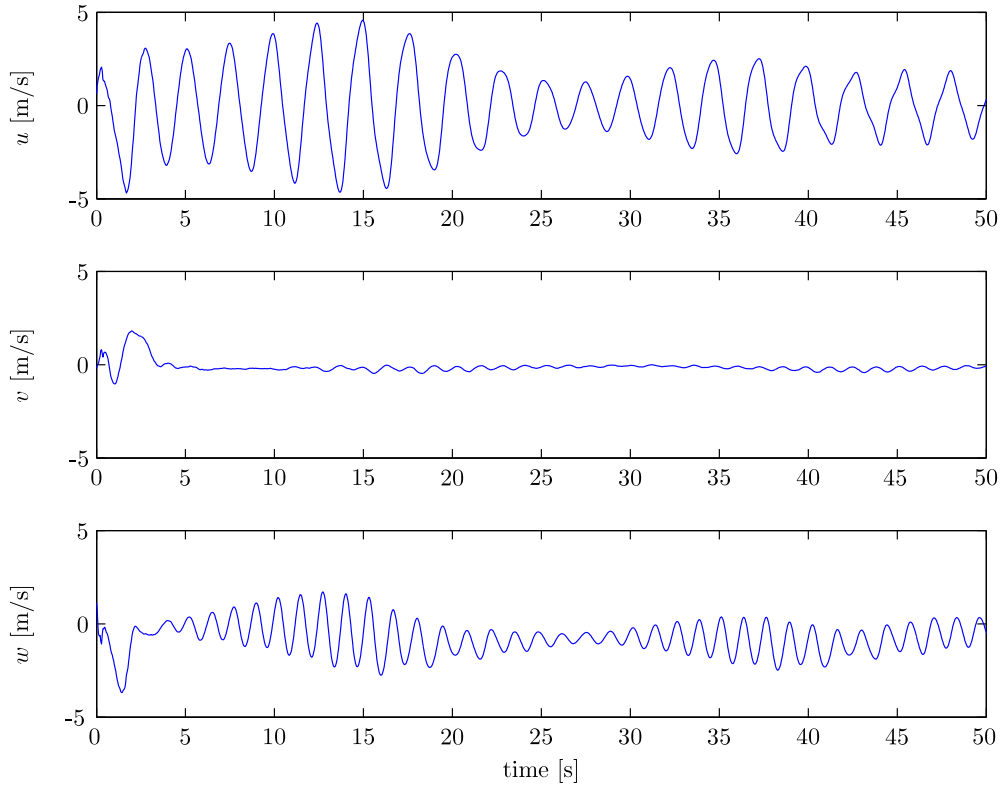


Figure C-30: HKF velocity estimates @ 25 Hz

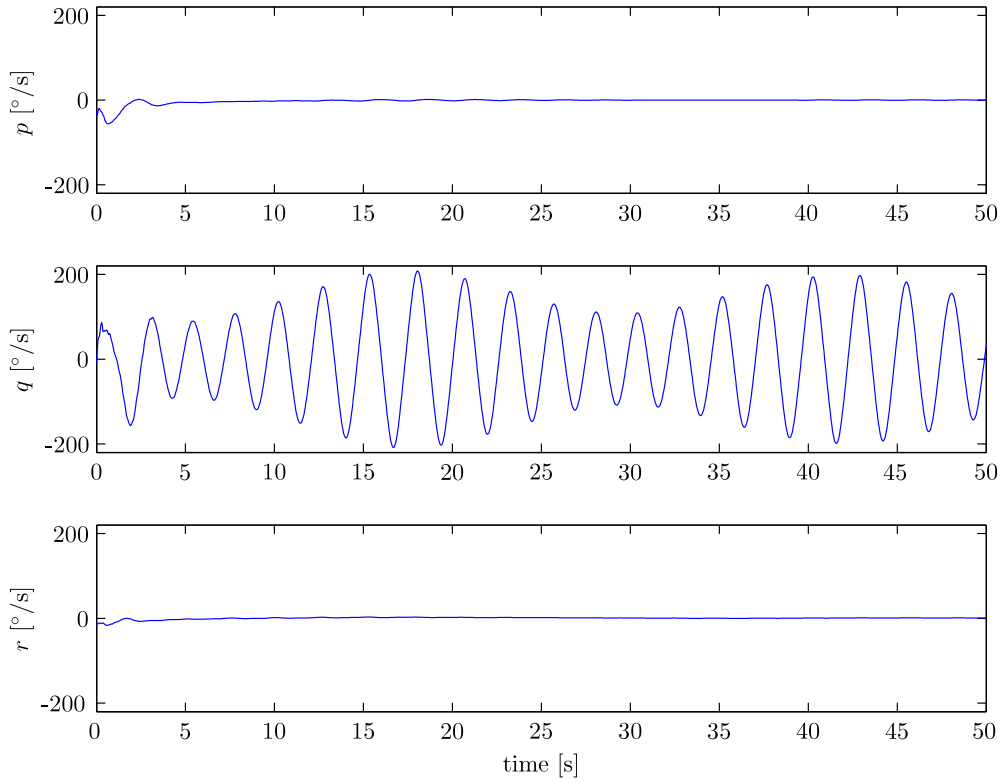


Figure C-31: HKF rotation estimates @ 25 Hz

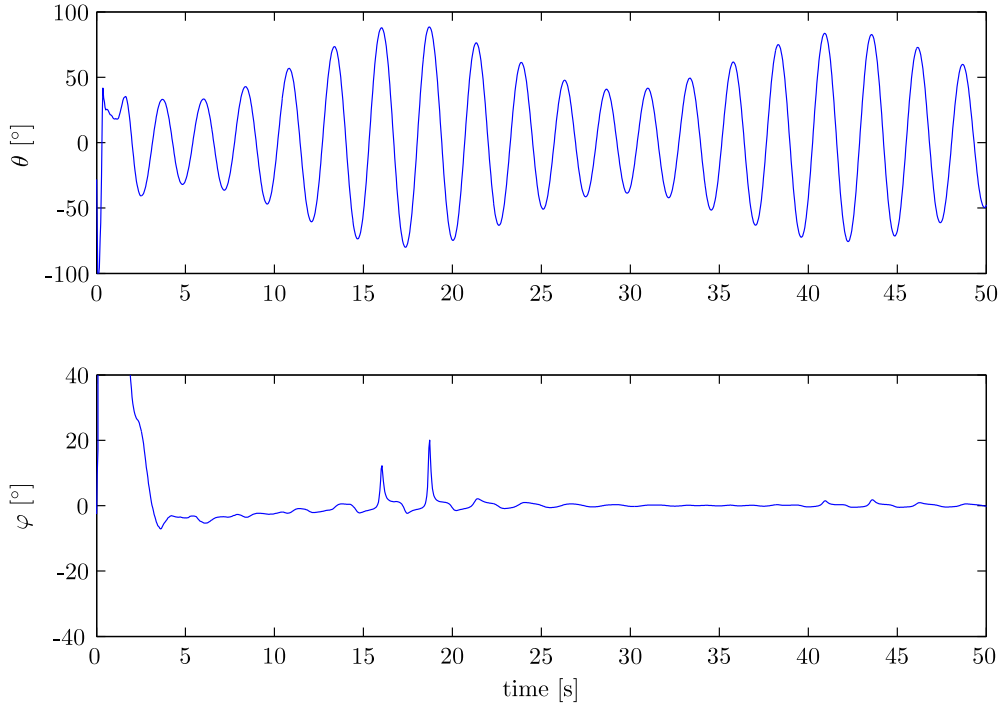


Figure C-32: HKF attitude estimates @ 25 Hz

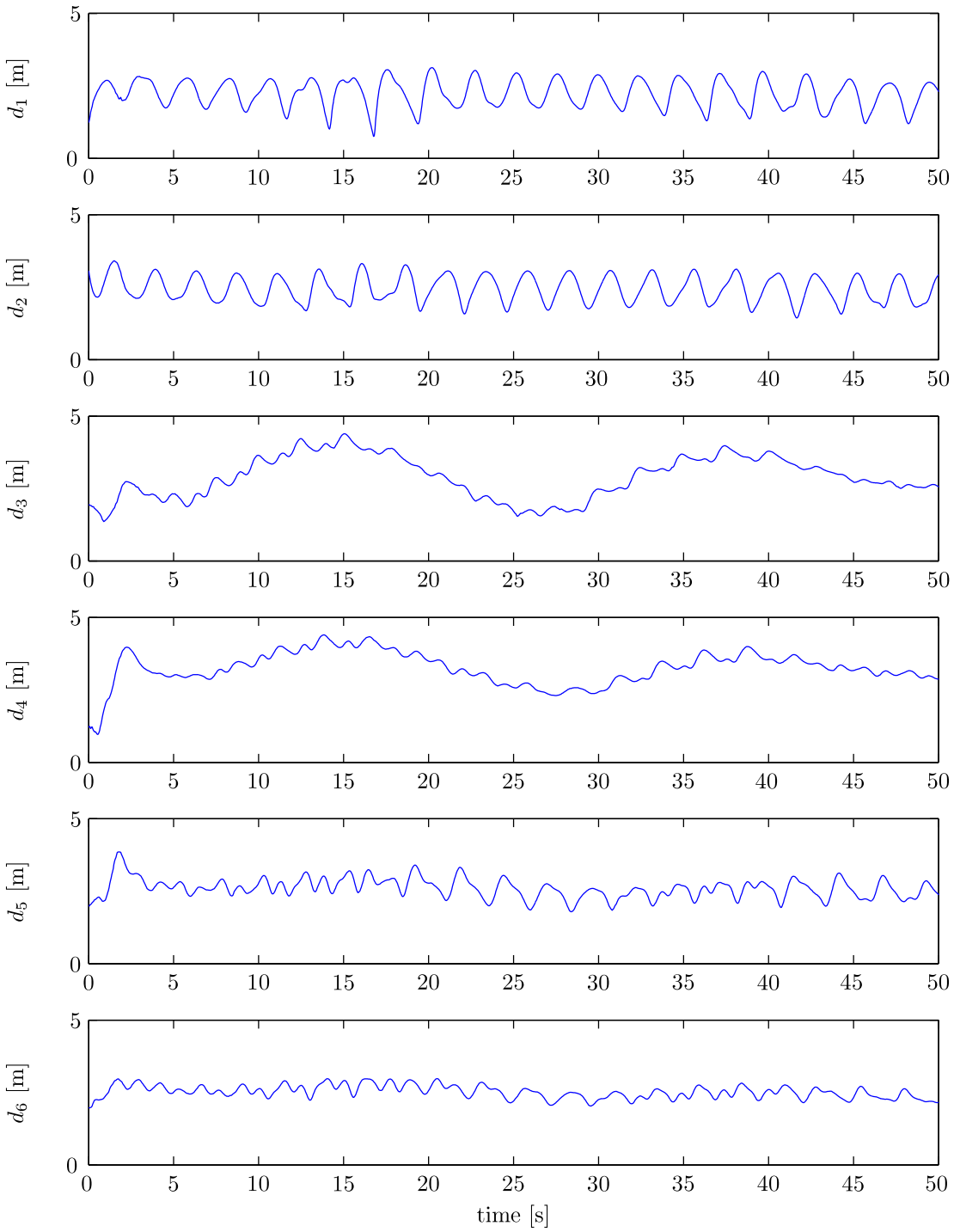


Figure C-33: HKF distance estimates @ 25 Hz

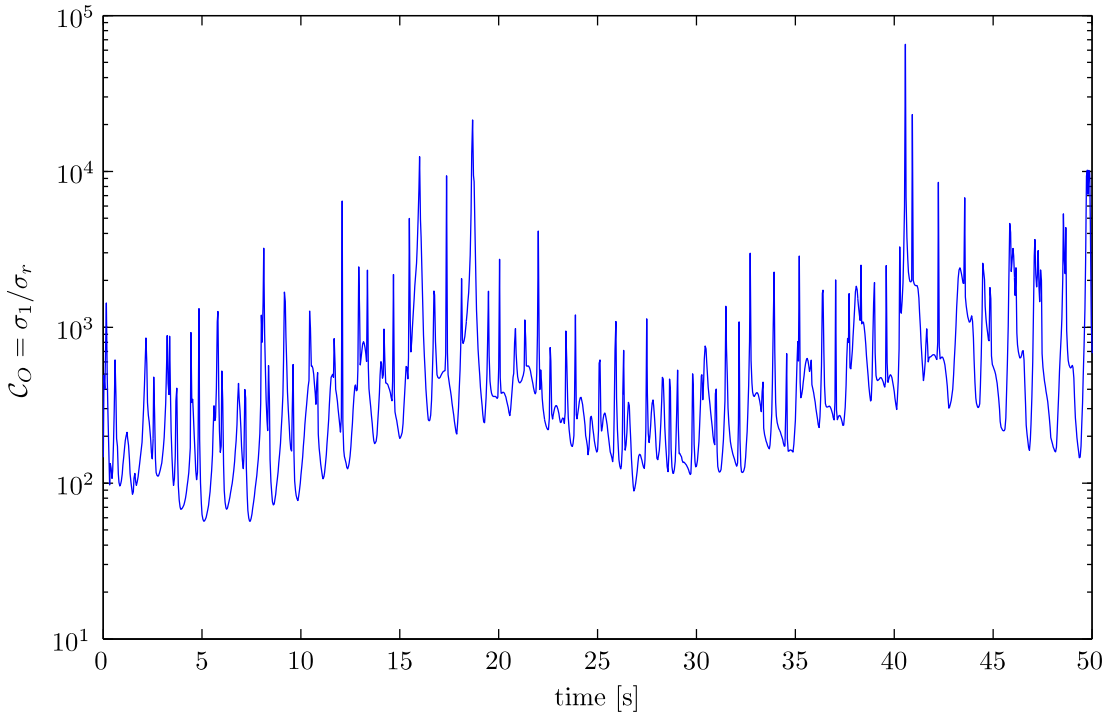


Figure C-34: HKF condition number @ 25 Hz

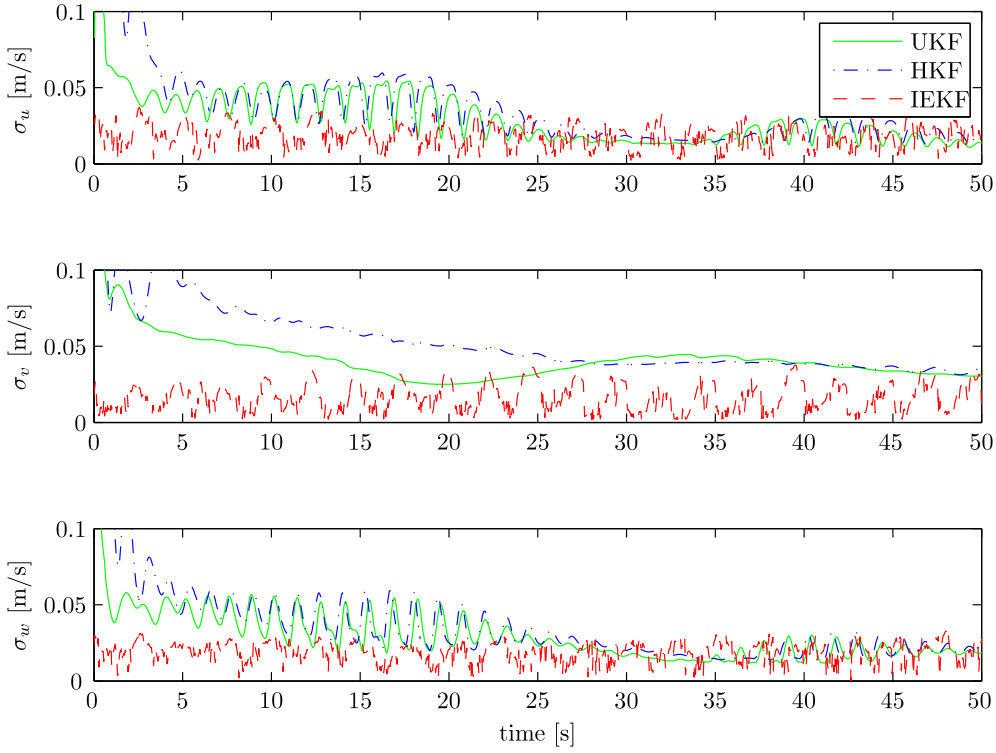


Figure C-35: Standard deviation estimates of the velocities @ 25 Hz

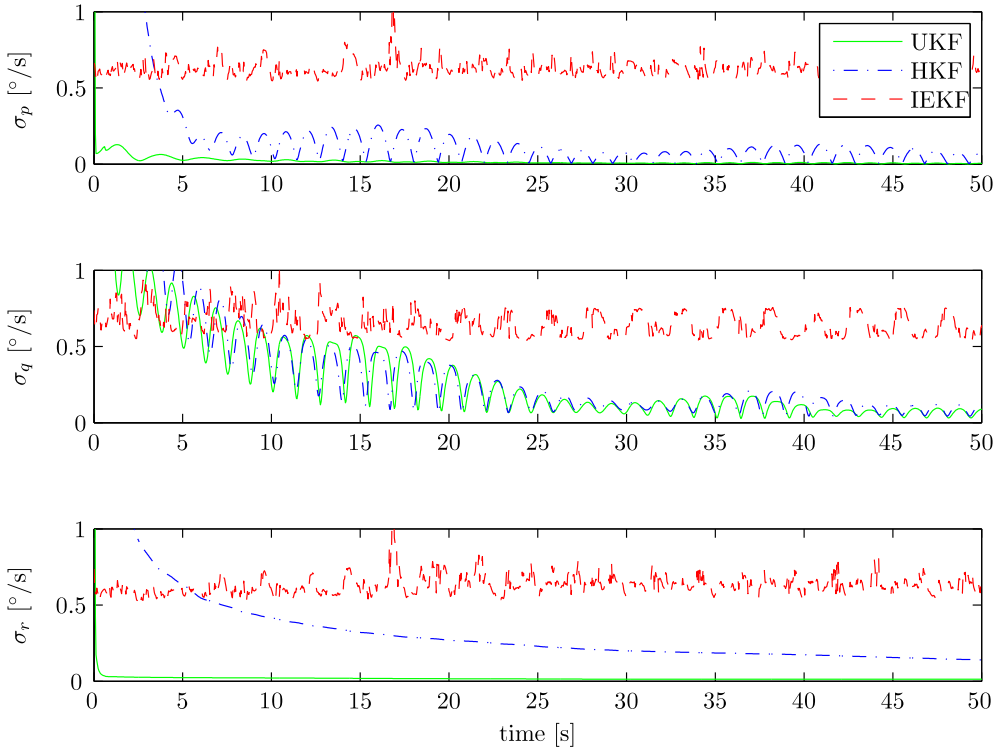


Figure C-36: Standard deviation estimates of the rotations @ 25 Hz

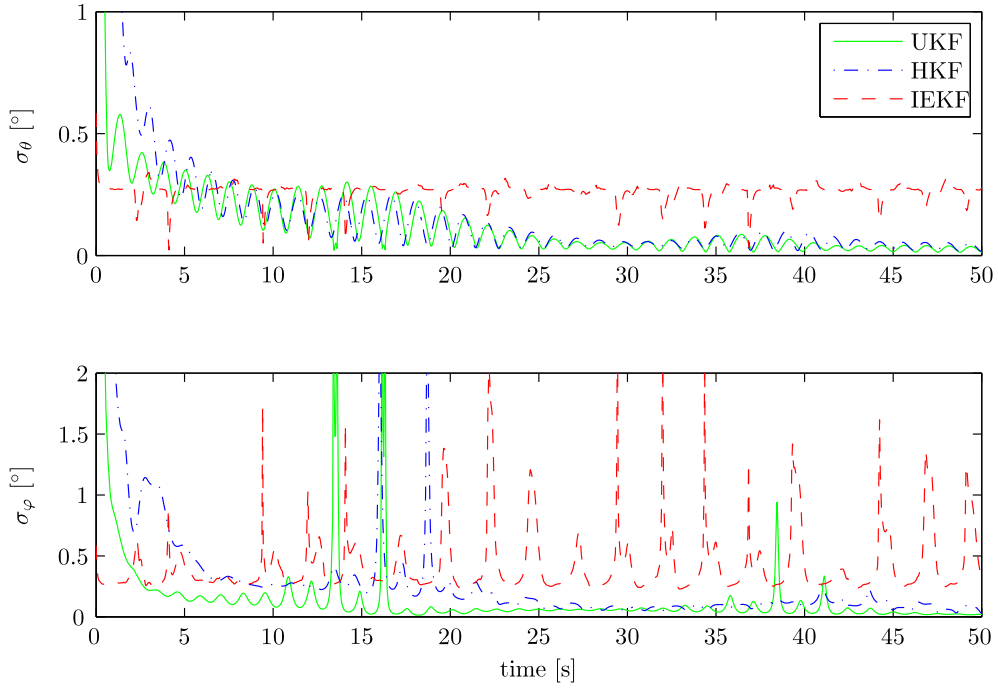


Figure C-37: Standard deviation estimates of the attitude @ 25 Hz

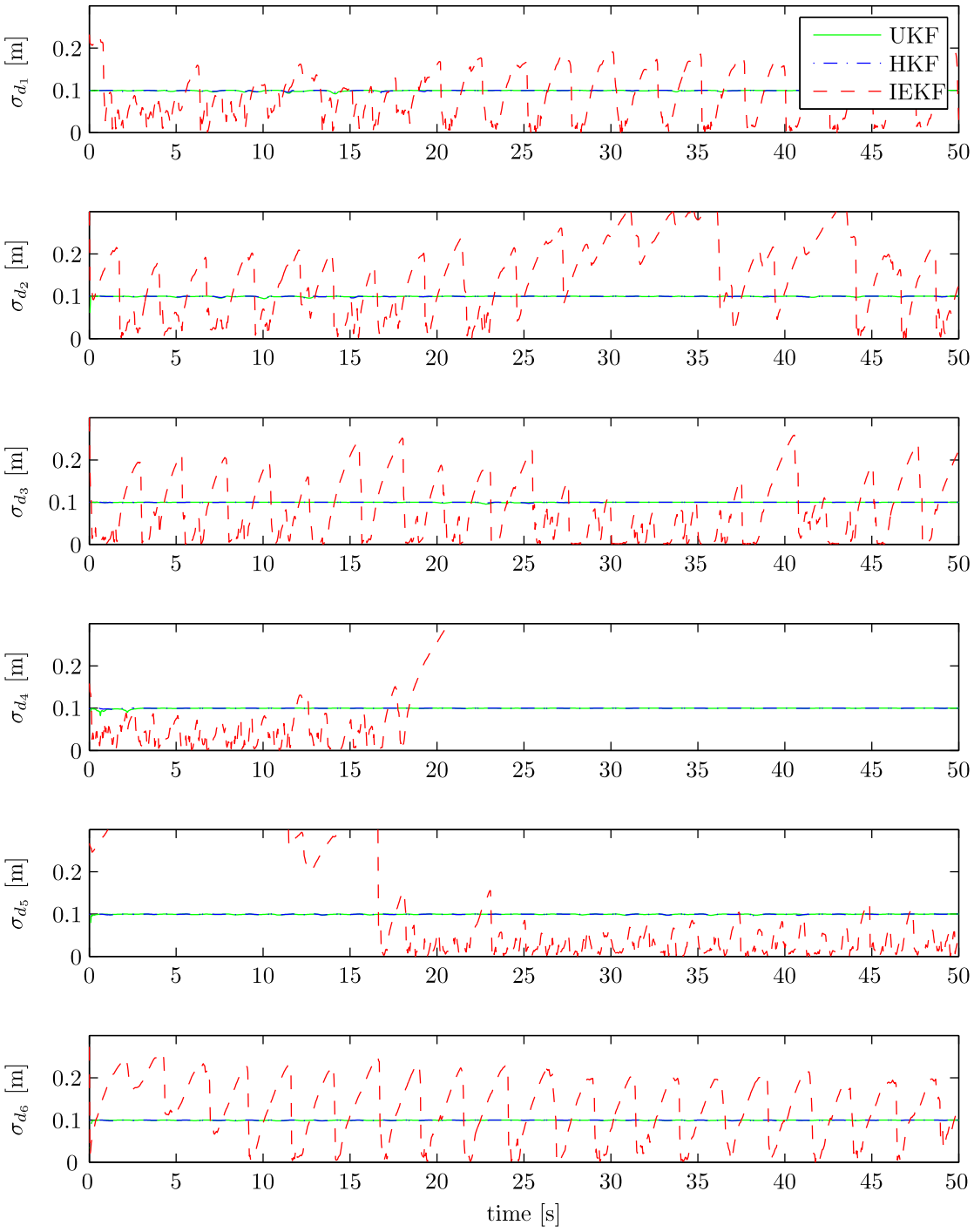


Figure C-38: Standard deviation estimates of the distances @ 25 Hz

Appendix D

Code Listings

D-1 Flow Diagram of the Code Running on the Microcontroller

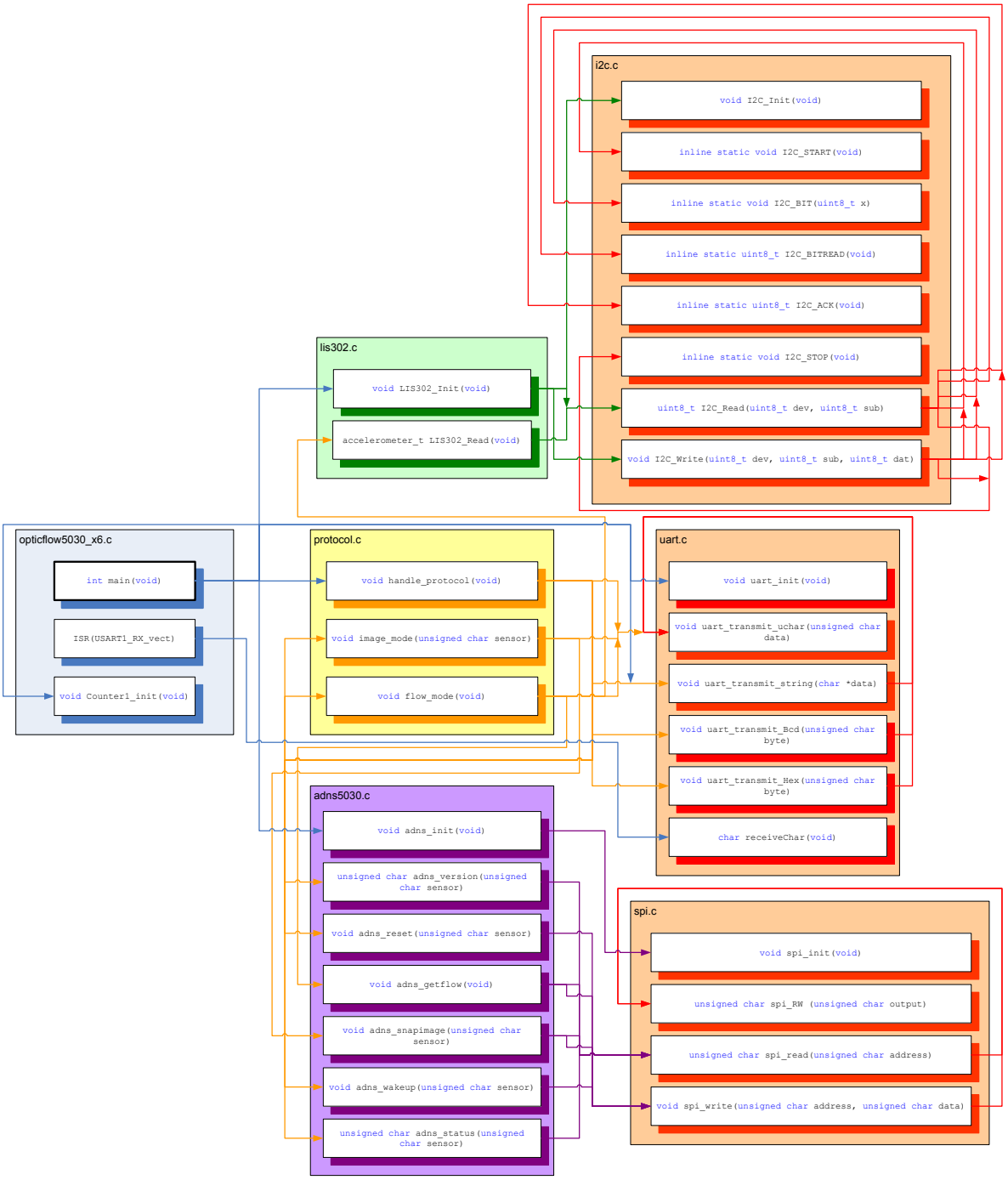


Figure D-1: Diagram of the functional flow of the microcontroller software. It shows all functions arranged by the files in which they are present. Each file groups functions with a particular purpose together. `opticflow5030_x6.c` contains the main function from where execution starts, `protocol.c` contains the main logic of the various operation modes, `lis302.c` controls the accelerometer IC using the Inter-Integrated Circuit Bus (I^2C) serial communication protocol contained in `i2c.c`, `adns5030.c` controls the optical flow ICs using the Serial Peripheral Interface Bus (SPI) protocol contained in `spi.c`, `uart.c` contains the Universal Asynchronous Receiver/Transmitter (UART) functions for communication with the host computer.
Each arrow represents a function call.

D-2 Embedded Matlab IEKF Algorithm

```
function Y = IEKF(U)

    Y = zeros(242,1);
    % Direction Cosine Matrix from SensorBoard to Pendulum
    SB2P = [-0.607873596459256,-0.655798968567627, 0.447680023625497;
            -0.406013862505740,-0.227812369076838,-0.885016535409802;
             0.682379977857542,-0.719742479864316,-0.127782348146782];
    % Reconstruct inputs P_k, yhat_k, u, omega, params, dt, g
    P_k = reshape(U(1:196),14,14);
    yhat_k = U(197:210);
    u = U(211:213);
    omega = U(214:225);
    accel = U(226:228);
    delta = 0.158;
    accel_p = SB2P*(accel + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 + yhat_k
        (5)^2)]);
    params = U(229:233);
    dt = U(234);
    N = round(dt*3000)+2;
    g = U(235);
    tuning = U(236:244);
    u_k = [(SB2P*(U(245:247) + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 +
        yhat_k(5)^2)]));U(248:250)];

    tp_Q1 = tuning(1);
    tp_Q2 = tuning(2);
    tp_Qd = tuning(3);
    tp_Qatt = tuning(4);
    tp_R = tuning(5);
    tp_Ra = tuning(6);

    G = 0.6*eye(14);G(9:14,9:14) = eye(6);
    % max allowable 2-norm of the error in the filtered state iterator
    epsilon = 1e-10;
    % max number of iterations per timestep
    max_j = 20;

    % System noise:
    fm = tp_Q1*[1;1;1];%2*[1E-2;5E-3;7.5E-3];
    tm = tp_Q2*[1;1;1];%*[5.1885E-6;6.0479E-6;1.1018E-005];
    stdw = [fm;tm;tp_Qatt*[1;1];tp_Qd*ones(6,1)];
    Q = diag(stdw.^2);
    % Measurement noise:
    stdv = [tp_R*ones(12,1);tp_Ra*ones(3,1)];
    R = diag(stdv.^2);

    % One stage ahead prediction using RK4 integration
    x_predict = RK4(yhat_k,N,dt,params,u_k,[accel_p;u],g);

    % covariance matrix of the state prediction error vector
    F = dfdx(yhat_k,params);
    Phi = expm(F*dt);
    Gamma = (Phi*dt)*G;
    P_predict = Phi*P_k*Phi' + Gamma*Q*Gamma';

    % Iterate the nominal states
    eta2 = x_predict;
    error = 2*epsilon;      % initiate error with a value larger than epsilon
    j = 0;
    Vdot = zeros(3,1);
    P = P_predict;

    while error > epsilon
        if  j >= max_j
            break
        end
        j = j+1;
        eta1 = eta2;
        H = dhdx(eta1,g,delta);
        % Kalman gain matrix
        [K,P] = SquareRootKalmanGain(H,P_predict,R);
%         K = P_predict*H'*inv(H*P_predict*H' + R);
        % measurement update
        z_kplus1 = [omega;accel];
        Vdot(1) = accel_p(1)-eta1(5)*eta1(3)+eta1(6)*eta1(2)-g*sin(eta1(7));
        Vdot(2) = accel_p(2)-eta1(6)*eta1(1)+eta1(4)*eta1(3)+g*sin(eta1(8))*cos(eta1(7));
        Vdot(3) = accel_p(3)-eta1(4)*eta1(2)+eta1(5)*eta1(1)+g*cos(eta1(8))*cos(eta1(7));
%         Vdot = (eta1(1:3) - yhat_k(1:3))/dt;
        eta2 = x_predict + K*(z_kplus1 - func2(eta1,Vdot,g,delta) - H*(x_predict-eta1));
        error = norm((eta2-eta1),inf)/norm(eta1,inf);
    end
```

```
    yhat = eta2;

    sd = sqrt(diag(P));
    F = dfdx(yhat,params);
    H = dhdx(yhat,g,delta);
    % Observability matrix
    W = zeros(15*14,14);W(1:15,:) = H;
    term = H;
    for ii = 1:13
        term = term*F;
        W((1+15*ii):(15 + 15*ii),:) = term;
    end
    sv = svd(W);
    tol = max(size(W))*eps(max(sv));
    r = sum(sv > tol);
    condition = sv(1)/sv(r);

    % Construct output vector Y
    Y(1:196) = reshape(P,196,1);
    Y(197:210) = yhat;
    Y(211:224) = sd;
    Y(225:238) = sv;
    Y(239) = r;
    Y(240) = condition;
    Y(241) = j;
end

function y = RK4(x0,N,dt,params,u_k,u_k_1,g)

    %   RK4(X0,N,DT,PARAMS,U_K,U_K_1,G)
    %     RK4 algorithm. Integrates a hard coded function func1.m

    h = dt/(N - 1);
    y = x0;
    K = zeros(length(x0),4);

    for i = 1:N-1
        u = u_k + (i-1)/(N-2)*(u_k_1 - u_k);
        K(:,1) = h*func1(y,u,params,g);
        K(:,2) = h*func1(y+.5*K(:,1),u,params,g);
        K(:,3) = h*func1(y+.5*K(:,2),u,params,g);
        K(:,4) = h*func1(y+K(:,3),u,params,g);
        y = y + (K(:,1) + 2*K(:,2) + 2*K(:,3) + K(:,4))/6;
    end
end

function xdot = func1(x,U,params,g)

    % x is the state vector: [u v w p q r theta phi d1...d6]',
    % U is the input vector: [Fx Fy Fz Mx My Mz]',
    % params is a vector with constants: [Ix Iy Iz Jxz m]'

    xdot = zeros(14,1);
    den = -params(4)^2 + params(1)*params(3);
    xdot(1) = U(1)-x(5)*x(3)+x(6)*x(2) - g*sin(x(7));
    xdot(2) = U(2)-x(6)*x(1)+x(4)*x(3) + g*sin(x(8))*cos(x(7));
    xdot(3) = U(3)-x(4)*x(2)+x(5)*x(1) + g*cos(x(8))*cos(x(7));
    xdot(4) = (U(4)*params(3) - x(5)*x(6)*params(3)*(params(3) - params(2))...
            + x(4)*x(5)*params(4)*(params(1) - params(2) + params(3))...
            + params(4)*(U(6) - params(4)*x(6)*x(5)))/den;
    xdot(5) = (U(5) - (params(1) - params(3))*x(6)*x(4) - ...
              (x(4)^2 - x(6)^2)*params(4))/params(2);
    xdot(6) = (U(6)*params(1) + x(4)*x(5)*params(1)*(params(1) - params(2))...
            - x(6)*x(5)*params(4)*(params(1) - params(2) + params(3))...
            + params(4)*(U(4) + params(4)*x(4)*x(5)))/den;
    xdot(7) = x(5)*cos(x(8)) - x(6)*sin(x(8));
    xdot(8) = x(4) + x(5)*sin(x(8))*tan(x(7)) + x(6)*cos(x(8))*tan(x(7));
end

function z = func2(x,Vdot,g,delta)
    % The observation equations with a rotated sensor board
    P2SB = [-0.607873596459256,-0.406013862505740, 0.682379977857542;
            -0.655798968567627,-0.227812369076838,-0.719742479864316;
             0.447680023625497,-0.885016535409802,-0.127782348146782];
    x_sb = zeros(6,1);
    % velocity in the sensor board frame
    x_sb(1:3) = P2SB*(x(1:3) + cross(x(4:6),[0;0;-delta]));
    % rotation in the sensor board frame
    x_sb(4:6) = P2SB*x(4:6);
    u     = x_sb(1);
    v     = x_sb(2);
    w     = x_sb(3);
    p     = x_sb(4);
    q     = x_sb(5);
    r     = x_sb(6);
    theta = x(7);
    phi   = x(8);
    d1    = x(9);
    d2    = x(10);
    d3    = x(11);
```

```
    d4    = x(12);
    d5    = x(13);
    d6    = x(14);
    gvec = g*[-sin(theta);sin(phi)*cos(theta);cos(phi)*cos(theta)];
    A_sb = P2SB*((Vdot + cross(x(4:6),x(1:3)) - gvec) +...
           delta*[x(4)*x(6);x(5)*x(6);-(x(4)^2+x(5)^2)]);

    z = [u/d1 - r;
         w/d1 + p;
         v/d2 - r;
         w/d2 + q;
         u/d3 + r;
         w/d3 - p;
         v/d4 + r;
         w/d4 - q;
         u/d5 - q;
         v/d5 + p;
         u/d6 + q;
         v/d6 - p;
         A_sb];
end

function F = dfdx(x,params)

% The jacobian of the dynamics
    u = x(1);
    v = x(2);
    w = x(3);
    p = x(4);
    q = x(5);
    r = x(6);
    theta = x(7);
    phi   = x(8);
    Ix = params(1);
    Iy = params(2);
    Iz = params(3);
    Jxz = params(4);

    F = zeros(14,14);
    F(1:8,1:8) = [0,r,-q,0,-w,v,0,0;
                  -r,0,p,w,0,-u,0,0;
                  q,-p,0,-v,u,0,0,0;
                  0,0,0,Jxz*((-Iy+Ix)*q/Iz+q)/(Ix-Jxz^2/Iz),((-Iz+Iy)*r+Jxz*(((-Iy+Ix)*p-Jxz*r)/...
                  0,0,0,((-Ix+Iz)*r-2*Jxz*p)/Iy,0,((-Ix+Iz)*p+2*Jxz*r)/Iy,0,0;
                  0,0,0,((-Iy+Ix)*q+Jxz^2*q/Ix)/(Iz-Jxz^2/Ix),((-Iy+Ix)*p+Jxz*(((-Iz+Iy)*r+Jxz*p...
                  0,0,0,0,cos(phi),-sin(phi),0,-q*sin(phi)-r*cos(phi);
                  0,0,0,1,sin(phi)*tan(theta),cos(phi)*tan(theta),q*sin(phi)*(1+tan(theta)^2)+r*...
end

function H = dhdx(x,g,delta)

% in vehicle reference frame
    u     = x(1);
    v     = x(2);
    w     = x(3);
    p     = x(4);
    q     = x(5);
    r     = x(6);
    phi   = x(7);
    theta = x(8);
    d1    = x(9);
    d2    = x(10);
    d3    = x(11);
    d4    = x(12);
    d5    = x(13);
    d6    = x(14);

% Jacobian in case with rotated states
    H = [-.6077482419736110/d1,-.4059806710008880/d1,.6823822753390010/d1,-.4059806710008880*...
          .4476800236254970/d1,-.8848370081747490/d1,-.1277564272619920/d1,-.8848370081747490...
         -.6556637310069400/d2,-.2278451690760790/d2,-.7197449031402851/d2,-.2278451690760790...
          .4476800236254970/d2,-.8848370081747490/d2,-.1277564272619920/d2,-.8848370081747490...
         -.6077482419736110/d3,-.4059806710008880/d3,.6823822753390010/d3,-.4059806710008880*...
          .4476800236254970/d3,-.8848370081747490/d3,-.1277564272619920/d3,-.8848370081747490...
         -.6556637310069400/d4,-.2278451690760790/d4,-.7197449031402851/d4,-.2278451690760790...
          .4476800236254970/d4,-.8848370081747490/d4,-.1277564272619920/d4,-.8848370081747490...
         -.6077482419736110/d5,-.4059806710008880/d5,.6823822753390010/d5,-.4059806710008880*...
         -.6556637310069400/d5,-.2278451690760790/d5,-.7197449031402851/d5,-.2278451690760790...
         -.6077482419736110/d6,-.4059806710008880/d6,.6823822753390010/d6,-.4059806710008880*...
         -.6556637310069400/d6,-.2278451690760790/d6,-.7197449031402851/d6,-.2278451690760790...
         -.4059806710008880*r-.6823822753390010*q,.6077482419736110*r+.6823822753390010*p,-.6...
         -.2278451690760790*r+.7197449031402851*q,.6556637310069400*r-.7197449031402851*p,-.6...
         -.8848370081747490*r+.1277564272619920*q,-.4476800236254970*r-.1277564272619920*p,.4...
end

function [K,P] = SquareRootKalmanGain(H,P_predict,R)

    k = size(R,2);
    l = size(P_predict,1);
    srP_predict = chol(P_predict')';
```

```
    srR = chol(R')';
    A = [H*srP_predict        -srR;
         srP_predict    zeros(l,k)];
    [Qt,Rt] = qr(A');
    [m,n] = size(Rt);
    G = Rt(1:m-l,n-l+1:n)';
    sqRe = Rt(1:m-l,1:n-l)';
    K = G*inv(sqRe);
    srP = Rt(m-l+1:m,n-l+1:n)';
    P = srP*srP';
end
```

D-3 Embedded Matlab HKF Algorithm

```
function Y = HKF(U)

    Y = zeros(242,1);
    % Direction Cosine Matrix from SensorBoard to Pendulum
    SB2P = [-0.607873596459256,-0.655798968567627, 0.447680023625497;
            -0.406013862505740,-0.227812369076838,-0.885016535409802;
             0.682379977857542,-0.719742479864316,-0.127782348146782];
    % Reconstruct inputs P_k, yhat_k, u, omega, params, dt, g
    P_k = reshape(U(1:196),14,14);
    yhat_k = U(197:210);
    u = U(211:213);
    omega = U(214:225);
    accel = U(226:228);
    delta = 0.158;
    accel_p = SB2P*(accel + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 + yhat_k
        (5)^2)]);
    params = U(229:233);
    dt = U(234);
    N = round(dt*3000)+2;
    g = U(235);
    tuning = U(236:245);
    u_k = [(SB2P*(U(246:248) + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 +
        yhat_k(5)^2)]));U(249:251)];
    n = length(yhat_k);
    % max allowable 2-norm of the error in the filtered state iterator
    epsilon = 1e-10;
    % max number of iterations per timestep
    max_j = 20;

    % HKF specific settings
    tp_Q1 = tuning(1);
    tp_Q2 = tuning(2);
    tp_Qd = tuning(7);
    tp_Qatt = tuning(3);
    tp_R = tuning(4);
    tp_Ra = tuning(5);

    % System noise:
    fm = tp_Q1*[1;1;1];
    tm = tp_Q2*[1;1;1];
    stdw = [fm;tm;tp_Qatt*[1;1];tp_Qd*ones(6,1)];
    Q = diag(stdw.^2);
    % Measurement noise:
    stdv = [tp_R*ones(12,1);tp_Ra*ones(3,1)];
    R = diag(stdv.^2);

    L =  n + length(stdw);       % length of the augmented state vector
    alpha = tuning(8);
    kappa = tuning(10);
    lambda = alpha^2*(L + kappa) - L;
    Ws0 = lambda/(L+lambda);
    Wi = 1/(2*(L+lambda));


    % 1 - Augment the state and covariance
    Xa = [yhat_k;zeros(14,1)];
    Pa = [P_k zeros(14,14);
          zeros(14,14) Q];

    % 2 - prediction sigma points
    chi_p = Xa*ones(1,2*L+1);
%     variation = sqrt(L+lambda)*sqrtm(Pa);  % Gebruikt A = X*X ipv A = X*X'
    variation = sqrt(L+lambda)*chol(Pa')';  % lower triangular Cholesky
    chi_p(:,2:end) = chi_p(:,2:end) + [variation,-variation];

    % 3 - instantiate each sigma point through the equations of motion
    for i = 1:2*L+1
        chi_p(:,i) = RK4(chi_p(:,i),N,dt,params,u_k,[accel_p;u],g);   % ###
    end

    % 4 - produce predicted state and covariance
    x_predict = Ws0*chi_p(1:n,1);
```

```
    for i = 2:2*L+1
        x_predict = x_predict + Wi*chi_p(1:n,i);
    end

%     P_predict = Wc0*(chi_p(1:n,1) - x_predict)*(chi_p(1:n,1) - x_predict)';
%     for i = 2:2*L+1
%         P_predict = P_predict + Wi*(chi_p(1:n,i) - x_predict)*(chi_p(1:n,i) - x_predict)';
%     end

    % Modified covariance matrix algorithm, ensures positive definiteness
    P_predict = zeros(n,n);
    for i = 2:2*L+1
        P_predict = P_predict + Wi*(chi_p(1:n,i) - chi_p(1:n,1))*(chi_p(1:n,i) - chi_p(1:n,1))';
    end

    % Iterate the nominal states
    eta2 = x_predict;
    error = 2*epsilon;      % initiate error with a value larger than epsilon
    j = 0;
    Vdot = zeros(3,1);
    P = zeros(n,n);
    while error > epsilon
        if  j >= max_j
            break
        end
        j = j+1;
        eta1 = eta2;
        H = dhdx(eta1,g,delta);
        % Kalman gain matrix
%         K = P_predict*H'*inv(H*P_predict*H'+R);
        [K,P] = SquareRootKalmanGain(H,P_predict,R);
        % measurement update
        z_kplus1 = [omega;accel];
        Vdot(1) = accel_p(1)-eta1(5)*eta1(3)+eta1(6)*eta1(2)-g*sin(eta1(7));
        Vdot(2) = accel_p(2)-eta1(6)*eta1(1)+eta1(4)*eta1(3)+g*sin(eta1(8))*cos(eta1(7));
        Vdot(3) = accel_p(3)-eta1(4)*eta1(2)+eta1(5)*eta1(1)+g*cos(eta1(8))*cos(eta1(7));
%         Vdot = (eta1(1:3) - yhat_k(1:3))/dt;
        eta2 = x_predict + K*(z_kplus1 - func2(eta1,Vdot,g,delta) - H*(x_predict-eta1));
        error = norm((eta2-eta1),inf)/norm(eta1,inf);
    end
    yhat = eta2;

    sd = sqrt(diag(P));
    F = dfdx(yhat,params);
    H = dhdx(yhat,g,delta);
    % Observability matrix
    W = zeros(15*14,14);W(1:15,:) = H;
    term = H;
    for ii = 1:13
        term = term*F;
        W((1+15*ii):(15 + 15*ii),:) = term;
    end
    sv = svd(W);          % ##
    tol = max(size(W))*eps(max(sv));
    r = sum(sv > tol);
    condition = sv(1)/sv(r);

    % Construct output vector Y
    Y(1:196) = reshape(P,196,1);
    Y(197:210) = yhat;
    Y(211:224) = sd;
    Y(225:238) = sv;
    Y(239) = r;
    Y(240) = condition;
    Y(241) = j;
end
```

D-4 Embedded Matlab UKF Algorithm

```
function Y = UKF(U)

    Y = zeros(242,1);
    % Direction Cosine Matrix from SensorBoard to Pendulum
    SB2P = [-0.607873596459256,-0.655798968567627, 0.447680023625497;
             -0.406013862505740,-0.227812369076838,-0.885016535409802;
              0.682379977857542,-0.719742479864316,-0.127782348146782];
    % Reconstruct inputs P_k, yhat_k, u, omega, params, dt, g
    P_k = reshape(U(1:196),14,14);
    yhat_k = U(197:210);
    u = U(211:213);
    omega = U(214:225);
    accel = U(226:228);
    delta = 0.158;
    accel_p = SB2P*(accel + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 + yhat_k
        (5)^2)]);
    params = U(229:233);
```

```
dt = U(234);
N = round(dt*3000)+2;
g = U(235);
tuning = U(236:245);
u_k = [(SB2P*(U(246:248) + delta*[-yhat_k(4)*yhat_k(6);-yhat_k(5)*yhat_k(6);(yhat_k(4)^2 +
    yhat_k(5)^2)]));U(249:251)];
n = length(yhat_k);

% UKF specific settings
tp_Q1 = tuning(1);
tp_Q2 = tuning(2);
tp_Qd = tuning(7);
tp_Qatt = tuning(3);
tp_R = tuning(4);
tp_Ra = tuning(5);

% System noise:
fm = tp_Q1*[1;1;1];
tm = tp_Q2*[1;1;1];
stdw = [fm;tm;tp_Qatt*[1;1];tp_Qd*ones(6,1)];
Q = diag(stdw.^2);
% Measurement noise:
stdv = [tp_R*ones(12,1);tp_Ra*ones(3,1)];
R = diag(stdv.^2);

L =  n + length(stdw);       % length of the augmented state vector
alpha = tuning(8);
beta = tuning(9);
kappa = tuning(10);
lambda = alpha^2*(L + kappa) - L;
Ws0 = lambda/(L+lambda);
Wc0 = Ws0 + 1 - alpha^2 + beta;
Wi = 1/(2*(L+lambda));

% 1 - Augment the state and covariance
Xa = [yhat_k;zeros(14,1)];
Pa = [P_k zeros(14,14);
      zeros(14,14) Q];

% 2 - prediction sigma points
chi_p = Xa*ones(1,2*L+1);
%    variation = sqrt(L+lambda)*sqrtm(Pa);  % Gebruik A = X*X ipv A = X*X'
variation = sqrt(L+lambda)*chol(Pa')';  % lower triangular Cholesky
chi_p(:,2:end) = chi_p(:,2:end) + [variation,-variation];

% 3 - instantiate each sigma point through the equations of motion
for i = 1:2*L+1
    chi_p(:,i) = RK4(chi_p(:,i),N,dt,params,u_k,[accel_p;u],g);   % ###
end

% 4 - produce predicted state and covariance
x_predict = Ws0*chi_p(1:n,1);
for i = 2:2*L+1
    x_predict = x_predict + Wi*chi_p(1:n,i);
end

%    P_predict = Wc0*(chi_p(1:n,1) - x_predict)*(chi_p(1:n,1) - x_predict)';
%    for i = 2:2*L+1
%        P_predict = P_predict + Wi*(chi_p(1:n,i) - x_predict)*(chi_p(1:n,i) - x_predict)';
%    end

% Modified covariance matrix algorithm, ensures positive definiteness
P_predict = zeros(n,n);
for i = 2:2*L+1
    P_predict = P_predict + Wi*(chi_p(1:n,i) - chi_p(1:n,1))*(chi_p(1:n,i) - chi_p(1:n,1))';
end

%    % 5 - Augment the predicted state and covariance
%    Xa_predict = [x_predict;zeros(15,1)];
%    Pa_predict = [P_predict zeros(14,15);
%                  zeros(15,14) R];
%    L = length(Xa_predict);
%
%    % 6 - Generate update sigma points
%    chi_u = Xa_predict*ones(1,2*L+1);
%    variation = chol((L+lambda)*Pa_predict,'lower');
%    chi_u(:,2:end) = chi_u(:,2:end) + [variation,-variation];
chi_u = chi_p(1:n,:);

% 8 - instantiate each sigma point through the observation function
gamma = zeros(15,2*L+1);
Vdot = zeros(3,1);
for i = 1:2*L+1
    Vdot(1) = accel_p(1)-chi_u(5,i)*chi_u(3,i)+chi_u(6,i)*chi_u(2,i)-g*sin(chi_u(7,i));
    Vdot(2) = accel_p(2)-chi_u(6,i)*chi_u(1,i)+chi_u(4,i)*chi_u(3,i)+g*sin(chi_u(8,i))*cos(chi_u
        (7,i));
    Vdot(3) = accel_p(3)-chi_u(4,i)*chi_u(2,i)+chi_u(5,i)*chi_u(1,i)+g*cos(chi_u(8,i))*cos(chi_u
        (7,i));
    gamma(:,i) = func2(chi_u(:,i),Vdot,g,delta);
end
```

```
    % 9 - produce the predicted measurement and its covariance
z_predict = Ws0*gamma(:,1);
for i = 2:2*L+1
    z_predict = z_predict + Wi*gamma(:,i);
end

P_zz = R;
for i = 2:2*L+1
    P_zz = P_zz + Wi*(gamma(:,i) - gamma(:,1))*(gamma(:,i) - gamma(:,1))';
end

%   P_zz = Wc0*(gamma(:,1) - z_predict)*(gamma(:,1) - z_predict)' + R;
%   for i = 2:2*L+1
%       P_zz = P_zz + Wi*(gamma(:,i) - z_predict)*(gamma(:,i) - z_predict)';
%   end

%   P_xz = zeros(n,15);
%   for i = 2:2*L+1
%       P_xz = P_xz + Wi*(chi_u(1:n,i) - chi_u(1:n,1))*(gamma(:,i) - gamma(:,1))';
%   end

P_xz = Wc0*(chi_u(1:n,1) - x_predict)*(gamma(:,1) - z_predict)';
for i = 2:2*L+1
    P_xz = P_xz + Wi*(chi_u(1:n,i) - x_predict)*(gamma(:,i) - z_predict)';
end

% 10 - Kalman gain
K = P_xz*inv(P_zz);

% 11 - the state update equation
z_kplus1 = [omega;accel];
yhat = x_predict + K*(z_kplus1 - z_predict);

% 12 - updated covariance
P = P_predict - K*P_zz*K';
sd = sqrt(diag(P));
F = dfdx(yhat,params);
H = dhdx(yhat,g,delta);
% Observability matrix
W = zeros(15*14,14);W(1:15,:) = H;
term = H;
for ii = 1:13
    term = term*F;
    W((1+15*ii):(15 + 15*ii),:) = term;
end
sv = svd(W);
tol = max(size(W))*eps(max(sv));
r = sum(sv > tol);
condition = sv(1)/sv(r);

% Construct output vector Y
Y(1:196) = reshape(P,196,1);
Y(197:210) = yhat;
Y(211:224) = sd;
Y(225:238) = sv;
Y(239) = r;
Y(240) = condition;
Y(241) = 1; % j;
end
```