School of Engineering & Design
Electronic & Computer Engineering

MSc Distributed Computing Systems Engineering

Brunel University West London

Simulation and Performance Analysis of a Distributed Position Correction Scheme for Unmanned Aerial Vehicles

Dionysios Satikidis

Supervisor: Dr. Marios Hadjinicolaou

March / 2011

A Dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science

School of Engineering & Design
Electronic & Computer Engineering

MSc Distributed Computing Systems Engineering

Brunel University West London

Simulation and Performance Analysis of a Distributed Position Correction Scheme for Unmanned Aerial Vehicles

Dionysios Satikidis

Signature: _____

Declaration: *I have read and I understand the MSc dissertation guidelines on plagiarism and cheating, and I certify that this submission fully complies with these guidelines.*

Abstract

Abstract summary ...

Acknowledgements

Acknowledgements ...

Contents

List of Abbreviations

I/O	Input/Output
4GL	Fourth Generation Language
ADC	Analog to Digital Converter
ALGO	Algorithm
API	Application Programming Interface(s)
BDM	Background Debug Mode
BMOF	Block Matching Optical Flow
BS	Block Size
CACD	Camera Calibration Domain
CCU	Central Control Unit
CLCS	Closed Loop Control System(s)
COOF	Corner Detection Optical Flow
CT	Cycle Time(s)
CV	Computer Vision
DMS	Deadline Monotonic Scheduling
DOF	Degrees of Freedom
DSM	Domain Specific Modelling
DSP	Digital Signal Processor

ECT	Enhanced Capture Timer
EOM	Equations of Motion
EVTF	Error-Value-Transfer-Function
FNC	Flow Number Calculation
FOE	Focus Of Expansion
FPGA	Field-Programmable Gate Array
FPS	Frames per Second
GAV	Generalized Acceleration Vector
GPS	Global Position System
GPV	Generalized Position Vector
GUAV	Gliding Unmanned Aerial Vehicle(s)
GUI	Graphical User Interface
GVV	Generalized Velocity Vector
HAL	Hardware Abstraction Layer
HighGUI	High Graphical User Interface
HIL	Hardware In the Loop
HSE	University of Applied Sciences Esslingen
HUAV	Hovering Unmanned Aerial Vehicle(s)
HWBIP	Hardware based image processing
IIC	Inter-Integrated Circuit
IMU	Inertial Measurement Unit

IS	Image Size
ISR	Interrupt Service Routine
LSb	Least Significant bit
MATLAB	MATrix LABoratory
MAV	Micro Aerial Vehicle(s)
MBD	Model Based Design
MC	Microcontroller
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical System
MIMO	Multiple-Input and Multiple-Output
MLL	Machine Learning Library
OC	Optical Correction
ODE	Ordinary Differential Equation(s)
OF	Optical Flow
OFCA	Off-board camera
OFIP	Off-board image processing
ONCA	On-board camera
ONIP	On-board image processing
OpenCV	Open Computer Vision
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative

RANSAC	Random Sample Consensus
RF	Radio Frequency
RPM	Revolutions per Minute
SBC	Single Board Computer(s)
SCI	Serial Communication Interface
SISO	Single-Input and Single-Output
SLAM	Simultaneous Localization And Mapping
SOAD	Shift of the Optical Axis Domain
SPI	Serial Peripheral Interface
SVTF	Set-Value-Transfer-Function
SWBIP	Software based image processing
TF	Transfer Function
Tform	Transformation Matrix
TT	Time Triggered
UAV	Unmanned Aerial Vehicle(s)
UND	Underground
WRT	with respect to
XML	Extensible Markup Language

List of Symbols

X_E	X-axis WRT E-frame
Y_E	Y-axis WRT E-frame
Z_E	Z-axis WRT E-frame
ϕ	Angular quadrocopter position around X_E WRT E-frame (Roll)
θ	Angular quadrocopter position around Z_E WRT E-frame (Yaw)
ψ	Angular quadrocopter position around Y_E WRT E-frame (Pitch)
x_e	Linear quadrocopter position along X_E WRT E-frame
y_e	Linear quadrocopter position along Y_E WRT E-frame
z_e	Linear quadrocopter position along Z_E WRT E-frame
$\dot{x}_e$	Linear quadrocopter velocity along X_E WRT E-frame
$\dot{y}_e$	Linear quadrocopter velocity along Y_E WRT E-frame
$\dot{z}_e$	Linear quadrocopter velocity along Z_E WRT E-frame
$\ddot{x}_e$	Linear acceleration of quadrocopter along X_E WRT E-frame
$\ddot{y}_e$	Linear acceleration of quadrocopter along Y_E WRT E-frame
$\ddot{z}_e$	Linear acceleration of quadrocopter along Z_E WRT E-frame
X_B	X-axis WRT B-frame
Y_B	Y-axis WRT B-frame
Z_B	Z-axis WRT B-frame
p	Angular quadrocopter velocity around X_B WRT B-frame (Roll)
q	Angular quadrocopter velocity around Y_B WRT B-frame (Pitch)

r_g	Angular velocity r measured by gyroscope
r_{of}	Angular velocity r measured by optical flow sensor
r	Angular quadrocopter velocity around Z_B WRT B-frame (Yaw)
x_b	Linear quadrocopter position along X_B WRT B-frame
y_b	Linear quadrocopter position along Y_B WRT B-frame
z_b	Linear quadrocopter position along Z_B WRT B-frame
u	Linear quadrocopter velocity along X_B WRT B-frame
v	Linear quadrocopter velocity along Y_B WRT B-frame
w	Linear quadrocopter velocity along Z_B WRT B-frame
$\dot{u}$	Linear quadrocopter acceleration along X_B WRT B-frame
$\dot{v}$	Linear quadrocopter acceleration along Y_B WRT B-frame
$\dot{w}$	Linear quadrocopter acceleration along Z_B WRT B-frame
ω	Speed of rotation
$\Delta\omega$	Deviation of rotation speed
ξ	GPV WRT E-frame
$\dot{\xi}$	GVV WRT the E-frame
ν	GVV WRT the B-frame
J_Θ	Transformation matrix from B-frame to E-frame
R_Θ	Rotation transformation matrix from B-frame to E-frame
T_Θ	Translation transformation matrix from B-frame to E-frame
Γ^E	Linear position vector WRT the E-frame
Θ^E	Angular position vector WRT the E-frame

V^B	Linear velocity vector WRT the B-frame
ω^B	Angular velocity vector WRT the B-frame
$\ddot{\Gamma}^E$	Linear acceleration vector WRT the E-frame
$\dot{\Gamma}^E$	Linear velocity vector WRT the E-frame
M_B	System inertia matrix WRT the B-frame
M_H	System inertia matrix WRT the H-frame
$C_B(\nu)$	Coriolis-centripetal matrix WRT the B-frame
$C_H(\zeta)$	Coriolis-centripetal matrix WRT the H-frame
m	Mass of body
I_{nxn}	Identity matrix of dimension n times n
I	Body inertia matrix
$G_B(\xi)$	Gravitational vector WRT the B-frame
G_H	Gravitational vector WRT the H-frame
$O_B(\nu)$	Gyroscopic propeller matrix WRT B-frame
$O_H(\zeta)$	Gyroscopic propeller matrix WRT H-frame
Ω	Sum of propellers' speed
E_B	Movement matrix WRT B-frame
$E_H(\xi)$	Movement matrix WRT H-frame
b	Factor of thrust
d	Factor of drag
l	Distance between centre of quadrocopter and centre of propeller
$\dot{\nu}$	GAV WRT B-frame

ζ	GVV WRT H-frame
$\dot{\zeta}$	GAV WRT H-frame
F^B	Forces vector of quadrocopter WRT B-frame
τ^B	Torques vector of quadrocopter WRT B-frame
$\dot{V}^B$	Linear acceleration vector WRT the B-frame
$\dot{\omega}^B$	Angular acceleration vector WRT the B-frame
U_1	Vertical thrust WRT the B-frame
U_2	Roll torque WRT the B-frame
U_3	Pitch torque WRT the B-frame
U_4	Yaw torque WRT the B-frame
$F_{n\{X_B,Y_B,Z_B\}}$	Thrust component of rotor n along X_B, Y_B or Z_B
S^2_{N-1}	Bias-corrected Sample Variance(s)

List of Figures

1 Introduction

During the last years the interest of robotic science and the development of robots increase enormous. Reasons for that are, the unstoppable technological progress of hard and software techniques, but also the necessity to replace humans with machines in dangerous, monotonous or unreachable industrial environments (medical, space, aviation and so far). One area of these interests is the aerial platform and the realization of Unmanned Aerial Vehicle(s) (UAV), which are mostly controlled via remote control or fly autonomous. These aircraft vehicles have several capabilities like in military or rescue operations with special environments like a burning house. For such indoor operation it is important to focus, that some kind of feedback sensors like GPS-sensors could not work satisfactory. In such cases UAV s faces problems with the self stabilization, because their physical behaviour is in generally unstable [19, p.1 Introduction]. Most of the approaches to stabilize UAV work with a clever combination of sensor equipment and control algorithms. Mostly this controller uses a Inertial Measurement Unit (IMU) which is mounted on board and includes accelerations sensors to detect movements in the given Degrees of Freedom (DOF). Actual acceleration sensors, which are used for this purpose, are Micro-Electro-Mechanical System (MEMS) or fiberoptic sensors which have a finite precision and unacceptable error propagation in case of integration for velocity or position detection [18, pp.11-13 Function Principles of MEMS, Sources of Error]. This problem also is a field of research at the Quadrocopter project of the University of Applied Sciences Esslingen (HSE) [HSE10, Website]. The goal of this Master's Thesis is to research the topic of flight stabilization and error drift elimination with a optical sensor. This approach has to

be tested and evaluated with the implementation of a simulation which focus the behaviour of the flight dynamics in relation to the quality of the optical measurements and the control system of the quadrocopter.

Figure 1.1: HSE Quadrocopter

1.1 Context of Project

The quadrocopter project of the HSE was launched with the goal to build a system from the scratch, which is developed by students, scientific workers and Professors. In a first project the Printed Circuit Board which hosts the parts necessary to control the quadrocopter was designed by students at the faculty of Mechatronics and Electrical Engineering in Goeppingen. This project groups' focus was then

on the simulation, implementation of the basic functions and visualization of the actual condition of the aircraft. The first two project groups already show the benefit of this project, a multidisciplinary development including hardware design, application development, embedded programming, simulation and the interfaces between these special fields. In the year 2009, the Faculty of Information technology adopted the development of the project with the aim to solve problems which came up in the previous development and to redesign the soft- and hardware architecture. So a new hardware design was developed in a corporation between the two faculties with the outcome of a Central Control Unit (CCU) which can detect inertial movements in six DOF and can control the four actuators of the quadrocopter via so called brushless controllers. One of the biggest unresolved problems since yet is the development of a robust controller and the elimination of drifts which especially come up at the hovering state. The practice and the experience of previous developments show, that it is indispensable to proof new developments with a simulation before they will be realized at the real UAV. So the main topic and the focus of this Master's Thesis will be on the development of a simulation which shows potential solutions of the mentioned problems and the research and evaluation of the outcome results.

1.2 Problem Description

Improvements of high density power storage, integrated miniature actuators and sensors, facilitate the development of Micro Aerial Vehicle(s) (MAV) and new areas of research for unmanned and autonomous flying systems [Sam04, p.1 Introduction]. This new area of interest brought also a new area of problems. One of these is the fact that the pilot of the aerial vehicle does not exist, because the UAV

flights autonomous or the pilot observes and controls the UAV via remote control. In both cases it is necessary that the UAV system can detect its absolute position, to provide the pilot a better quality of control or furthermore to manoeuvre autonomous. The following articles also describe this problem with different views and approaches [Sla09, p.2 Localization and path planning] [Kar10, p.1 High precision aircraft positioning] [Seb08, p.2 Teleoperated Robot Control].

This necessity of location determination requires measurement unit, which provides the detection of the movements in the given DOF. Because of low-cost and low-complexity reasons, the most popular components which are used to reach a nearly satisfactorily level of flight stabilization are inertial sensors. These sensors combined together called IMU and have the ability to detect the acceleration of translational or rotational movements. Furthermore a control system can be used in a closed loop together with the IMU and can correct theoretically nearly every disturbance in a continual system [26, pp.45-64 System Control] [Ped05, pp.49-51 Control Strategy]. The limitation of resolution of the sensors and the necessity of integrating the acceleration values for position and velocity determination, have in real systems a big impact in aspects of error propagation and detection of smoothly movements [5, p.5 Low-acceleration Drift]. The quadrocopter of the HSE also includes a IMU for stabilization and faces the same problems. These problems can be eliminated with the extension a of optical system which can track the absolute position or of the ego-motion UAV.

1.3 Approach

1.4 Aims and Objectives

The aim of this dissertation is to provide a simulation architecture that can be used as prototype development platform for a distributed visual movement detection and control of a quadrocopter. Furthermore the characteristics of the distributed image processing and movement detection have to be analysed in relation to the variation of configurations and critically assessed.

One essential objective is that the configuration of the simulating components provides the option to simulate a range of hardware components which are not purchased until now. By way of example the simulation of the on-Board camera has to provide options of configuration for the image size, underground structure and so far. The simulation of the communication between UAV and host also has to provide a variation of transmission rate and further behaviour which could affect the visual movement detection at the base station. The efficiency in relation with the quality of function is an important indicator for the success and acceptance of the distributed movement detection approach. So it is important to get an insight to the possible characteristics of the simulated components with the result to find a way which satisfies the efficiency and quality aspects.

Another important objective is that the interfaces between the simulating components are clearly specified and allow a way of modular exchangeability of simulation components with the real objects. This purpose has to allow a more precise investigation of the behaviour of the real hardware related components and the option to test software for the UAV target, like the On-Board control algorithm,

at the base station.

The realization of the simulation therefore has to provide an encapsulated and flexible architecture and has to simulate behaviour like delays and jitters for simulated components. Thereby the simulation has to adjust a real-time-behaviour in the complete simulated environment and to allow so measurements and prediction of feasibilities with the simulated configuration.

2 Methodology and Project Organisation

2.1 Software Development Process

2.1.1 Model Based Design

Model Based Design (MBD) is a popular method to encounter problems which come up in the development of mechatronic products. These systems almost involve mechanical, electrical, control and embedded components which are developed in different teams of engineers with a specialized focus on the part of the complete project. Some of these components are related on the results of other components before they can be developed. This problem can be solved with MBD and the advantage to develop modules of the complete project by simulating their environment. So the development can run highly parallel with the benefit that the modules can be continuous tested in each phase of the project [10, pp.1-2 Challenges of mechatronic product development]. Beside the direct challenges of mechatronic projects, also problems exists with the multiple domain modelling and development notations and languages. Legacy approaches almost uses several solutions of domains and mapped these from the idea domain into the solution domain. The problem thereby is the additional effort to spit up the problem in domain terms and to map the outcomes together. So another concept which is adopted by modern MBD architectures is to create in a Domain Specific

Modelling (DSM) language a model which is generated to code form the domain specific framework and to use the benefit to model the functionality and not indirect the code(See figure 2.1[1])[30, pp.55-60 Chapter 3.3 Difference from other modelling approaches]. One of the biggest disadvantages of MBD architectures is that the DSM language is not as fine grained and low-level as classical programming languages. This can causes problems especially in projects with very special requirements to the functionality which cannot be reflected without a workaround into the DSM language.

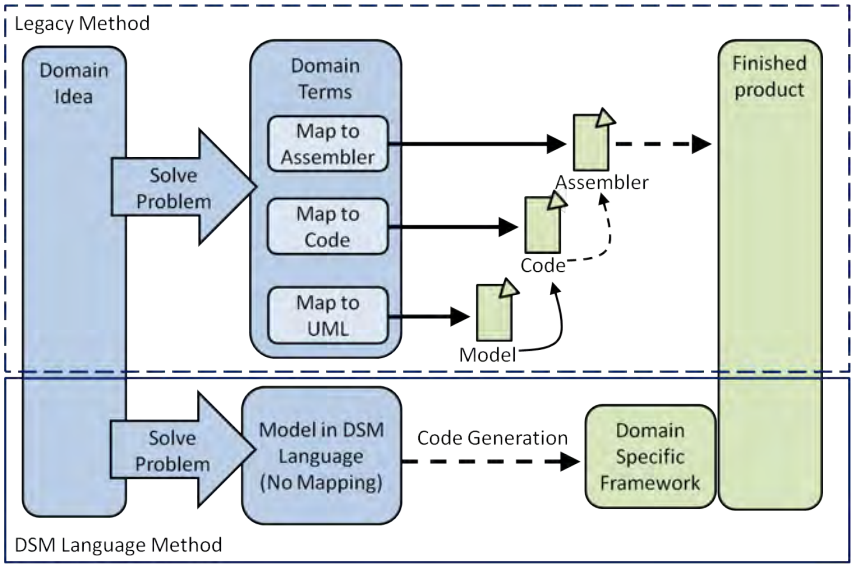


Figure 2.1: Legacy DSM vs. DSM Language approach

The development section in figure **??** includes the phases and the corresponding key capabilities of MBD. The first phase of MBD is the realization of the researched and required components into a simulation environment. Thereby physical components, environmental model and algorithms are abstracted to systems by using domain-specific modelling tools with a well defined edges and intercommunication. The developed systems in the design phase can be tested simultaneously to analyse the system performance and correctness. Other key

[1]This figure is derived from [29, p.13 Modelling functionality vs. modelling code]

capabilities of MBD are given in the implementation phase. Mostly MBD tools allows to generate embedded code from the designed systems or to combine handwritten code with the build simulation of the design phase. So the implemented modules can be similarly tested in the adopted simulation environment. Finally components which have passed the tests at the implementation phase can be integrated together.

Ultimately the final product can also be tested with the MBD tool by simulating the environment of the product e.g. in a Hardware In the Loop (HIL) test bench. [2, Model Based Design, MathWorks]

2.1.2 Spiral Model

The most of the UAV stability and movement detection approaches were developed iterative under consideration of the upcoming problems and obstacles [11, Erd03, Iterative Development, Single and Dual Camera Feedback] [31, 32, Iterative Development, Landing and Position Control Development]. This procedure model also will be appropriate to this project, because the potential risks are difficult to identify. So the outcome of the development process have to be a prototype which can be evaluated, tested and extended. A process model, which provides an appropriate structure to face the iterative prototyping strategy under the consideration of the risk aspects, is given with the spiral model. The classical spiral model has typically four phases in which the product is developed in an incremental evolutionary process. Derivates of this classical model which focus the customer evaluation for quality improvements may three, five or six phases. In the context of this project, the classical four phases spiral model is used for scientific and feasibility study and does not have to provide further customer communication phases [24, pp.36-38 The Spiral Model].

A typical cycle of the four phases spiral model (2.2) begins with the identification of the objectives which have to be elaborated like performance, functionality, flexibility and so far. The next step is to evaluate the alternatives relative to the objectives and constraints and to determine significant sources of risks. After that, the next level iteration of the product is planned. In the special case of

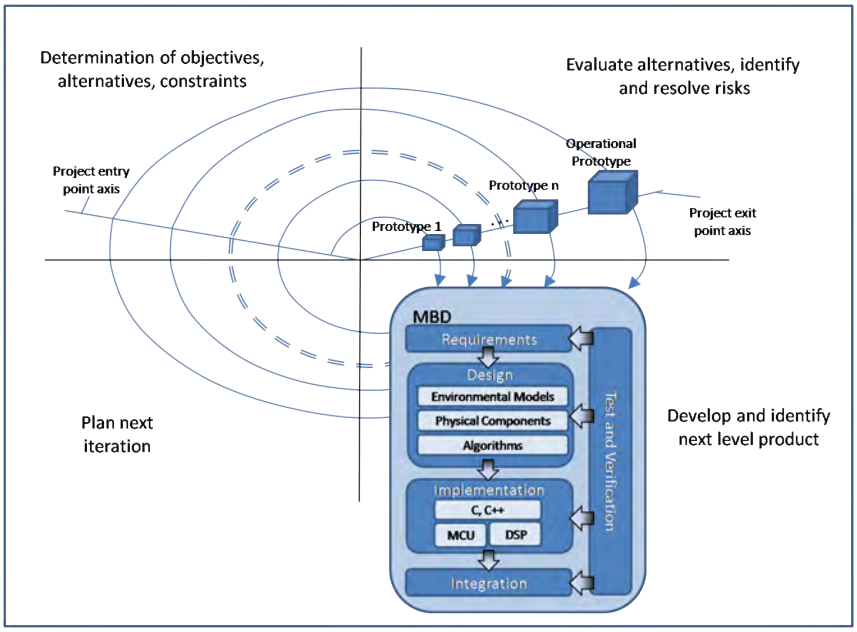


Figure 2.2: The Spiral Model in combination with Model Based Design

this project it is comfortable to use MBD in this phase by using the results of the previous phase as input. This input can be a planed prototype or requirements which describe the changes to execute. The output of the MBD phase can be used again in the planning phase for the next iteration [4, pp.64-69 Spiral Model of the Software Process].

2.2 Tools and Architectures

2.2.1 MATLAB/Simulink

The software package MATrix LABoratory (MATLAB)[2], developed by the company MathWorks footnoteSee [Mat11b, The MathWorks, Inc.], stands for solutions of high-performance numerical computation and visualization. Thereby

[2]See [Mat11a, MATLAB]

MATLAB's main features and capabilities can be summarized in the groups, visualized in figure 2.3 [3]. MATLAB's build-in functions provide tools for linear algebra computations, data analysis, signal processing, optimization, numerical solution of Ordinary Differential Equation(s) (ODE) and further types of scientific computation. For visualization, MATLAB supports a numerous of graphical functions for 2-D and 3-D graphics and animations. Modules which are programmed in C/C++ or Fortran can be integrated to MATLAB by using the external interface using the Mex-files. This facility allows to simulate the environment of an embedded algorithm and to test these, to include legacy modules of previous developments or libraries which are programmed in the supported languages. MATLAB also provides a own Fourth Generation Language (4GL)(also denoted as MATLAB) DSM language for programming which focus the fast development of functions or applications. Various toolboxes of MATLAB allow the development of special applications such as symbolic computations, image processing, statistics etc. and reflect the idea of DSM. The list of toolboxes keeps growing over time and releases of MATLAB [21, p.3 What is MATLAB?]. Beside the programming interface of MATLAB, Simulink provides a environment for multi-domain simulation and MBD of dynamic embedded systems. This graphical interface also supports the access to the complete MATLAB continuum of tools [4].

[3] This figure is derived from the schematic diagram presented in [21, A schematic diagram of MATLAB'S main features]

[4] See [Sim11, Simulink]

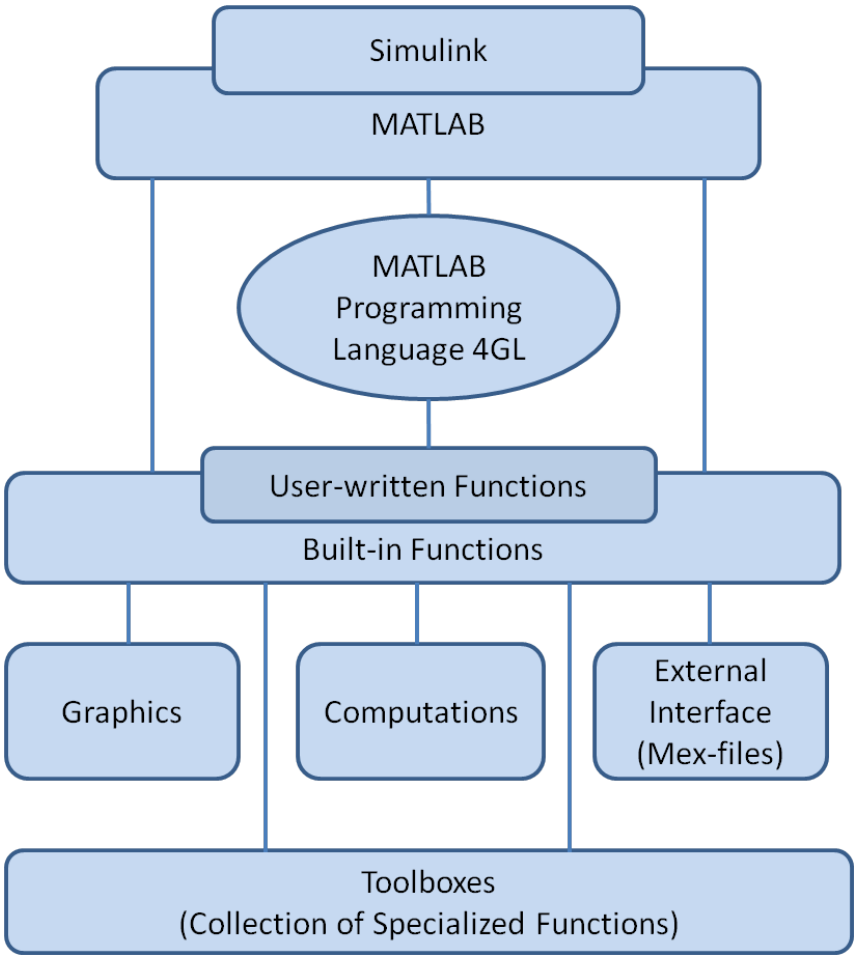


Figure 2.3: MATLAB/Simulink's schematic diagram of main features

2.2.2 OpenCV

Open Computer Vision (OpenCV)[5] is one of the most popular open source computer vision libraries for scientific and industrial projects. It contains over 500 functions that span many areas in vision such as medical imaging, factory product inspection, security, camera calibration, stereo vision, robotics and so far. The main components of OpenCV are visualized in figure 2.4 [6]. The component Computer Vision (CV) contains the basic image processing and higher-level computer vision algorithms. Another important part which is strongly related with computer vision is the machine learning ability. So OpenCV provides statistical classifier and clustering tools which allow this ability in the Machine Learning Library (MLL) component. OpenCV's Input/Output (I/O) component High Graphical User Interface (HighGUI) contains functions and routines for performant loading and storing of image or video data and a Graphical User Interface (GUI). Finally the core data structures and algorithms are located in the component CXCORE which supports data transforms, matrix algebra, object persistence, memory management, error handling and further core capabilities [12, pp.1-33 Chapter 1, Overview].

2.3 Strengths and Risks

2.4 Project Management

[5] See [Ope10]

[6] This basic structure overview bases on the figure presented in [12, p.33 Chapter 1, The basic structure of OpenCV]

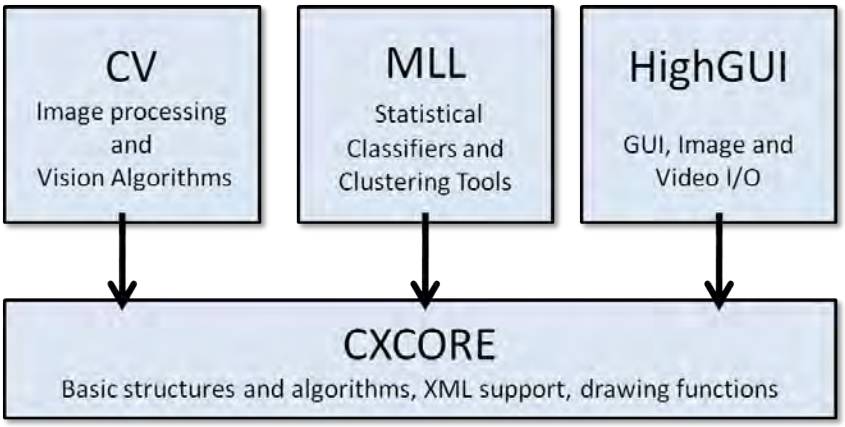


Figure 2.4: OpenCV's basic structure

3 Literature Review

The realization of an optical distributed error correction scheme for UAV contains several challenges of different topics of technical information technology, which have to been introduced for this project in the context of a literature survey. These topics have to been with the focus to create a simulation. That means the theory of function and the complexity of these topics have to be examined and possibilities of simplification have to be given.

The first part of the survey will introduce the basic quadrocopter flight dynamics and approaches to derive the mathematical model of motion. The result of this model will be the Equations of Motion (EOM) in a form which can be used for simulation and reflects as much as possible the reality.

A comparison of the researched existing visual approaches for error correction of UAV will be the focus of the second part of this survey. Thereby the strengths and weaknesses in focus of different aspects have to be given. Furthermore this part will introduce optical movement detection technique which will be investigated here under consideration of the limitations and the abstraction of the reality.

Finally the third part of the survey will give an overview of control approaches in which the determined movement detection value can be processed for aircraft stabilisation. Further methods will be introduced which allow to measure and classify the performance of a control system and its characteristics under consideration of digitalisation aspects.

3.1 Quadrocopter Flight Dynamics

3.1.1 Principles

For a better appreciation of the quadrocopter flight dynamics, the following figure[1] visualizes the influence of the thrust in relation to the movements in the DOF. Thereby the values of ω [rad/sec]represents the least needed speed of rotation, for creating the required thrust for the hovering state. This state can be described as a state, in which forces in the x- and y-axis equals zero and the uplift force in direction of the z-axis has the same absolute value as the gravitational force. The value of $\Delta\omega$ characterizes the purposed deviation of the required rotation speed in hovering state and is used for navigation of the quadrocopter.

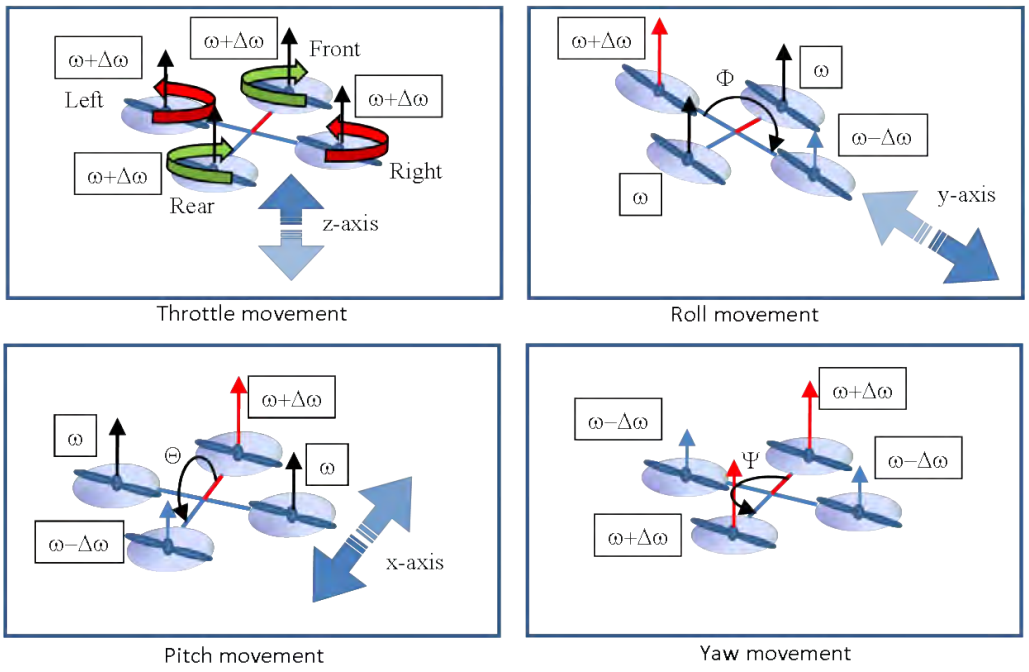


Figure 3.1: Degrees of freedom of a quadrocopter

To keep the equilibrium of rotational kinematics and to prevent self-rotation, the

[1]This picture extends the pictures presented in[33, pp.8-11 Basic concepts]

motor direction of rotation equals crosswise. The quadrocopter has six DOF which can be distinguished as angular and translational movements. Translational movements can be executed in x-, y- and z- axis. Accelerating all rotors with the same speed ω to the value of $\Delta\omega$ will affect a throttle movement U_1 [N] in z-direction. Movements to the negative direction of the z-axis are possible, if the thrust of the four rotors is smaller as the gravitational force of the aerial vehicle. The roll movement U_2 [N m] can be described as a change of the angle around the x-axis. Thereby the left and right rotors execute a force difference by slowing down the one and simultaneous increasing the other speed with $\Delta\omega$. Related to the thrust difference and angular movement, the aerial vehicle creates a force to the y-axis. Equivalent to roll, the pitch movement U_3 [N m] is executed with a change of the angle around the y-axis. Also the pitch movement creates a translational movement across the x-axis. Pitch and roll can only reach a stable angular state and accelerate to the x- or y-axis, if the value of $\Delta\omega$ is the same at the diagonal rotors. Otherwise, the quadrocopter would pure rotate across the corresponding axis. The yaw movement U_4 [N m] is a rotation around the z-axis. This angular movement results in combination of pairwise different thrusts and takes as long as these thrusts are different [2].

3.1.2 Equations of Motion

Simulations of dynamic systems are based on physical models which describe their motion. So the behaviour of the quadrocopter also can be described by EOM, with respect to (WRT) the input parameters of the model. Thanks to these equations, it is viable to predict and define the positions, velocities and acceler-

[2] The theory and the identifiers of this chapter bases on [33, pp.8-11 Quadrotor model and system]

ations of the quadrocopter by investigating the four motor speeds. Bouabdallah [26, pp.15-24 System Modelling] demonstrates two methodologies to derive the EOM with the Newton-Euler and Euler-Lagrange formalism. The Euler-Lagrange formalism derives the EOM by calculating the difference of the kinetic and potential energy of the system under consideration of the general coordinates and forces [3]. The Newton-Euler formalism bases on the Newton-Euler equations [25, p.106 The original recursive Newton-Euler equations] which describe the combined translational and rotational dynamics of a rigid-body with respect to the centre of mass. Both formalisms follow different approaches to derive the EOM, but are based on the same coordinate systems. These two coordinate systems have different origins and describe movements from different perspectives. One coordinate system can be considered as observer perspective to the quadrocopter because it describes the absolute position in space and is called earth inertial frame (E-frame). In contrast to that, the second coordinate system is a body fixed frame (B-frame) and is defined as system of relative movements which originates in the middle of the symmetric rigid quadrocopter body [4]. Bresciani [33, pp.8-23 Quadrotor model and system] shows in his work the identification and derivation of the EOM by using the Euler-Newton formalism and gives reasons why EOM are more conveniently formulated in the B-frame or further in a mixture called hybrid frame (H-frame) [5]. A part of these congenial reasons focus the simplicity to convert on-board measurements to the B-frame coordinates and the matter of fact that the control forces almost are given in B-frame. The other part describes simplifications in the mathematical derivation causes of B-frames symmetrical behaviour and time invariance of the inertias [6].

[3] See [9, pp. 218-219 The Lagrangian Method]

[4] An origin which is coincidence to the centre of mass simplifies the Newton-Euler equation enormous, See [25, p.98 The original recursive Newton-Euler equations]

[5] See [33, p.19 Hybrid frame]

[6] See [33, p.12 Conveniently formulated EOM]

In the following figures[7], we can see the both coordinate systems and the movements in the DOF which are interesting.

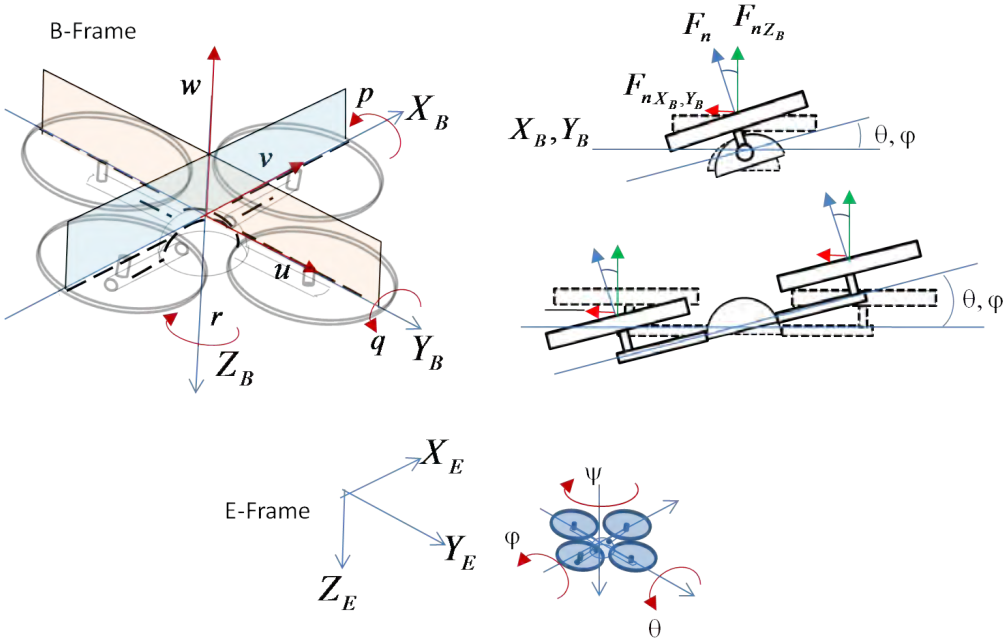


Figure 3.2: Quadrocopter Movements and Coordinate Systems

The starting point of Brescianis derivation is to define an equation, which describe a generic 6 DOF rigid-body (See 3.1) with the Generalized Velocity Vector (GVV) $\dot{\xi}$ WRT the E-frame, the GVV ν WRT the B-frame and the generalized matrix J_Θ which includes the rotation and the translation submatrix R_Θ, T_Θ which transforms ν to $\dot{\xi}$ (See 3.3). The definition of the GVV and the Generalized Position Vector (GPV) WRT the corresponding frames are visualized in figure 3.2 and can be described with the equations 3.4 and 3.2. Thereby ξ is composed of the linear position vector Γ^E [m] and the angular position vector Θ^E [rad]. Similar to the position vector, the velocity vector is composed of a linear velocity vector V^B [m/s] and an angular velocity vector ω^B [rad/sec]. The essence frame of interest is a mixture of the both mentioned frames, called H-frame, which contains the linear velocity vector $\dot{\Gamma}^E$ [m/s] from E-frame and the angular velocity vector

[7] This picture extends the coordinate system picture presented in[26, p.20 OS4 coordinate system]

ω^B [rad/s] from B-frame 3.5. Brescianis interest was to define a frame which can be transferred to an acceleration vector WRT the H-frame and to facilitate the EOM under consideration of typical IMU values of a quadrocopter.

Bresciani uses the Euler-Newton equations (See 3.6)[8] to derive the matrix form which is composed of the system inertia matrix M_B and a Coriolis-centripetal matrix $C_B(\nu)$. A 6 DOF rigid-body dynamics system inertia matrix takes the mass of the body m [9] [kg] and its inertia I [Nms^2] into account.

Furthermore Bresciani recognized that the quadrocopter dynamics can be divided to three contributors which describe the gravitational vector $G_B(\xi)$ [10], the gyroscopic torque $O_B(\nu)$*Ω[11] and the movement vector with the inputs of the system $E_B*\Omega^2$(See 3.6) [12]. So it is possible to create a relation between the Newton-Euler equations (See 3.6) and the sum of these three contributors (3.6), to isolate the Generalized Acceleration Vector (GAV) $\dot{\nu}$, and to derive the EOM WRT the B-frame (See 3.8). Equivalent to that, the substitution of GAV $\dot{\nu}$ with GAV $\dot{\zeta}$ WRT the H-frame in 3.7, gives an equation which can be solved to $\dot{\zeta}$ for the determination of the required EOM and a quadrocopter model with typical IMU values (See 3.9).

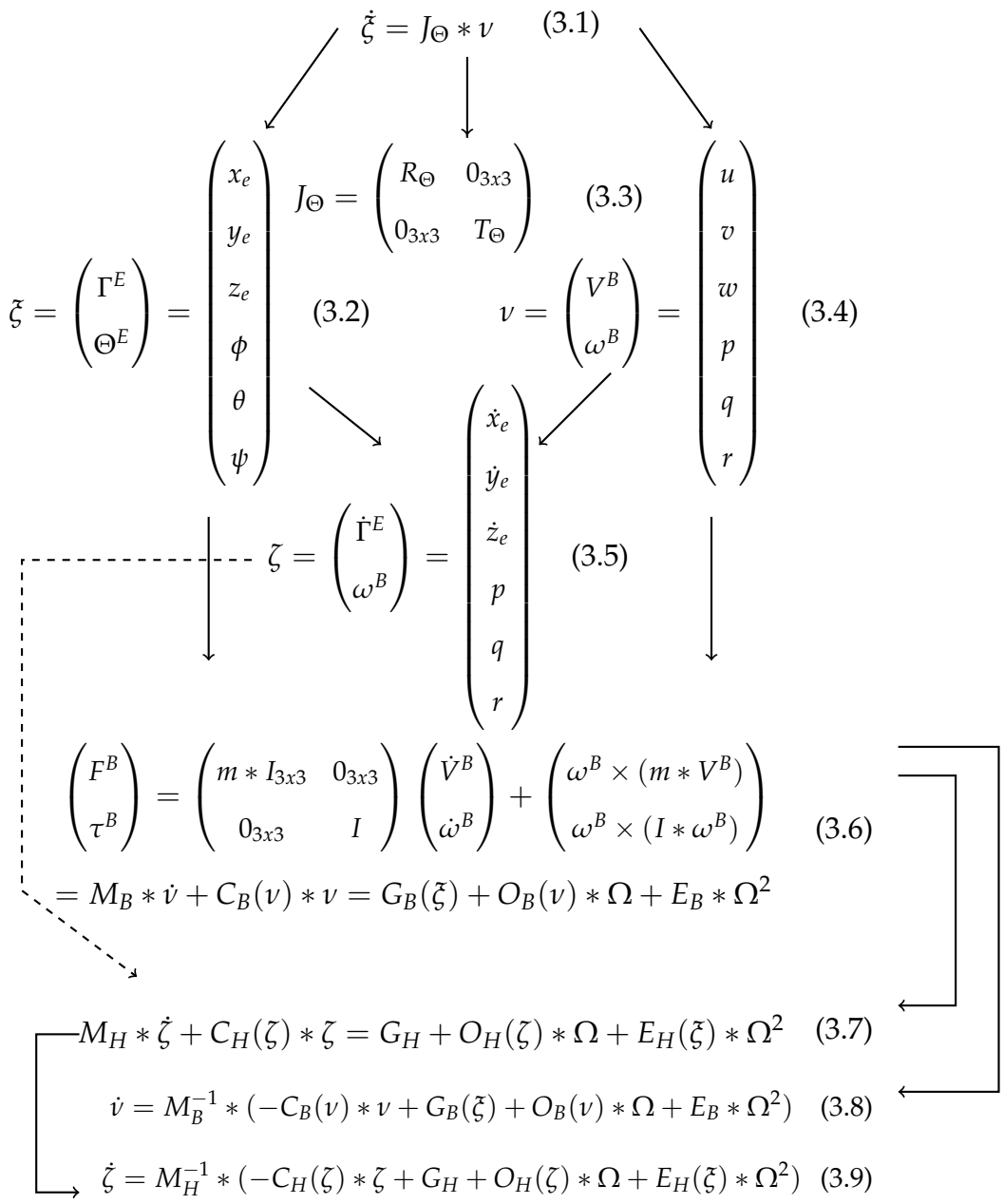
[8] See [33, p.13 The dynamics of a generic 6 DOF rigid-body]

[9] The notation I_{3x3} means 3 times 3 identity matrix (See I_{nxn})

[10] $G_B(\xi)$ considers the acceleration of gravity g [m/s^2] and influences the linear forces, See [33, p.15 The gravitational vector]

[11] $O_B(\nu)$*Ω considers the gyroscopic effects of the propeller rotation which influence the angular forces. Thereby $O_B(\nu)$ is the propeller matrix and Ω [rad/s] the proppelers' speed vector See [33, p.16 The gyroscopic torque]

[12] The constant matrix E_B 3.11 includes the thrust b [Ns^2], d[Nms^2]and ll [m] (distance between the center of the quadrocopter and the and the center of a propeller) factors of the input system. The inputs are given with Ω^2 because the forces and torques are proportional to the squared propellers' speed, See [33, pp.129-135 Aerodynamics calculation]



The following equations show the derived EOM of the GAV $\dot{\zeta}$ WRT the H-frame. Apparent from the rotational DOF of the quadrocopter, the linear accelerations $\dot{\Gamma}^E$ WRT the E-frame are influenced by a trigonometrical equation of the angles Θ^E and the sum of the uplift forces U_1. This behaviour also is vi-

sualized in the cross-sectional and front side view WRT the specific angle and B-frame axis in the figure 3.2. Further the equations show the angular accelerations $\dot{\omega}^B$ WRT the B-frame and the corresponding influences which derive from the propellers' force constellation U_1, U_2, U_3, U_4(See chapter 3.1) the angular velocities from ω^B and the body inertias WRT the mass axis [13] [14] [15] [16].

$$\dot{\zeta} = \begin{pmatrix} \ddot{x}_e \\ \ddot{y}_e \\ \ddot{z}_e \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} (sin\psi sin\phi + cos\psi sin\theta cos\phi)\frac{U_1}{m} \\ (-cos\psi sin\phi + sin\psi sin\theta cos\phi)\frac{U_1}{m} \\ -g + (cos\theta cos\phi)\frac{U_1}{m} \\ \frac{I_{YY}-I_{ZZ}}{I_{XX}} qr - \frac{J_{TP}}{I_{XX}} q\Omega + \frac{U_2}{I_{XX}} \\ \frac{I_{ZZ}-I_{XX}}{I_{YY}} pr + \frac{J_{TP}}{I_{YY}} p\Omega + \frac{U_3}{I_{YY}} \\ \frac{I_{XX}-I_{YY}}{I_{ZZ}} pq + \frac{U_4}{I_{ZZ}} \end{pmatrix} \tag{3.10}$$

$$U_B = E_B \Omega^2 = \begin{pmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ lb(-\Omega_2^2 + \Omega_4^2) \\ lb(-\Omega_1^2 + \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{pmatrix} \tag{3.11}$$

[13] The formula 3.10 bases on [33, p.21 The GAV WRT H-frame]

[14] J_{TP} [N m s2] is the total rotational moment of inertia around the propeller axis, See [33, p.16 Propeller inertia]

[15] I_{XX}, I_{YY}, I_{ZZ} [kg m2] are moments of inertias around the specific axis, See [33, p.136 Inertia matrix]

[16] The identifiers of this chapter are derived from [33]

3.2 Vision Sensors

3.2.1 Motivation

An enormous quantity of researches, which is sponsored by industry companies and universities, was executed to find a better approach to stabilize UAV by using different sensors. Movement detection approaches, which are ultrasonic, sonar or Radio Frequency (RF) based show that it is necessary to have known reference points to get a reliable result [Jue09, p.2 Ultrasound indoor localization with reference points][20, pp.4-5 Radio Model Localization]. The problem of this approach is that the environment has to be prepared before the UAV flight. This preparation is a drawback in point of flexibility in different operation places. Anyway, approaches with ultrasonic, RF or sonar sensors show that the localization of UAV needs a kind of global feedback to correct the UAV absolute position. One of the first motivations for a vision based sensor was presented by Ettlinger et al. [27, pp.1-2 Visual-Based Localization and Control]. In this paper, the authors suggest, that vision is the only practical solution for obstacles of reference free flight stability and showed an On-board approach for a Gliding Unmanned Aerial Vehicle(s) (GUAV) by detecting the horizon with a forward looking camera and estimating and control the flight attitude[5, p.27 Vision Sensors]. The intension of a reference free vision based stability approach for solving the IMU error drift is the motivation for the investigation of visual approaches which will be introduced and compared in this chapter.

3.2.2 Distribution and implementation approaches

This chapter focus the comparison of different approaches of vision sensors' implementation and distribution, which were researched in several scientific works. These approaches are separated in distribution approaches (Off-board image processing (OFIP), On-board image processing (ONIP), On-board camera (ONCA), Off-board camera (OFCA)) which are visualized in figure 3.3, and the implementation approaches Hardware based image processing (HWBIP) and Software based image processing (SWBIP). Problems, such as a limited power resource, a poor level of algorithm complexity for ONIP resulting from the limited calculating On-Board performance and the endeavour to economise weight, lead to outsourcing the image processing to a remote system via a wireless communication, which is not concerned to the On-Board problems. A drawback of this approach was determined by Langer et al. [31, pp.5-7 Off-Board Image Processing] which use OFIP to track a landing pad for autonomous landing and show that the wireless transmission delay has a impact on the sampling rate of the algorithm. Tippetts [5, p.27 Vision Sensors] mentions in his work the limitation of the wireless communication in OFIP as a drawback for range of the aircraft [17]. The OFCA approach was researched by Altug et al. [11, p.76 Localization and Control with an Off-Board camera], with the result of a less sensitive feature detection and position localization as the ONCA approach. OFCA tracking is also shown in the developments of extremely reliable and precise localization of a UAV and is used in the development of aggressive autonomous flights of multiple MAV in the experiments of Mellinger et al. [8, Trajectory Generation and Control with Off-Board cameras] [Dan07, pp.363-364]. The advantage of OFCA tracking system is that the image capturing and position tracking is executed outside the UAV and

[17] The comparison of OFIP and ONIP approaches is visualized in 3.4

prevents complex On-Board calculations for movement and position estimation [18]. The disadvantage of this OFCA method is that they need a special flying environment, which is build up with an OFCA motion tracking system[19] [Nat10, pp.1-2 The GRASP Testbed].

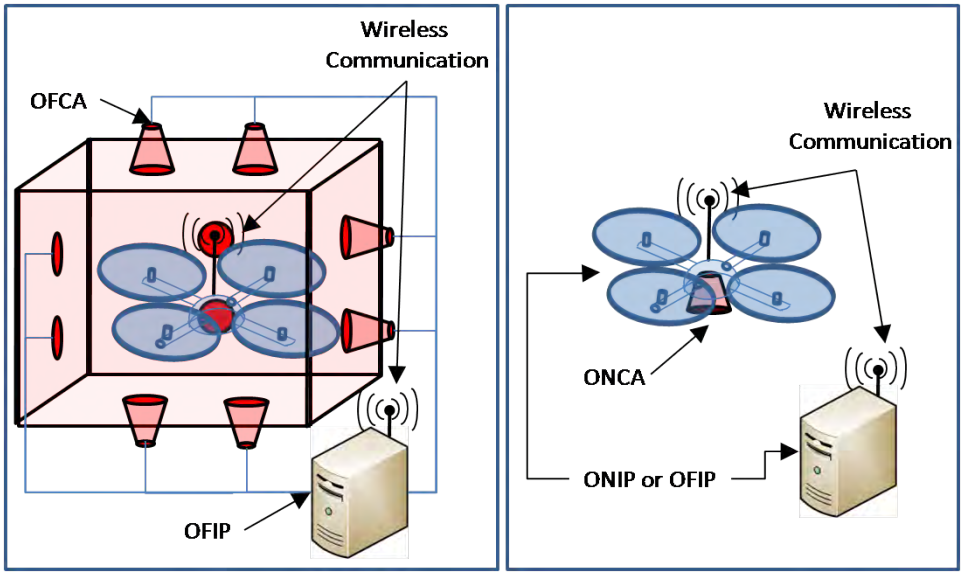


Figure 3.3: On- and Off-Board camera and motion tracking system approach

Tippetts [5, pp.21-22 On-Board image processing with an FPGA] realized an ONIP approach with a Field-Programmable Gate Array (FPGA) which uses a complex feature tracking algorithm but runs with high sampling rate [20]. The characteristics of the feature tracking with a FPGA can result a fast movement tracking method, but it is not an efficient method in the behaviour of power consumption because the hardware is not optimized for the image processing tasks. In contrast to the drawbacks of FPGA, Langer et al. [32, On-Board image processing with mice sensors] and Beyeler et al. [3, pp.4-5] showed an approach for detecting the spatial movement of a UAV with optical mice sensors which have an optimized hardware for image processing and are lightweight. These sensors

[18] This approach focus the E-frame localisation and is robust against position drifts and errors
[19] The comparison of OFCA and ONCA approaches is visualized in 3.4
[20] In this case a high sampling rate means that the image processing runs nearly equal to the sampling rate of the IMU

calculate the optical flow of the captured images and estimate the movement direction of the UAV in hardware and can provide a high sample rate. In contrast to the fast movement detection, a disadvantage is that these sensors have limitations related to the operating environments. These limitations are the concrete light and distance range which is required from the manufacturer [ST 05, pp.7-15 Limitations of the operating environment][32, p.6 Performance of the optical flow based position controller]

SWBIP approaches have a more flexible extension and change behaviour for prototyping of vision based solutions, but they don't work as fast as HWBIP approaches. Stowers et al. [17]realized a heading estimation for a quadrocopter with an onboard Single Board Computer(s) (SBC) which runs the open source computer vision toolkit OpenCV [Ope10]. This software based image processing approach shows strengths in the modularity of the image processing architecture and in the interchangeability of the vision system[21] [17, pp.1-6 Software Based Vision Processing][12, Software Structure and Portability]. Figure 3.4 includes the summary of the examined approaches of this chapter. We can see that no approach has outstanding set of beneficial characteristics. Each approach brings advantages and disadvantages, so that the decision of the appropriate approach is in the focus of the investigation.

3.2.3 Vision based movement detection algorithms

Sequential captured images contain a huge amount of information about the absolute and relative movement of objects in every direction. So several vision based movement detection approaches were researched in the topic of UAV sta-

[21] The comparison of HWBIP and SWBIP approaches is visualized in 3.4

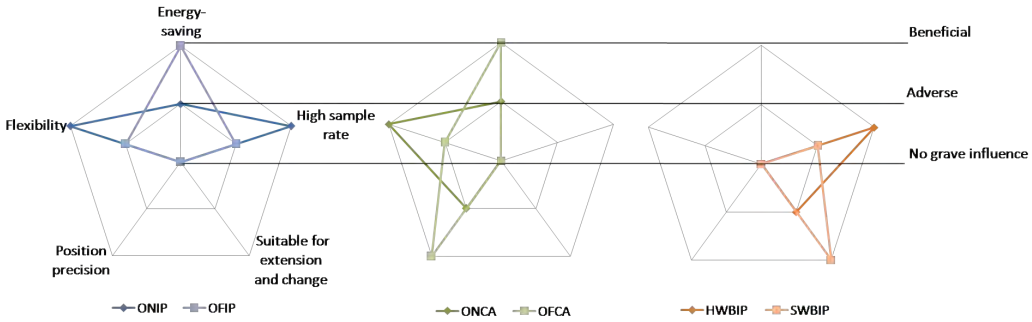


Figure 3.4: A comparison of the distribution behaviour of vision sensors

bility with different requirements to the information which extracted from the vision process[22]. A simple approach for a relative vision based control of a UAV was implemented by Boabdallah [26, pp.110-114 Position Sensor] by using a down looking camera and the Canny edge detector [Joh86] and the Douglas Peuker Algorithm for curve equalisation [DP73]. The drawback of this approach is that the field of vision must contain forms with edges which mean that the approach cannot result a satisfactory result if no edges are detected. A further approach for detecting relative movements and to build up a map for autonomous navigation, is visual Simultaneous Localization And Mapping (SLAM). This approach tracks features in the field of vision and reconstructs a relation to tracked features of previous images. The realization of SLAM [Bri07] in a UAV was executed in the work of Bloesch et al. [19] under consideration of real-time characteristics. The behaviour of the algorithm shows that the localization of the tracked features and the simultaneous mapping has a big impact at the time delay of the calculation. So the experiments and the control algorithms were researched and designed for 7.5Hz sample rate. Another popular tracking method for movement detection is

[22] These investigated approaches are visualized in figure 3.5. This figure includes pictures presented in [14, SLAM][13, Edge detector][15, p.27 Horn and Schunck optical flow]

given with the Optical Flow (OF), which can be described as movement observation of tracked objects or pixels in a sequence of images. Algorithms to calculate the OF were introduced by Horn and Schunk [Ber80], Lucas and Kanade [Bru81]. A few years after introducing the Lucas-Kanade-Algorithm, Lucas described the theoretical approach of visual navigation by using the OF. Thereby he described the possibility to detect movements and to calculate correspondence velocities. These velocities can be combined to a vector field which can describe movements in every direction [7, pp.40-45 Optical Navigation Theory].

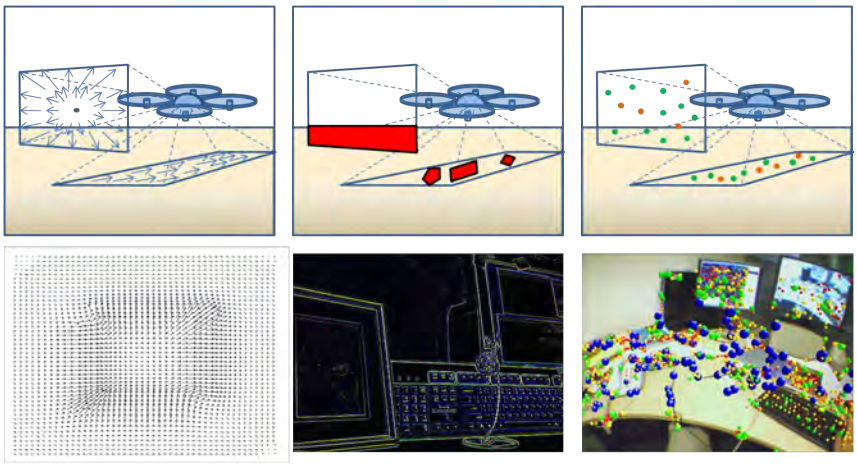


Figure 3.5: Samples for UAV stabilisation with Optical Flow, Edge detection and SLAM

3.2.4 Optical Flow

Researches which are related to image processing, mostly use camera models to reduce the complexity of the reality. This approach is also practicable in researches with OF. The figure 3.6 visualizes a camera model with a spherical, which illustrates a complex real camera and a flat image plane, which simplifies the complex model. Such simplification is achievable, if a mathematical transformation of the image plane can result a distortion-free image plane. Fach [18, pp.29-31 Chapter 3.3, Distortions] classifies reasons for distortions in a Camera

Calibration Domain (CACD) and the Shift of the Optical Axis Domain (SOAD). Pincushion- and barrel- distortions are summarized as radial distortions and caused by focal distance in relation to the captured image dimension. Further decentering distortions are based on the wrong alignment of the optical axis to the projected plane. Further Fach [18, pp.29-31 Matehmatical description of distortions] shows that these distortions of a physical camera can be eliminated with the equations shown in appendix 7.2. Thereby the problem of distortions is reduced to factorization of each distorted coordinate for the complete image in the initial camera calibration phase to eliminate distortions of the physical behaviour of the lenses and the image plane of the camera, and further in situations in which the camera is moved not planar to the projected plane. In the second case a fixed IMU of the camera system can provide the difference to the orthogonal gravity vector g, and can allow the transformation of the captured image in real time [23].

Following the OF projection suggested by Wei [34, pp.71-96 Obstacle Detection Using Optical Flow], there is described as a 2D projection of a 3D motion in the world on the image plane. Thereby Wei uses the ideal perspective projection model to introduce the projection of a point $P = (X, Y, Z)$, from the camera frame into the image frame $p = (x, y, f)$ where f is the focal length. The OF definition is derived with the assumption that the point P moves in the time t' to the Point P' with the corresponding projection on the image plane p and p'. The velocity on the image plane can be calculated as $v = p - p' = (x, y, 0)^T - (x', y', 0)^T$, where the component of z is cancelled out causing the 2D projection. Wei uses this projection to describe the problem of retrieving the OF velocity with 6 DOF. Further Wei described that researches for OF velocity determination only can based on a

[23] Such a camera was introduced in the year 2009 as standalone system from Fraunhofer Institute for Manufacturing Engineering and Automation ,See [6, pp.1-2]

reduced DOF model[24].

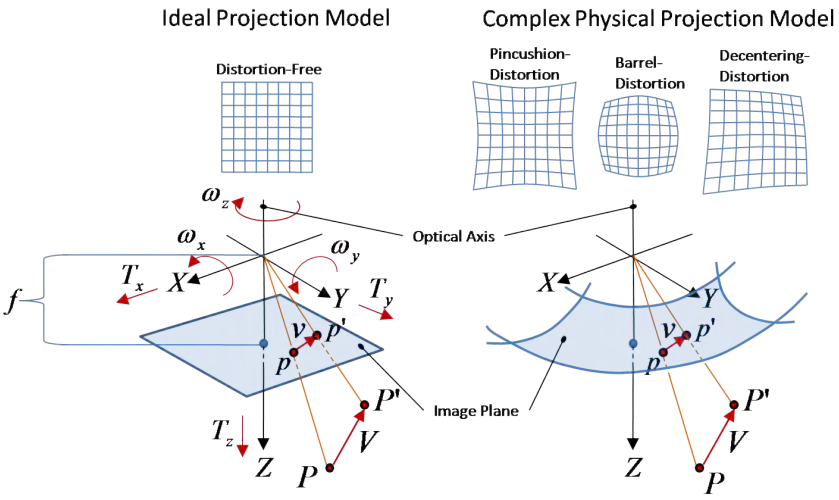


Figure 3.6: Complex Physical and reduced projection camera model

After introducing the projection camera models, the possibilities of OF determination can be focused with a top-down separation of algorithm classifications. These classifications groups OF algorithms to feature based, gradient based and correlation based methods. Feature based OF algorithms usual execute three steps for the determination of the OF field. First images of a sequence are scanned for good features. Such features can be corners or regions of contrast peeks. Further found features are compared trough the image sequence with the result to determine for each step an OF vector field. The feature based OF is robust against the aperture problem, which implies that just a part of the orthogonal movements can be detected in a image sequence with edges [A.Tratkowsky S.24]. The figure **??** [25] visualises the peeks of contrast regions, in this case the three hot-air balloon in the square of the image, which can used as good features. Based on this example, the drawback of the feature based OF is given in cases if no features can be found.

[24]The DOF of a ideal projection model are shown in figure 3.6 The can be summarized in a translation vector $T = (T_x, T_y, T_z)^T$ and a rotational vector $\omega = (\omega_x, \omega_y, \omega_z)^T$, See [34, pp.74-77 Motion Model Deduction]

[25]The presented image is downloaded from the website: http://www.1800skyride.com/

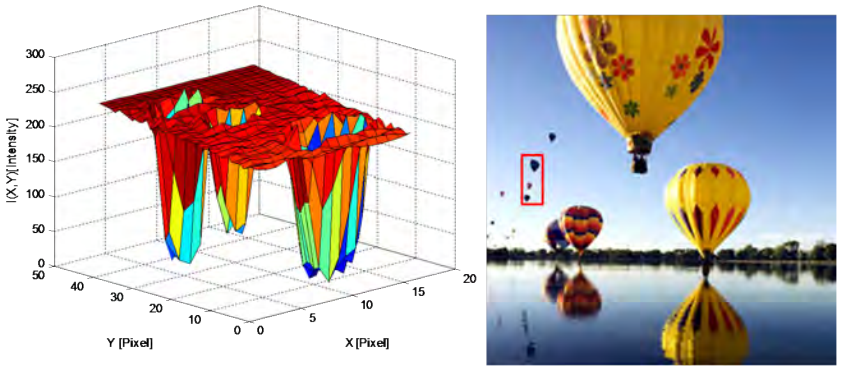


Figure 3.7: Detected Features Visualised as Peeks

In contrast to that, gradient based algorithms can determine a optical flow field, with no found features. This characteristic bases on the calculations of such algorithms, which use the intensity of images to determine the vector field. Thereby the movements of the intensities are determined with executing a minimising algorithm to the neighbourhood intensities. These determination bring the disadvantage of intensive calculation complexity, which can be a drawback for real-time applications [Ber80] [Bru81]. An approach to improve the calculation complexity of the gradient based OF determination with the pyramids approach of Lucas and Kanade [Bru81] is shown in figure 3.8. The idea is thereby to downscale the pixel resolution with grouping pixels into blocks with the average grayscale and to refine the image more and more until the intensity distances are determined satisfactory.

Correlation based approaches use the strategy of searching and matching segments trough a image sequence. Such algorithms work satisfactory, if the distribution of intensity in the processed regions of the image is constant over time. Another more conventional name, for this kind of algorithm, is block matching (See figure 3.9). Thereby the algorithm tries to match blocks from previous to the actual image in the defined section. The block size and the corresponding

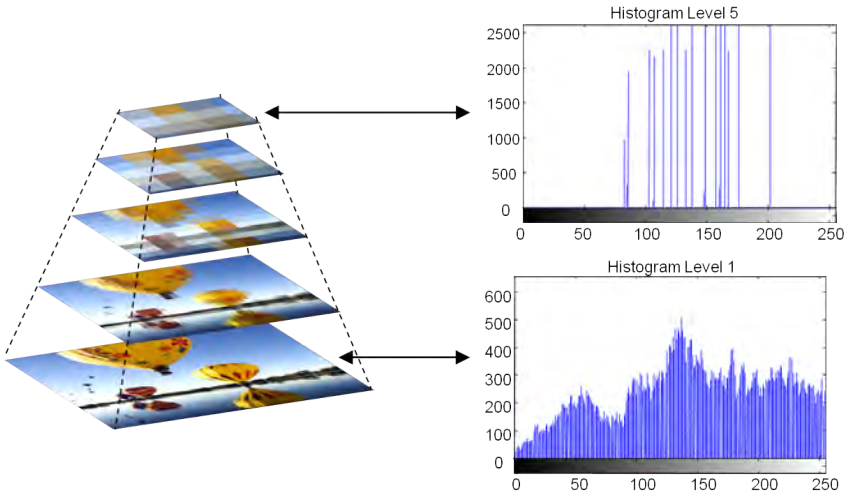


Figure 3.8: Lucas-Kanade-Algorithm with Pyramidal Scaling

search section influences the calculation time of processing. The result of the block matching algorithm is the vector shift of the blocks. This shift can be transformed in relation to the sample rate to the velocity vector field OF. One of the biggest advantages of correlation based approaches is the simplicity of processing and the and the uncomplex mathematical theory which allows a fast and easy evaluation of the process. Drawbacks of this approach are the memory intensity as well as the error sensibility which bases on wrong matches. Further the execution time characteristic of such approaches is related to the search algorithms and the realisation. Generally all approaches, introduced in this chapter, can be configured and executed unprofitable to the characteristics of the incoming image sequence which can further be a reason for unsatisfactory results.

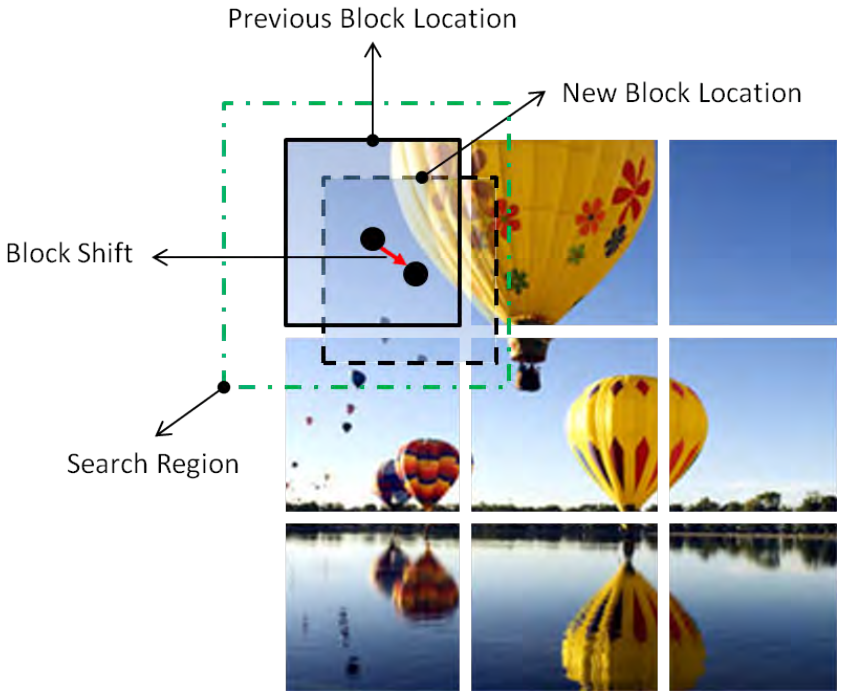


Figure 3.9: Block Matching Algorithm

3.3 Control Systems Characteristics

3.3.1 Control approaches

The closed loop feedback architecture is an approved method for controlling systems and nearly used in the most of the systems controls in industry and society. So the most of the researches in the UAV stabilisation topic were and still are executed with closed loop control architectures which are built up as Multiple-Input and Multiple-Output (MIMO) or as multiple Single-Input and Single-Output (SISO) systems. The differences between the researched approaches are the amount of the inputs and outputs of the physical process, the type of controller, the used measurements and the sampling rate. Thereby the type and behaviour of the controller is close related to the measurements and the physical process [23, p.3]. The classical control approach using Proportional-Integral-Derivative

(PID) controller was researched in several Hovering Unmanned Aerial Vehicle(s) (HUAV) projects [Ved08, pp.24-31] [26, pp.43-68]. These researches show that the classical PID controller is not robust enough to handle with complex models which include several integration states, but has to be transformed to cascaded PID controllers. Basically the implementation of a PID controller can be executed with the determination of three parameters for the given process. These parameters are the proportional, integral and derivative gain of the controller, which have an impact on the dynamic behaviour of the controlled value [23, pp. 695-698 PID Controllers]. Another approach for control improvement was introduced by Luenberger [Dav71] and describes a closed loop control approach, called state-observer, which simulates the process in real time parallel to the true process by using the input and output vector of the closed loop system and corrects the control strategy. Bloesch et al. [19, p.5] have shown in their research, that the problematic of sensors with non-negligible time delay can be solved to an adequate result by using a state-observer. The reason for that is that the state-observer allows to configure and to control the stability behaviour of the closed loop feedback architecture. If a process can be observed or furthermore controlled, is related to the states in which it can be and furthermore to the measurements of the DOF [23, pp.632-636 Controllability and Observability]. In the other hand, the state-observer has a big impact to the calculation capacity of the UAV target, because the real-time process simulation of a 6 DOF rigid-body becomes very complex.

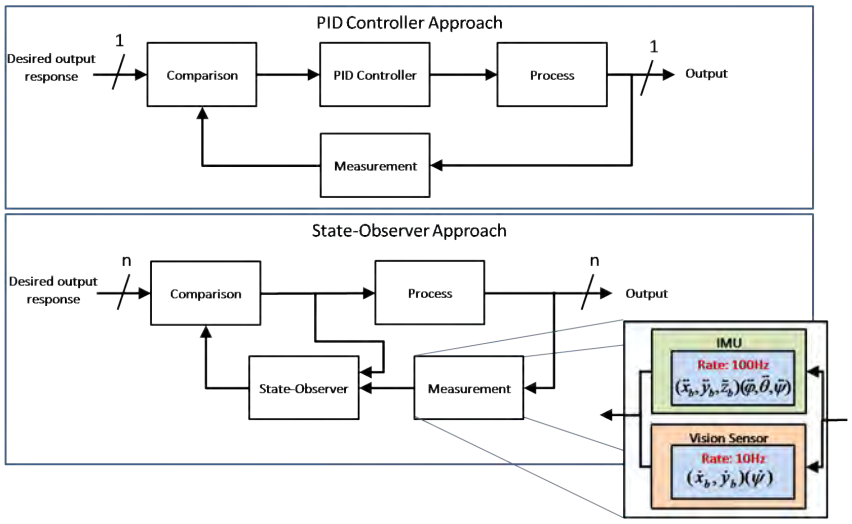


Figure 3.10: PID-Controler and State-Observer in closed loop System with variable sample sensor rates

3.3.2 Performance measures in the time domain

Control systems usual are designed and evaluated in different domains, under consideration of the special behaviour of these. The closed loop feedback architecture can be described with a transfer function which shows the behaviour of the closed loop feedback system in respect to the system boundaries. The time domain is used to visualize the time variant behaviour of the controlled value, and gives definitions of performance measures. These measures are visualized in figure 3.11[26] and describe the following characteristics [27].

- Stability: A control system is called stable, if the controlled value y(t) reaches a constant value and does not oscillate between a set of values.

- Precision: The precision describes the ability of steady state error minimiza-

[26] This picture extends the diagram presented in [22, Chapter 2, p.11 Characteristics of control systems]

[27] The content of the characteristics is derived from [22, Chapter 2, p.12 Requirements of control systems][23, Chapter 5, pp.228-230 The Performance of Feedback Control Systems]

tion of a control system. The best case of precision is $e(\infty) = y_{set}(\infty) - y(\infty) = 0$.

- Transient oscillation: The oscillation process of a system is described with the needed time of the function is needed to reach the range of tolerance T_s and is called settling time. This time depends on the rise time which describes the first entrance point of the range of tolerance. A short rise time can bring the disadvantage of a big maximal overshoot y_{max}, which should be kept minimal.

- Robustness: A control system is characterized as robust, if it does not become instable if system values change over time in scope of their tolerances. Such tolerances can occur from temperature variations or tribological reasons.

- Load of actuator: The load of the actuator(s) should be minimal. That means that the maximum actuator value should be equal to the needed value to prevent unnecessary work.

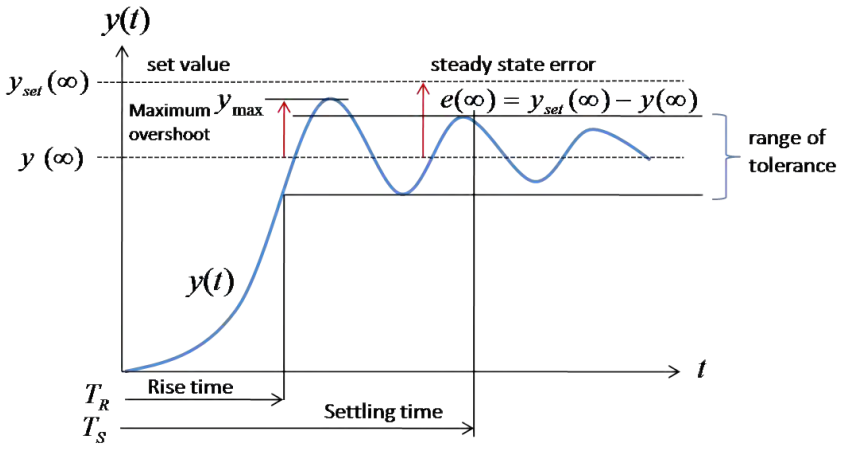


Figure 3.11: Performance measurement values of control systems

3.3.3 Continuous and discrete time and frequency domains

The necessity of the linear approximation and the advantage of simplification of systems, led the analysts to the use of the Laplace transformation. This method transfers relative easily solved algebraic equations in the Laplace domain for the more differential equations in the time domain[23, pp.41-47 The Laplace Transform]. Combined with the fact that the sampling rate of a computing system can have a big impact to the stability of it, the z-transformation allows transferring functions from the continuous frequency domain to the discrete frequency domain [23, pp.749-754 The z-Transform]. These two transformations and the corresponding retransformations are visualized in the figure 3.12[28] and can be executed in a simple form by using transformation tables [29]. The frequency domain equations are describes as functions of the complex number s. This continuous frequency parameter is discredited by using the following abbreviation $z = e^{sT}$ where T is the sampling rate of the discrete system. The factor k in the discrete domain symbolizes the k^{th} sample value of the function and can be described as parameterization factor of the sample time T [22, Chapter 4, p.19 z-Domain set value transfer function of a closed loop control system].

3.3.4 Stability criteria of transfer functions

The mentioned simplification of Closed Loop Control System(s) (CLCS) in the following chapter can be shown by reducing the complete system to a quotient which in contains polynomials for the numerator and denominator. This charac-

[28] This figure bases on the theorems in [22, Chapter 4, p.17 Definitions and Rules of the z-Transform]

[29] See, Transformation tables in [23, pp.42 Important Laplace Transform Pairs] and [23, pp.751 z-Transforms]

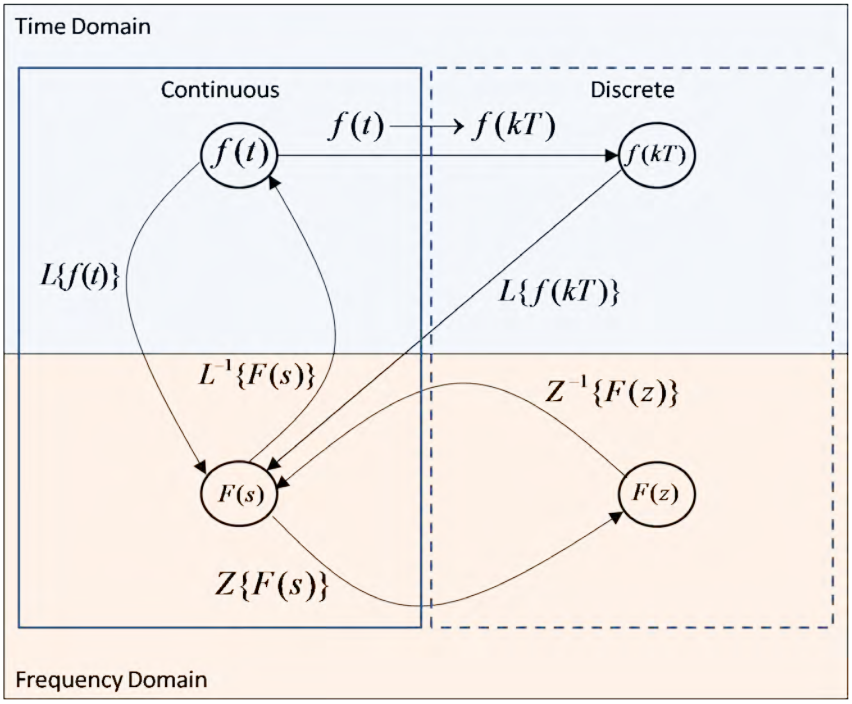


Figure 3.12: Transformation possibilities of continuous and discrete time and frequency domains

teristically quotient is called Transfer Function (TF) and describes the behaviour of a system in relation to its boundaries. In the continuous domain the TF can be created as Set-Value-Transfer-Function (SVTF) (See formula 3.12) or as Error-Value-Transfer-Function (EVTF) (See 3.13) which allows investigations from two different input perspectives of the CLCS. The discrete closed loop control system can be used to describe the dynamic control behaviour of a discrete controller which controls a continuous process from the SVTF perspective (See formula 3.14). The interface between the process and the controller in this case is realized with sample and hold components which discretize the continuous process [23, p.747 Sampled-Data Systems]. The figure 3.13[30] visualizes the generic architecture of CLCS in the continuous and discrete domain. Thereby the figure

[30]This picture extends diagrams presented in [22, Chapter 4, p.2 Generic linear Closed Loop Control System],[22, Chapter 4, p.19 z-Domain set value transfer function of a closed loop control system], [22, Chapter 4, p.21 Stability]

shows that the components of a global TF of a CLCS also can be denoted as TF which describe the specific behaviour of the controller, the process [31] and measurement[32].

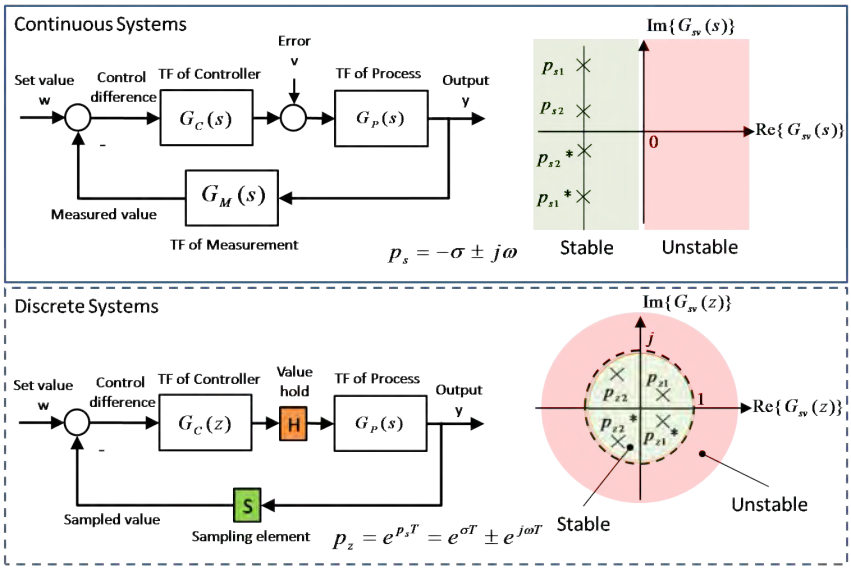


Figure 3.13: TF of continuous and discrete CLCS and their stability behaviour

The corresponding TF of the introduced domains are shown in the formulas 3.12, 3.13, 3.14[33]. Especially the SVTF contains important information of the stability behaviour. So continuous SVTF are stable, if the poles p_s of the quotient are located at the real negative section of the complex plane. Equivalent to that the stability of a discrete SVTF is given, if the poles p_z are located in the unit circle of the complex plane [34][35]. This stability behaviour in the complex plane is visualized in the figure 3.13. Consequently the sample rate of digital systems and the corresponding sensors has a big impact to the stability of a CLCS [36].

[31] In specific literature of control systems, sometimes the synonym plant is used for process
[32] The measurement component in the discrete domain is substituted with a sample component which holds the value of the process in the desired appearance
[33] See, [23, p.754 Closed-Loop Feedback Sampled-Data Systems]
[34] This causes the quantisation relation of the frequency $z = e^{sT}$
[35] CLCS which have poles located on the edges of the stability sections are called semi-stable. Such CLCS are totally not robust and also assigned to the set of unstable systems
[36] See, [23, p.756 Stability Analysis in the z-Plane]

$$G_W(s) = \frac{Y(s)}{W(s)} = \frac{G_C(s) * G_P(s)}{1 + G_C(s) * G_P(s) * G_M(s)} \tag{3.12}$$

$$G_V(s) = \frac{Y(s)}{V(s)} = \frac{G_S(s)}{1 + G_C(s) * G_P(s) * G_M(s)} \tag{3.13}$$

$$G_W(z) = \frac{Y(z)}{W(z)} = \frac{G_C(z) * G_P(z)}{1 + G_C(z) * G_P(z)} \tag{3.14}$$

4 Design and Implementation

4.1 Analysis of Existing Quadrocopter Architecture

Before the focus can be concentrated to the distributed error correction scheme of the HSE quadrocopter system, the overall existing architecture has to be introduced in this section. The importance of this introduction causes the necessity to understand and analyse the system which has to be simulated. An overview of the realized and pending components is visualized in figure 4.1. This overview includes the main systems of the general architecture which are Quadrocopter, Remote Control, Global Position System (GPS) and the Base Station. Thereby the Quadrocopter is controlled over a permanent unidirectional wireless communication to the remote control, which cyclic sends a pulse width sum signal with the steering information. Thereby the RC-Receiver interfaces an microcontrollers' Enhanced Capture Timer (ECT) of in the CCU, which allows the real-time decoding of the information. Further for observing and testing purposes, the architecture allows to observe and to configure parameter and state variables of the quadrocopter in mid-air. To fulfil this, the quadrocopter can send and receive data via a serial[1] ZigBee[2] communication from the base station, which runs a Java-Application that captures and visualizes the data and allows configurations in real-time[3]. By regarding the quadrocopter system, it can be considered

[1] This wireless communication interfaces the Serial Communication Interface (SCI) of the CCU

[2]

[3] In this case real-time means that the data is pre buffered at the quadrocopter and asynchronous send to the base station. This allows a fine grained resolution of data capturing and evades time delays of synchronous sending data

as an architecture with CCU mounted and uncoupled components for detecting movements of the system. These components have special behaviour in aspects of resolution, quantisation etc. and bring a impact to the control and flying behaviour of the quadrocopter.

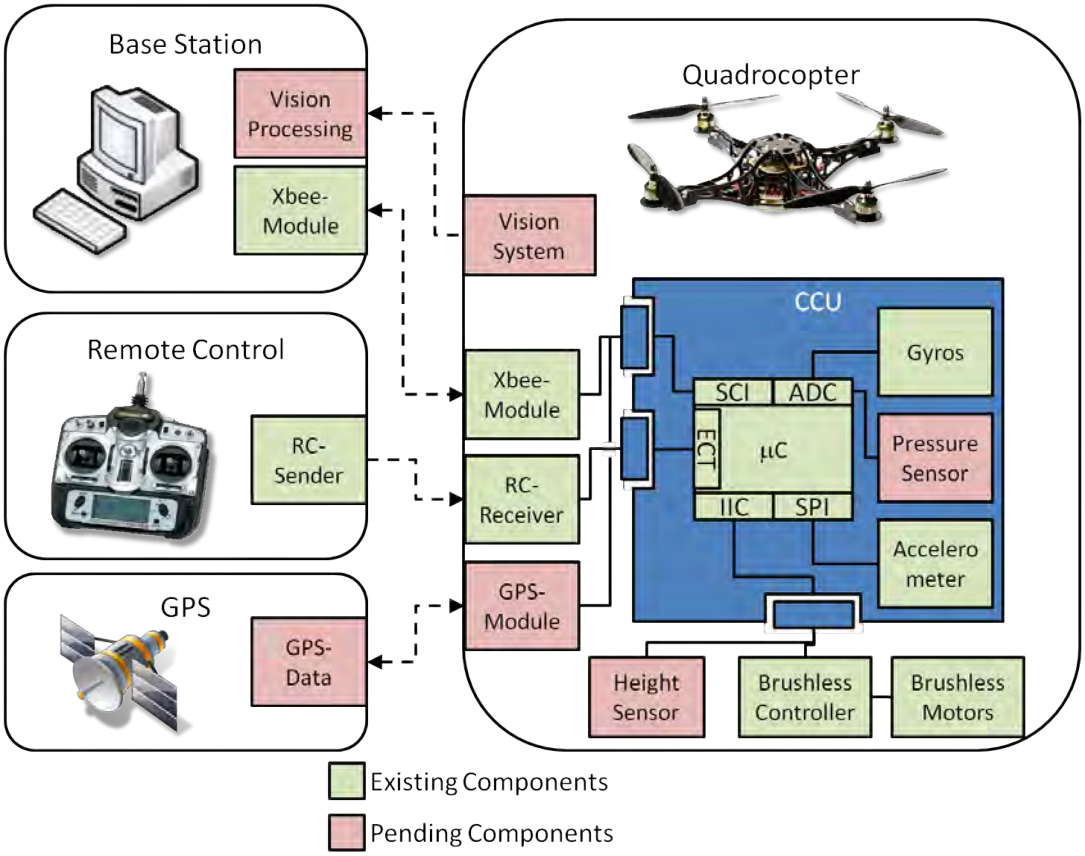


Figure 4.1: Existing Quadrocopter architecture and future developments

4.1.1 Central Control Unit

The CCU of the HSE Quadrocopter, which is one of the main interesting components in relation with a simulation, is a outcome of a interdisciplinary corporation of the HSE faculty of electronics and mechatronics in Göppingen [4] and the HSE

[4] See http://www.hs-esslingen.de/hochschule/fakultaeten/mechatronik-und-elektrotechnik.html

faculty of information technology Esslingen [5]. The dedicated central unit of the CCU is a MC9S12XDT256 [6] microcontroller from Freescale, used for the flight control task and additional tasks sensors mounted on the Printed Circuit Board (PCB) for calculation of actual position in space. The architecture includes an accelerometer [7] measuring the acceleration in X_B-, Y_B- and Z_B-direction, three gyroscopes [8] measuring roll-, pitch- and yaw-velocity and an air pressure sensor to determine the height. Furthermore a battery sensor (voltage divider) on the board allows the calculation of the new set-points for the brushless controllers [9], because the resulting RPM/thrust of the motors depends on the actual battery voltage. For communication with the base station, a XBee [10] module is used directly mounted on the flight control PCB. Besides that, there are a couple of external interfaces available for remote control, Inter-Integrated Circuit (IIC) bus to command the brushless controllers and Background Debug Mode (BDM) as programming interface. The interfaces for a distance sensor, a GPS receiver and a servo (e.g. for a camera mounting) are available for future extensions (See pending components in figure 4.1)[16, p.57 Central Control Unit]. By taking a closer look on the IMU, it is apparent that the location of the sensors is designed WRT the velocity or acceleration, which has to be measured. So the sensor which measures the accelerations of the translation movements WRT the B-frame is mounted in the middle of the CCU to avoid the measuring of rotational components. Similar to that, the gyroscopes also mounted to positions, with the focus to zeroise measurements of unwanted values. So it is possible to describe with these combination of sensors the mathematical model presented in chapter

[5] See http://www.hs-esslingen.de/hochschule/fakultaeten/informationstechnik.html
[6] See [HCS, Data-sheet of HCS12X family]
[7] See [LIS, Data-sheet gyroscopes]
[8] See [ADX, Data-sheet gyroscopes]
[9] See [BLC, Wiki Brushless Controller Bl-Cl V1.2]
[10] See [XBe, XBee Pro Module]

3.1.2. As described in chapter 3.3.4 another important point in digital systems is the quantisation behaviour. In case of the given CCU there two aspects which have to be considered. One aspect is the sample time of the control system and the other the resolution, sensitivity, noise etc. of the IMU. The sample time of the system thereby depends on the load of the microcontroller. This load can be configured or changed by changing the embedded software with the focus to reduce the load. In contrast to that, the behaviour of the sensors cannot be changed, because these work autarchical and provide the specified performance defined in the data-sheet. So it is important to regard this behaviour of the sensors in the simulation of the system. As visualised in figure 4.2, the translational sensors provides a different behaviour as the rotational sensors. This causes to the interface and the internal functionality of the sensors. The acceleration sensor provides a digital readout of the measures via Serial Peripheral Interface (SPI). Thereby the quantification is given with the quotient of the gravitational acceleration and the sensor sensitivity $(g/340)/LSb$ [$(m/s^2)/bit$]and the limits are specified with $6g$ in each direction. Differently to the digital readout, the gyroscopes which build up the rotational part of the IMU provide an analog signal with maximal detection of $5.23 rad/s$, which is represented with a voltage level between ground and reference voltage of 3.3V, and need to be digitalized with the controllers' Analog to Digital Converter (ADC). This ADC provides a resolution of 10bit and combined with the gyroscope it results to a resolution of $0.0142(rad/s)/LSb$ [11]. So the IMU provides measures across the B-frame which fulfil the rotational part of the GVV ν and the translational part of the derivative GAV $\dot{\nu}$.

[11] The calculations of this chapter are derived from [Rob10]

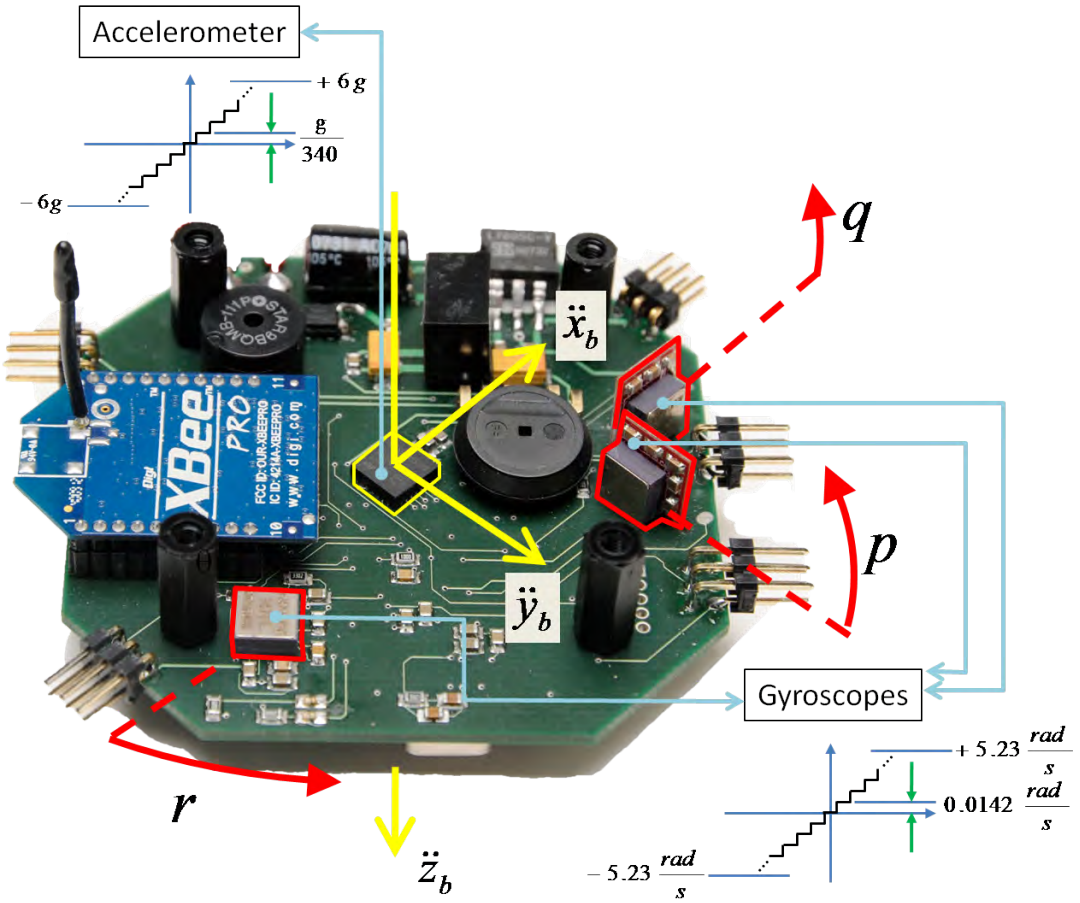


Figure 4.2: CCU and IMU

4.1.2 Characteristics of Sensors and Actuators

An important aspect, in relation to the quality of control and further the flight behaviour, is the characteristic of the IMU and actuators in view of precision. In the case of the CCU here, the noise sources are the noise of the Microcontroller (MC), the noise drift of the MEMS characteristic in relation to the temperature and the vibration of the motors. Because of this mix of noise sources, the determination of the average noise cannot determined by scrutinizing the data-sheets of the sensors. A way of determination, which is executed here, is an experiment which includes the capturing and investigation of the real sensor readout in the preferred hovering state. Experiments with running engines and shut off engines,

show that the biggest impact to the total noise originates from the imbalance of the rotors. This effect originates to the fact that small irregularities of the rotors have a big impact to the noise in case of the needed rotor speeds for the hovering state. The results of the captured sensor data in a duration of 11 seconds are visualised in 4.3 with the corresponding S^2_{N-1} derived in appendix 7.3. The S^2_{N-1} in this case is an measure which shows the dimension of noise influence, and further the sensor values which get lost because of they cannot detected.

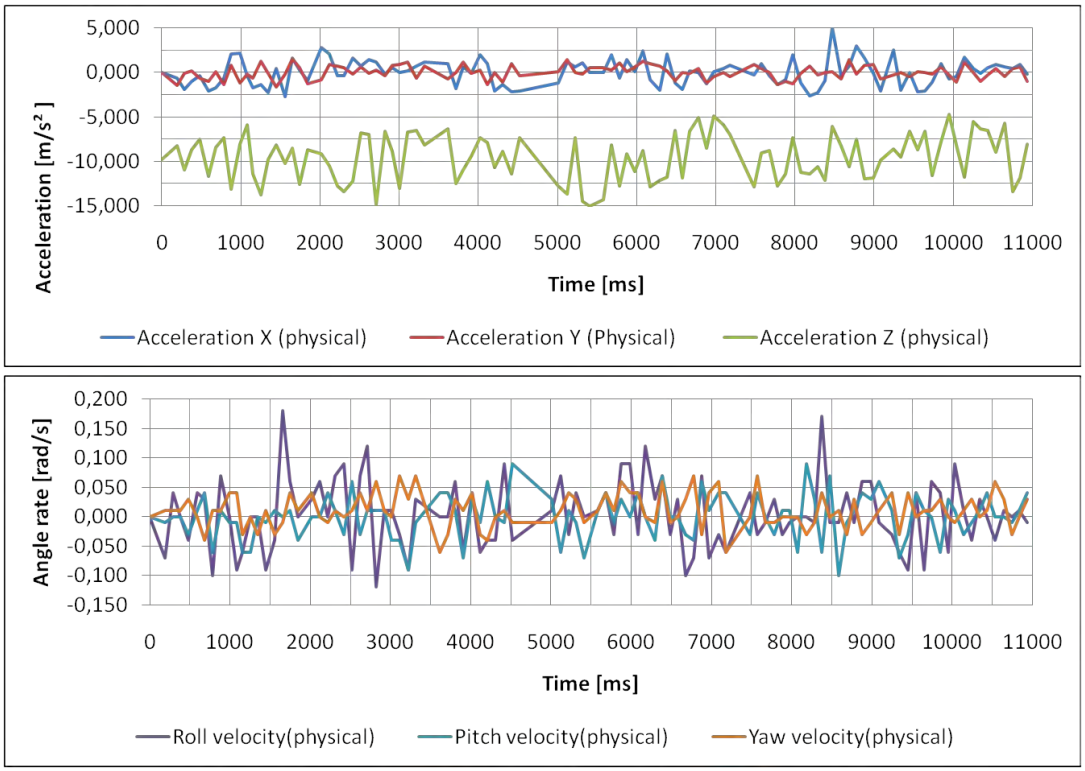


Figure 4.3: IMU noise analysis in hovering state

The investigation of the actuators can be executed by using a stroboscope with sufficient resolution of the frequency. This approach also was applied in this project here. The picture 4.4 visualizes the experiment, which was captured with a satisfactory exposure time of the camera. So the frequency of a rotor can be located, by vary the frequency of the stroboscope. Finally the located rotor will optically stand still, and the other rotors will optically rotate in the original direc-

tion, or in the reverse direction. This behaviour demonstrates which engines run faster or slower as the located engine.

Figure 4.4: Experimental environment for set-point drift measure

With the execution of the described experiment, it is possible to identify the set-point drift of the particular engines. The results of the different set-point drift are visualised in the figure 4.5. The mentioned set-point of the engines is a 8bit value, which is set to the control register of the brushless controller. Thereby the force, needed to lift up the dead weight of the quadrocopter equivalent to the force needed for hovering state, is nearby the set-point of 100 on a scale from 0 to 255. As visualized in the curves in 4.5, the particular engines have a characteristic realization of the required set-point input, which can vary from ideal set-point. These engine characteristics also depend to the mounted rotors and further environmental and physical behaviour which can varies over time. So the exact reconstruction of the engines' characteristics, can be unsuitable for a simulation. A better approach is to define an area of set-point drifts and to consider the variations inside this range in the construction of the control system.

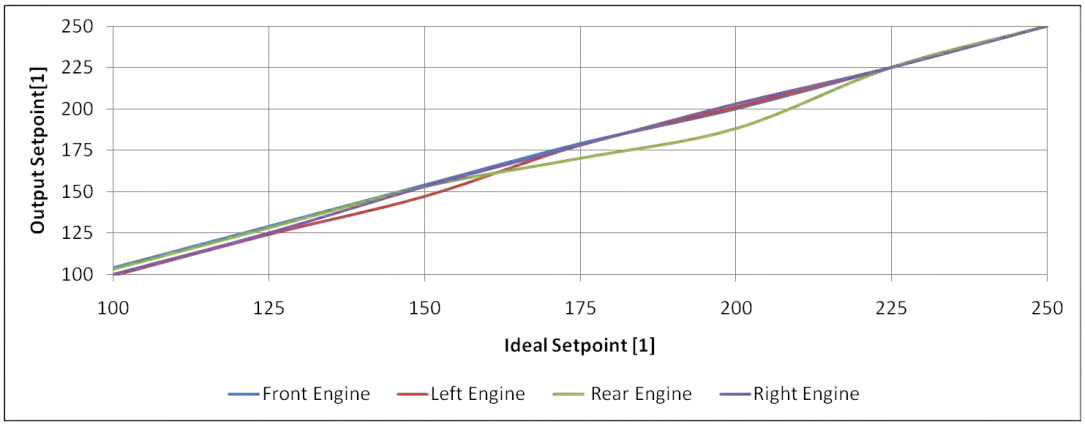


Figure 4.5: Set-point analysis of engines

4.1.3 Embedded Software Architecture

Aspects like exchangeability, modular simulation, modular testing and dependency minimisation lead to an encapsulated modular layer design of the embedded software of the quadrocopter. The composition of the quadrocopters' embedded software, shown in the component diagram in 4.6 [12], is extended with the real-time behaviour of the call hierarchy. Thereby modules, which use an asynchronous interrupt and a corresponding service routine to handle events, are marked with a flash icon. Another important information is presented with a special dependency which contains the Cycle Time(s) (CT) of the component call. This value helps to understand and to research the sample behaviour of the sensors and further the control behaviour. Generally as well as in this project here, the Hardware Abstraction Layer (HAL) encapsulates the control registers, which are directly related to the peripherals and the MC and allows a generic access to the functionality of the hardware. The quadrocopter HAL contains generally five points of service access. These interfaces provide beside the service of initialisation and update of the quadrocopters' real-time image, access to the commu-

[12] This diagram is derived from the architecture presented in [16, p.77 Embedded Software]

nication with the basestation, a on board diagnosis system for noticeable visual and acoustic warnings and notification and the access to the system timer to facilitate Time Triggered (TT) execution in the main routine. Further the real-time image, located in the data layer, provides a protected access to the actual state and data of the quadrocopter. So the state of the quadrocopter, is accessed by the application layer, uncoupled from the HAL and at a central data pool which allow only one state of each value. At the top level of the layer architecture, the application layer contain the control, filter and basestation communication logic of the system. Thereby the main routine uses a TT call strategy to realize Deadline Monotonic Scheduling (DMS) with specific CT of the components. Two important points of this scheduling architecture, also described by Audsley et. al [Aud01] , is the fact that the sum of the execution times of every routine (including Interrupt Service Routine (ISR)) has to be smaller as the smallest deadline which is in this case 5ms, and further the biggest deadline has to be smaller as the repeating period of execution. This constraints limit the calculation complexity of filter, control or further on-Board algorithms drastically and also re interesting in case of an simulation.

4.2 Adopted approach

After the introduction of the existing quadrocopter architecture and the analysis about the relevant characteristics of the soft- and hardware-modules, the adopted approach which will researched here has to been presented in this chapter. Based on the introduction of the distribution approaches, presented in chapter 3.2.2, the ONCA in combination with OFIP and SWBIP approaches will researched here. As mentioned in the comparison of these approaches, the advantages and dis-

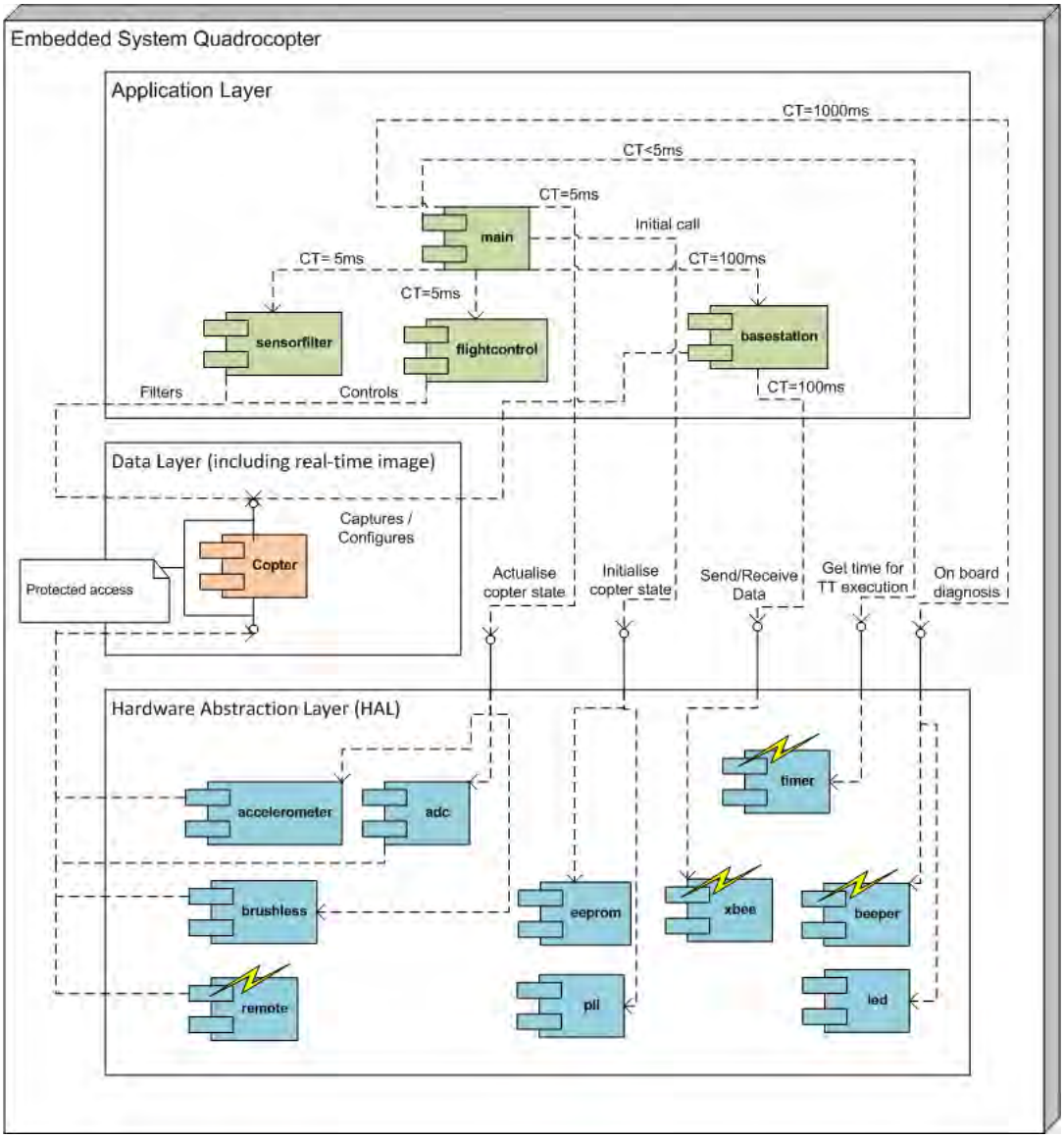


Figure 4.6: Software Components and their relationship of the Embedded System Quadrocopter

advantages are related to the aspects of interesting. So the aspects of simulation prototyping and behaviour investigation of the image processing, as well as the delay tolerances in relation to the control behaviour lead to the mentioned approach decision. As shown in figure 4.7 the adopted architecture is composed of the simulation of the base station with the corresponding image processing and the simulation of the quadrocopter components which are in focus of interest.

Beginning with the set values of the embedded system, the remote control component provides time variant signals of the angular set values for roll and pitch, the angular velocity value of yaw WRT the B-frame and the thrust value. These set values are the primary input for the body related control, which interfaces the physical model simulation with the same input values which are provided from the CCU, and the output values which control the actuators. Withal the input values are transformed to a desired form, which is used for PID control. Beside the IMU values of the CCU, the physical model provides an image sequence related to the position values of the quadrocopter WRT the B-frame. Thereby the image capturing process is simulated with a high resolution underground image and the projection of the desired image resolution on it. These image sequence is provided to the simulated on-board vision system according to the ONCA architecture. Further the vision system includes an send process with an output image-buffer, to provide the images to the base-station, and an receive process with an input correction-value-buffer. At the other side of this wireless correspondence the base-stations' process also used a buffered I/O, and determines the translation (u and v) and rotational values (r_{of}) from the calculated OF field. Finally the additional controller which realises the drift elimination in the hovering state is located in the quadrocopter system, and observes the set values of the remote control. The idea thereby is to find a value state of the set and sensor values, which symbolises that the hovering state is desired by the pilot, and further the quadrocopter body is ready to reach that state. To realize that behaviour a state machine, located in the module "Check Hovering State", decides in real-time the activation of the Vision System. So the set values of a second controller, located in the separate position control module, are switched from zero to the received correction values. This has the effect that the body controller is delegated by the second position controller and corrects the position with a correction flight

manoeuvre. If the pilot sends new set values, after the quadrocopter has reached the hovering state, the position controller is deactivated and allows the pilot and further the body controller an uninterrupted batch-processing of set values.

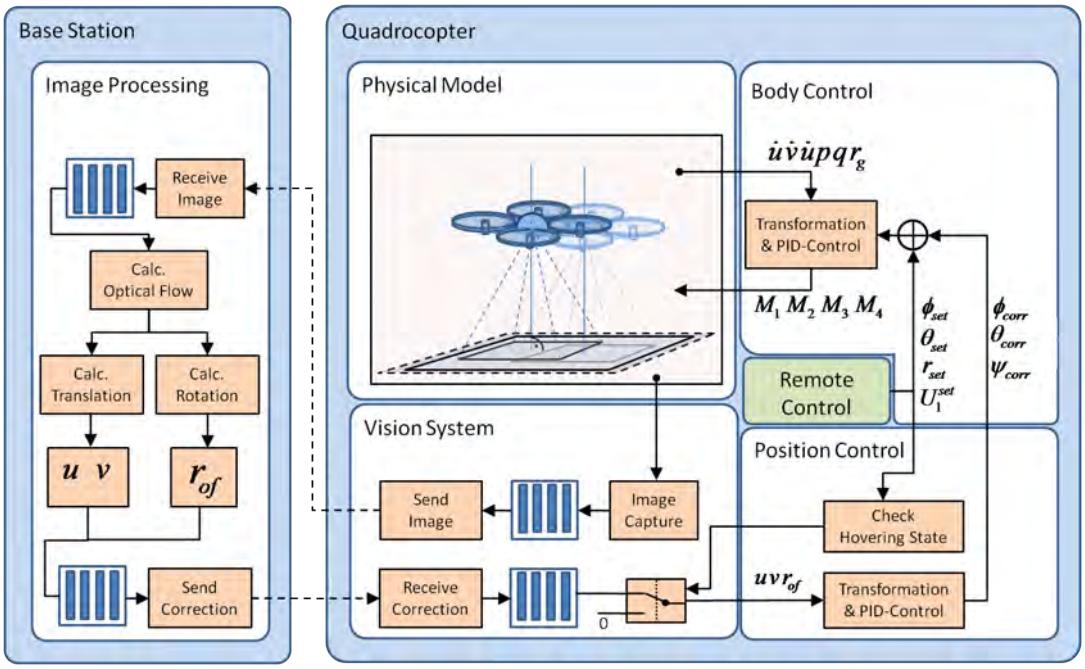


Figure 4.7: Adopted approach architecture

4.2.1 Control Approach

Before starting the design of the quadrocopter controller, it is helpful to understand and analyse the abstract behaviour of the separated, physical processes, and to proof the correct function of the approach. This strategy lead in this project here to the construction of a simulation which shows the behaviour of realising an angular set-point value to one axis of the quadrocopter which include two engines as actuators. As visualised in figure 3.2, two opposing engines realise the angle movements in the specific pitch- or roll-axis. These engines can realise an individual gain to the given set-point and influence the precision of the control system. Another point which is problematically in this case, is the fact that

the feedback of the CLCS, the angle acceleration has to be integrated two times. The combination of the unprecise gain of the engines and the two integrations influences the stability of the CLCS enormous. To simulate this behaviour and to impart the expected effects, a simulation is build up and executed in simulink with two configurations. As visualised in 4.8 the plant includes two engines, simulated as a sequence of a direction and individual gain. Furthermore two integrators transform the angular acceleration to a angular velocity and position which is used for feedback for the cascaded PID-Controllers. This feedback is designed as switch which can be interrupted, to fulfil a tests scenarios with single and cascaded PID control approaches.

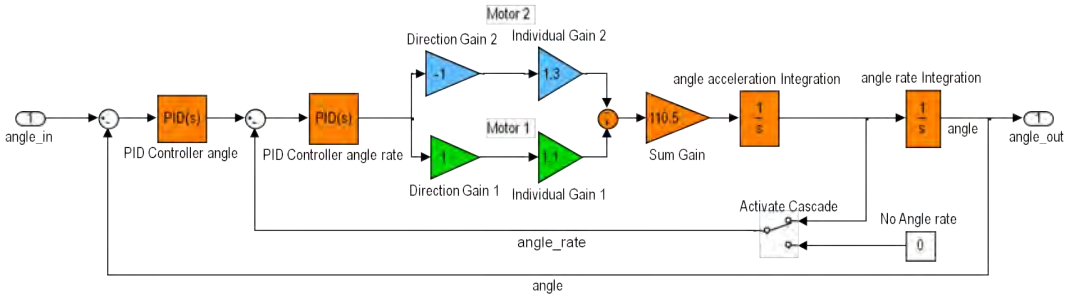


Figure 4.8: Abstract CLCS of one Angle

As introduced in chapter 3.3, the stability behaviour of CLCS can be shown with a pole diagram and the investigation of the location of the poles (See Pole Map 4.9). Another method, to proof the stability in the frequency domain, is given with the Nyquist Criterion [23, pp. 487-493, Relative stability and Nyquist Criterion] and describes that each integration in a system affect a shift of 90° of the frequency phase. The criterion describes that the phase at the x-axis' point of 0dB of the corresponding gain has to be bigger as $-180°$. So also this theorem proofs, that the determination of a value which passes two integrators, must have a PID-cascade to ensure that the feedback prevents a critical frequency phase shift. These relations are visualised as bode diagram in figure 4.9 with logarithmic axis of the

gain, frequency and phase. Furthermore the step response of the PID-cascade and the single feedback PID-controller (See Step Response 4.9) shows the stable and the oscillating unstable behaviour of these configurations.

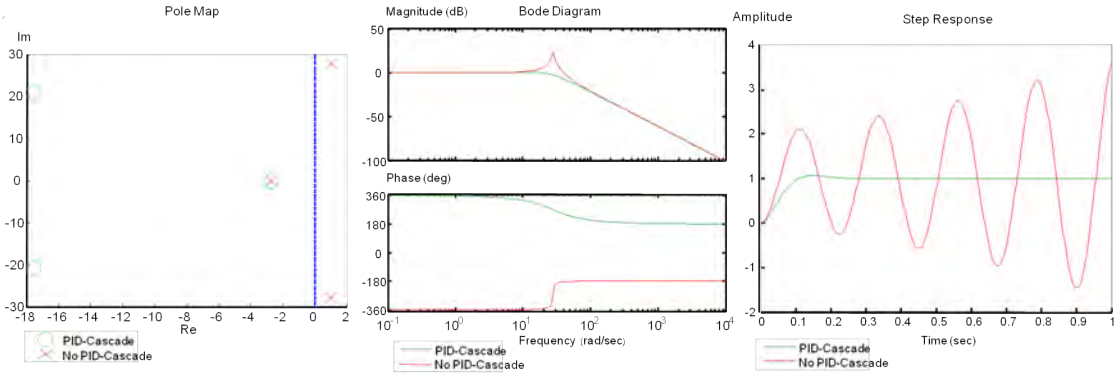


Figure 4.9: Pole Diagram, Bode Diagram and Step Response of one Angle System

4.2.2 Problems, Limitations and Assumptions

This chapter focus the problems that exist in the real implementation of a vision system for error drift correction of an UAV, and further shows abstractions and assumptions that are made for the structure and implementation of this projects' simulation. Such characteristics can be separated into domains, which provide the basis of problem and limitation analysis and allow to define a abstraction level. This abstraction level can be used to focus analyse just the characteristics of an specific part of a solution. This allows to concretise and locate problematic behaviour before searching for a solution at the wrong section. A concrete example for this argumentation can be given by regarding the camera domain. Characteristics like the focal length influences the sharpness of the image and influences the image processing algorithm to calculate the OF. But this characteristic can be classified as problem which have to be solved after the characteristic of more basic problems are investigated. So the focal length f can be abstracted with setting

the height Z, between the image plane of an ideal projection model and the underground, equal to f and to assign both values the fixed value of one meter. This allows to project the values of the ground points $P = (X, Y, 1)^T$ directly to points on the image plane $p = (x, y, 1)^T$ [13]. This abstraction is also assumed in this project. Further assumptions of the camera domain are the camera does not create distortions [14], the optical axis is projected rectangular to the underground [15] and the exposure of the environment is constant over time [16]. So the assumption in relation of the DOF of the camera system can be described as same as the DOF of the quadrocopter body, but reduced in the rotational movements around the axis which originates in the middle of the image plane. The quadrocopter body domain defines the behaviour of the movements of the hovering state, which can be assumed as a 5 DOF by reducing the movements across the axis Z_B. The realization of these two different DOF system can be realized with a real-time image correction process as described in 7.2 or with a hinge mechanism which keeps the image plane planar to the ground. In the simulation, this assumed state of the camera domain, is realised with a state machine and the orthogonal projection of a simulated camera image on a underground image as mentioned in chapter 4.2. Another domain which includes limitations and has to be defined here, is given with the underground domain. The assumptions which have to be made here is that no objects exist which moving relative to the quadrocopter e.g. no moving light reflections and the underground image contains points and contrast which can be evaluated in the OF algorithm. For example, a underground image of a snow-covered landscape which contains segments with no contrast is not us-

[13] A detailed mathematical deviation of this abstraction presented in [18, Chapter 2, pp.15-16 instructions of transformation]

[14] This fact also is implicated with the usage of the ideal projection model

[15] This implicates that the camera movement is planar to the ground and the camera is mounted correctly on the UAV. This assumption is adopted from [34, p.78 Motion Model Deduction]

[16] Experiments with a real camera involve a correct camera calibration (see appendix 7.2), and a environment preparation of the exposure to realise these characteristics

able for this approach. Finally the communication of the base-station and the quadrocopter is abstracted as delay. This abstraction excludes communication fail, synchronisation or segmentation problems and assumes a smooth operation of the image and correction value transmission and reception. This behaviour is also symbolised with synchronous buffering systems in figure 4.7

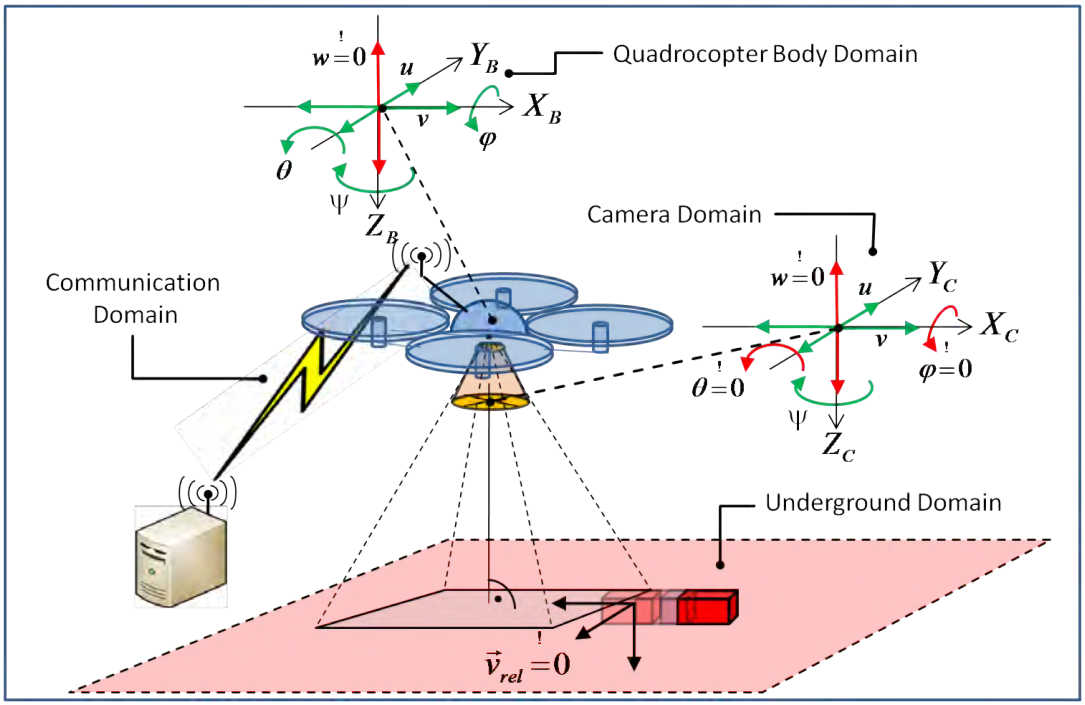


Figure 4.10: Problems and limitations separated in domains

4.3 Simulation of Embedded System Quadrocopter

The architecture of the embedded systems simulations, is constructed by following the modular design. The focus of this execution was to full-fill the aim of an exchangeable and flexible simulation architecture, which can be changed and reconfigured with minimal effort. This design criteria results to the architecture of Simulink modules, as visualised in 4.11, and allows a hierarchical realisation

of the quadrocopters' system simulation. The separation of modules contains a control library which allows to use several control modules in relation to the development state. As shown, the first development is executed as prototype in a native Simulink model without having complexity or resource limitations. This model is refined to a embedded matlab model, which contains only Matlab code which is cyclic executed. This refinement allows to simulate the quantisation and limitation behaviour of a embedded controller and further to evaluate if the implementation algorithm works correctly. The last step is to implement or to generate the embedded controller to the quadrocopters' embedded system target language C, and to evaluate this with the HIL module over a serial interface. This last step includes aspects like resource limitation analysis, algorithmic calculation complexity ect. Based on this method, this project here focus the development of a Simulink controller in the abstract level by using the IMU and Optical Correction (OC), which can be implemented and realised in the same way as the existing control algorithm which controls only by using the IMU. The sensors and actuators are grouped in a own library and allow generic customisation of a sensor and actuator model and can be exchanged if the real hardware is changed. This exchange behaviour is also given in the other components of the embedded quadrocopter system simulation and can be extended with implementation of new modules which satisfy the existing interface architecture. The optical-sensor has a special function in this project. It allows running tests and simulations of the quadrocopter system with a reduced simulation complexity. This is fulfilled because this module can generate the behaviour of the image processing signal with mathematical functions and can so reduce the simulations' execution time. A further modularisation design decision was executed with the EOM to allow reusing the simulation with different EOM and a Dynamics module. Finally the utility library contains tools for analysis, visualisation and steering of the quadrocopter

simulation, which have a important function in the performance test process of this project.

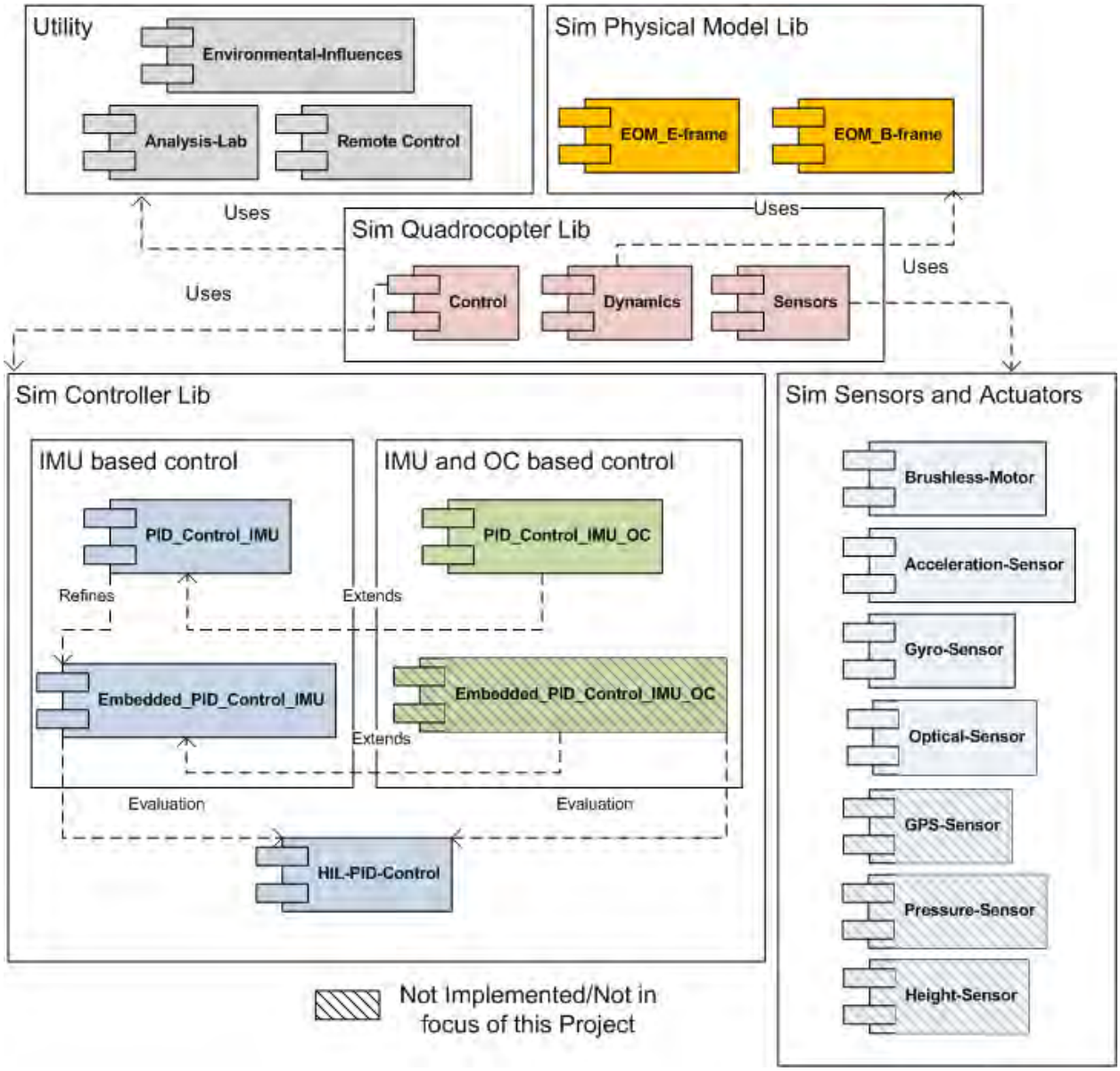


Figure 4.11: Modular Overview of the Embedded Quadrocopter System Simulation

The customised embedded quadrocopter system architecture is shown in figure 4.12 as Simulink model which contains a CLCS architecture. The set values from the remote control can be generated in different ways and support a virtual flight with a 3D-mouse or further a flight manoeuvre constellation of steering values.

This signal is a composition of pulses defined with the Heaviside-function [17] multiplied with ramp-signals and can be accessed for batch-simulation. These set values are realized in motor values by the control component. This component executes, with respect to the mentioned development state, a controller simulation till a HIL communication. In each case, the controller knowledge of the quadrocopter movements are provided by the sensors. In this special customisation the input signals of the sensors are provided by the Dynamics block and can be assumed as optimal. Thereby the sensor block provides the signal with disturbances like noise and quantisation as in reality. Further the correction values for the drift elimination, are already known by the sensor block and can be also disturbed by regarding the signal quality of the optical movement detection. Another way to determine these correction values, is to send these via the output of this model to the base station and to receive the corresponding correction. The signal pos-vel-acc includes all signals, produced by the Dynamics block, and provides a generic interface between the blocks which works with it. Another strategy to test the stability of the control algorithm is the creation of environment influences. This can be executed, similar to the remote control, with the generation of a velocity vector signal stimuli which is add as error to the actually optimal velocity value calculated by the EOM. This error signal also can be accessed outside of the model for batch-processing simulation.

4.3.1 Flight dynamics

The realisation of the dynamics block in Simulink, is based on the researches presented in chapter 4.1.2 and on the EOM presented in chapter 3.1.2. First the

[17] See [Wei11]

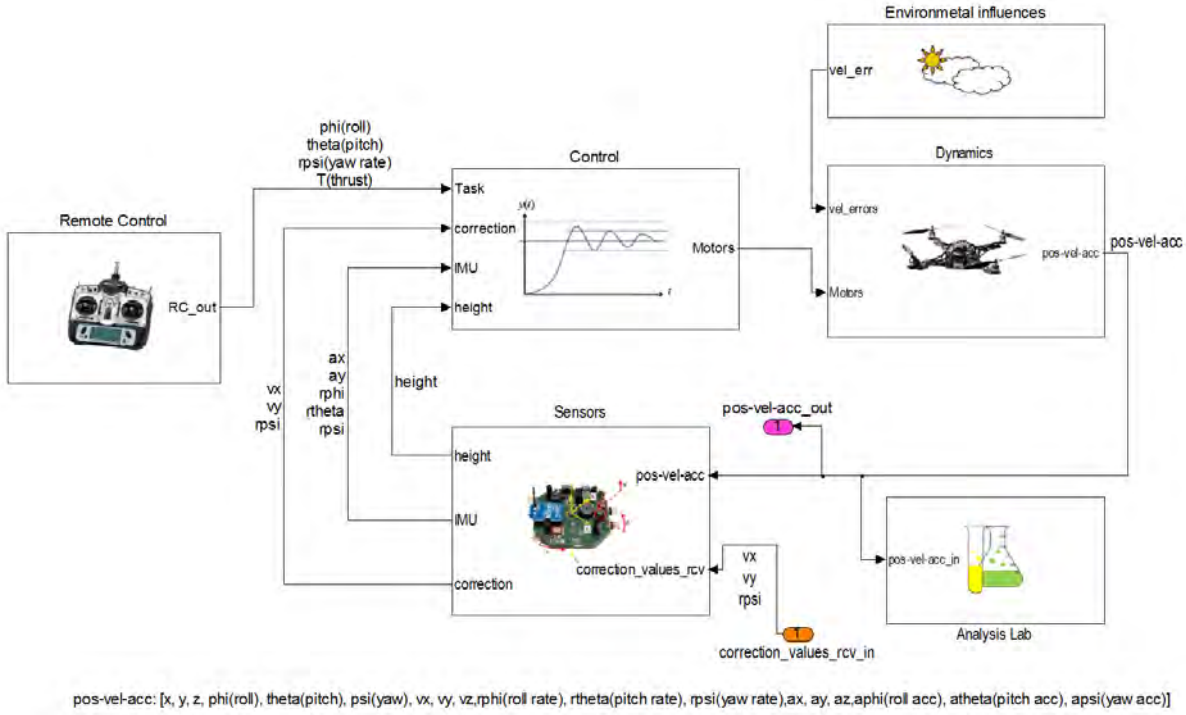


Figure 4.12: The Customised CLCS Architecture in Simulink

derived GAV $\dot{\zeta}$ WRT the H-frame was analysed to determine the dependencies of the ODE. Based on these dependencies, the physical model was build up with the aim to ascertain all positions, velocities and accelerations of the quadrocopter body needed for this project. The result is presented in a schematic view in the figure 4.13. Thereby this module gets the register set-values, corresponding to the brushless controllers in the reality (See4.1.2), and passes the mentioned pos-vel-acc vector. In this process, the actuators engine blocks transform the set-point to a corresponding force F_n [18] and Revolutions per Minute (RPM) value Ω_n (See 4.3.3). These values of the different motors are composed to the movements U_n, by executing the calculations presented in equation 3.11, and further to the resulting RPM value Ω. Furthermore these output values are passed to the calculations, which based on the component $\dot{\omega}^B$ WRT the B-frame of the equation 3.10. Additionally to determine the velocity and the angle, this vector is integrated two

[18]The index n stands for the number of engine and ranges from 1 to 4

times. A special behaviour of this block is that the calculation of the $\dot{\omega}^B$ needs also its own values. So the start values have to be initialised in this case for the initial execution with the values zero. Equivalent to the B-frame component of $\dot{\zeta}$, the E-frame component $\ddot{\Gamma}^E$ from the equation 3.10 is used to calculate the translation acceleration, velocity and position vectors. Thereby the translations also passed trough two integration processes. These calculated values are combined to the output vector, and provided as output. The GVV V^B, which is needed to simulate the on-board behaviour of the acceleration sensor, is determined with the inverse rotational matrix multiplied with translation GVV Γ^E. Further signals which are not needed in this project are also contained in the pos-vel-acc vector. The generic aspect of this approach is, that a generally EOM dynamics model also will have the same output values, and will be compatible to the same interface, if it has 6 or less DOF. Furthermore the determination of all movement values allows the simulation of future developments with each kind of movement detection sensor.

4.3.2 Control

Observing the analysed control behaviour, presented in chapter 4.2.1, the final quadrocopter controller is MIMO controller which includes several cascaded PID SISO controllers. The figure 4.14 visualises the detailed controller architecture. Starting at the input values of the control module, the angular velocities, the translational acceleration values and the set values are combined for control usage. Thereby the different sources of the values are symbolised with superscript symbols like Gyr for gyroscope, Opt for optical sensor, Acc for acceleration sensor and Set for set-value. These input values run first through a process of transfor-

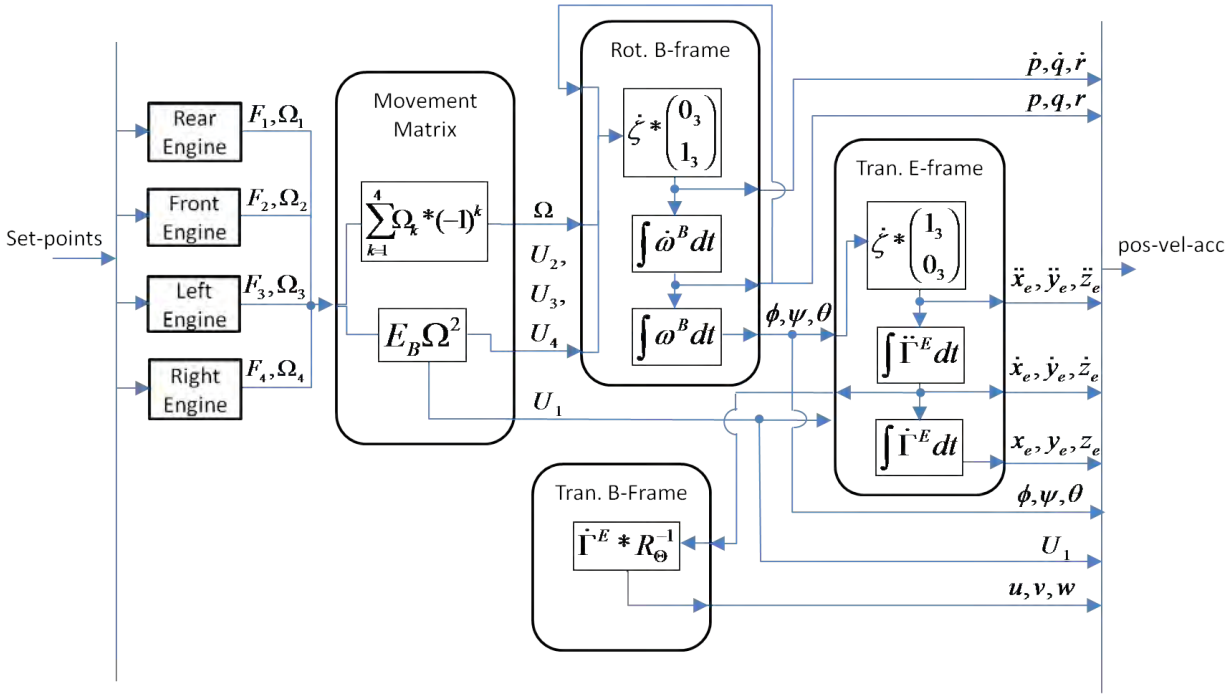


Figure 4.13: Schematic Realisation of Dynamics Module

mation, which converts the raw sensor values to usable physical values for the control process. In this case, the angles for pitch and roll are not determined with a integration, but are calculated using the translational accelerations of the corresponding rotation axis. This is possible because the gravitational orthogonal force to is related to these rotations, and can be used to determine the angles without a integration process. The conversion of the raw values to physical values allows an easier control realisation and processing and is executed with the conversion behaviour presented in chapter 4.1.1. The converted physical values are passed to the Position Correction process and to the Body Control process. Thereby the Position Correction process passes the set-values to an internal state-machine which detects the Hovering state and activates the position correction controllers. For realising this, the state-machine checks the set-values of the angles and checks if these are smaller as a defined threshold, which symbolises that there is no activity of the remote control. After reaching this threshold, the state-machine switches

to a state which delays the activation of the position correction controller for the time t_{wait}. This time is needed for the body controller and the related cascaded PID controllers on it, to reach the steady-state of the hovering position. The activation of the position controllers, which try to reach minimise of the input values of the optical sensor, affects that the output values of these controllers are integrated to the first stage of the cascaded PID controller in the body control. The result is, that this collaboration of the controllers affects the needed behaviour for position correction of the hovering state, because the error value of one controller architecture is passed as set value to the other. Finally the cascaded PID control realises movements, given from remote control or manoeuvres needed for position correction and passes the angular values to a conversion block for the motor set-points. This block realises the variations of the particular motors, by realising the desired thrust set-value U_1. Another important characteristic in the design of this simulation is the option to run the body and position control PID controller with different sample times (t_s^{opt} and t_s^{imu}). This behaviour originates to the fact, that the IMU sensors can provide a higher sample rate as the optical sensor, and the sample rate can influence the characteristics a CLCS (See 3.3.4).

4.3.3 Sensors and Actuators

A simulated dynamic model, as introduced in chapter 4.3.1, provides as mentioned optimal movement describing signals which does not reflect the reality. So an artificial process which creates a disturbing signal fused with the optimal signal, can be used to investigate and simulate the realistic behaviour of the sensors and actuators. These characteristics, also regarded in this project here, with the result to transform the optimal signals with the experienced knowledge of

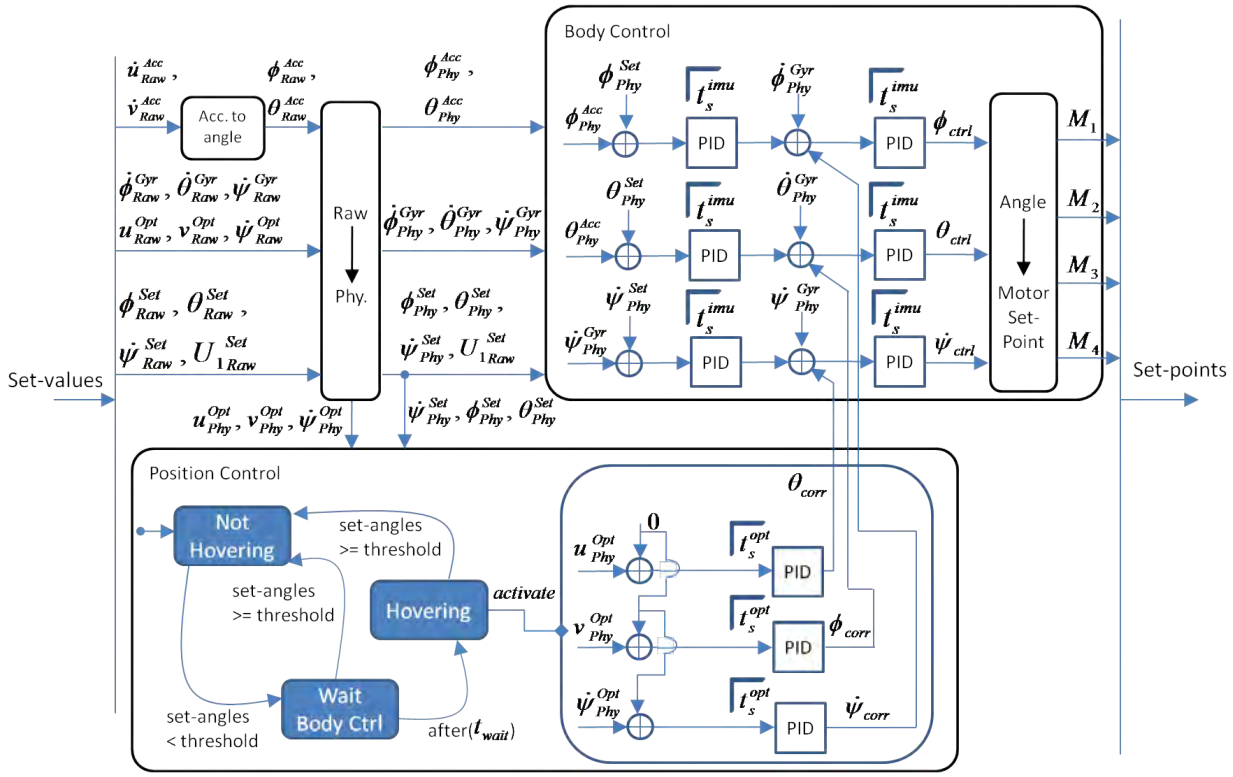


Figure 4.14: Quadrocopter Control Architecture

chapter 4.1.2. The results are presented as Simulink models in 4.15. Both components, sensor and actuator, contain similar functions to limit and amplify (Saturation and Gain) the incoming signal. The sensor (Gyroscope and Acceleration) further contains a quantification, offset and noise process to fulfil these typical effects. The sensor model, can so be customised individually by configuring these mentioned characteristics, and can so provide a model close to reality. In the other hand in the actuator model the only configurable characteristic is the motor gain, which is used to simulate the error characteristic of the engines presented in chapter 4.1.2. Beside this, the brushless motors access the same tables for the Set-point to thrust and thrust to RPM conversion, which are determined experimental. Further the important behaviour of the actuator delay is simulated as a delay process in the frequency domain. The presented generic blocks of this chapter, are customised once for the simulation of the existing hardware of the

quadrocopter and integrated in the sensor module CLCS shown in figure 4.12. So in focus of this project here, the characteristics of these blocks are not varied for further investigations.

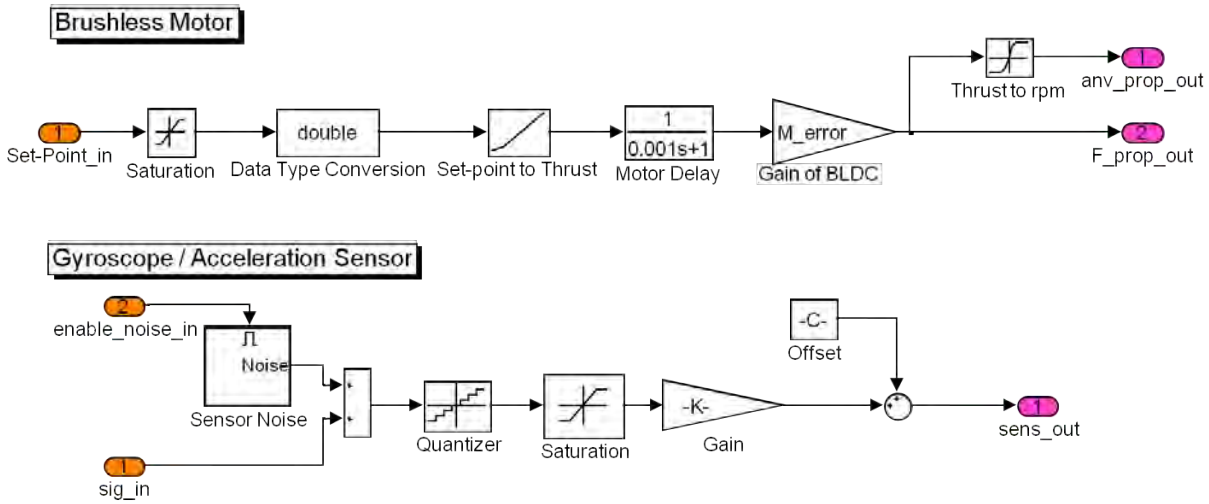


Figure 4.15: Quadrocopter Control Architecture

4.4 Simulation of Base Station

The aims considered in the base stations' simulation, are the exchangeability of image processing algorithms and modules, and further the ability to vary characteristics of the captured images. The aspect of algorithm exchangeability is realised with a architecture, which separates the image processing from interpretation of the derived values. Regarding this aspect, different image processing modules provide different form of values. Especially several algorithms which are implemented in different frameworks show Simulink and the corresponding calculations for the different output value interpretation. As shown in figure 4.16, the OF algorithms further are also logically separated in high and low level algorithms. This hierarchy architecture aspect provides the ability to mix high and

low level algorithms from different frameworks. Beside the algorithm architecture, the simulation of the base station provides variation of source and sink of the processed video. The source thereby can be an synthetic video which is generated with the assumptions mentioned in chapter 4.2.2, or a real captured video. Thereby the synthetic video provides the ability to vary the image size and frame rate, which can not be executed in real videos. Similar to the simulation of the embedded system quadrocopter, the movements of the camera view can be steered with the remote control module or with the environmental influences. Thereby the environmental influences can provide a desired trajectory with variable velocity or acceleration, which can be used as reference to evaluate the quality of optical movement detection. Finally the analysis lab, as mentioned in the embedded system quadrocopter simulation, also provides the option to observe and analyse customised models, build up with the modules of the simulation.

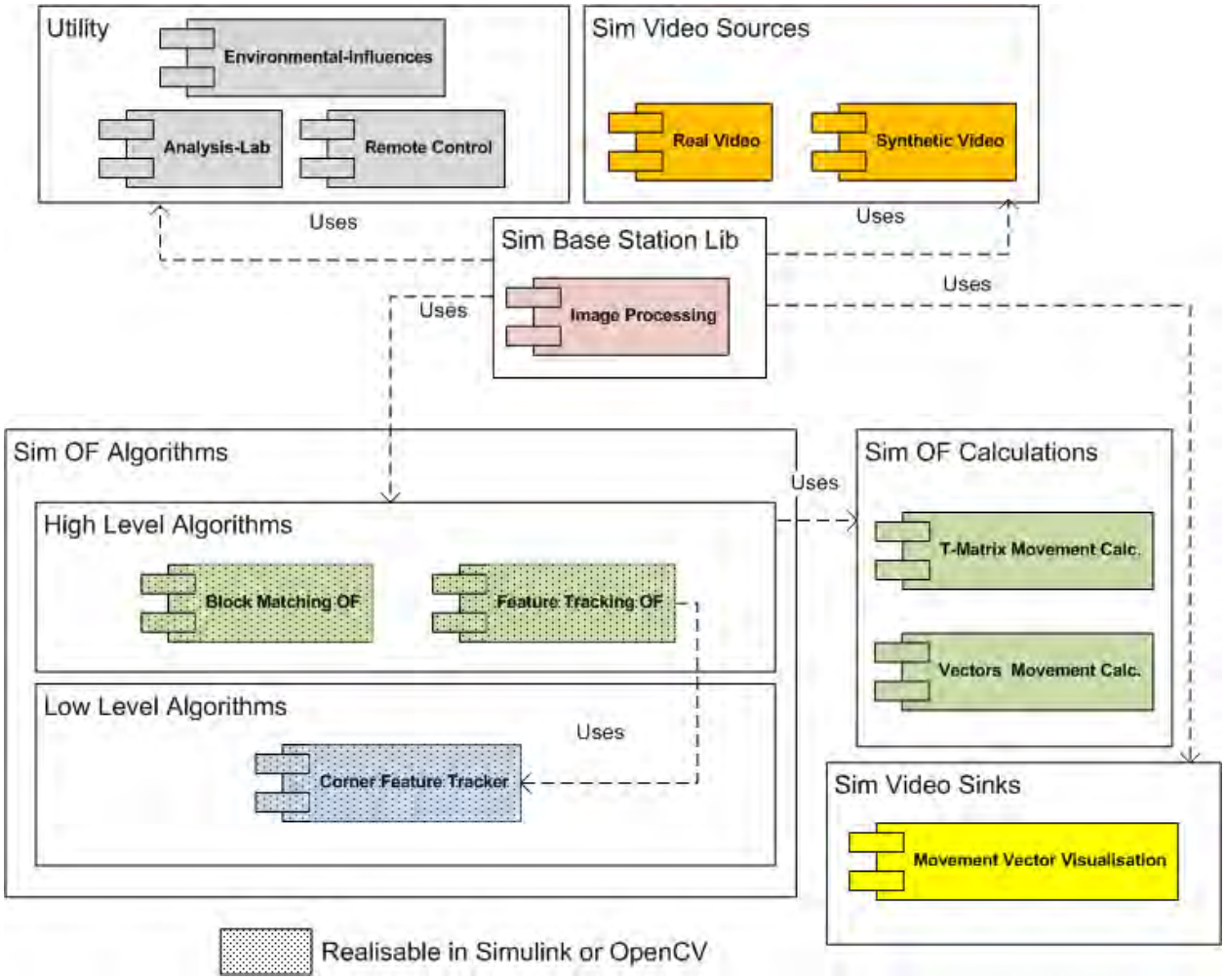


Figure 4.16: Quadrocopter Control Architecture

4.4.1 Video Sources: Synthetic Video vs. Real Video

As mentioned, the video input of the optical movement detection process, can be created synthetic or a real captured video can be used. Regarding the synthetic video, the frame rate and image size can be configured because the generation of the image stream is created with a transformation of the image window executed on the ground image. This transformation is executed using a transformation matrix, which describes the translation and rotation of the next image. The sample-rate configuration thereby is realised in the integral which contains the

sample time of the image capture t_s^{Opt}. In contrast to that, the real video source process, can just decrease the sample rate by selecting and forwarding each n^{th} image. Further it is not possible to determine the trajectory of the image movement, because the video caption was executed with in relation to the physical movement of the camera [19].

It can be summarised that the synthetic video can be used for prototyping and executing test cases. The benefit thereby is that the possibility to capture measures, influenced by unexpected effects, is very low and the input-data of the test cases is created and realised exact. So the experimental results can be regarded as isolated from unexpected influences, and can provide the essential behaviour of the tested components. In the other hand the real video provides a realistic trajectory scenario, which contains a lot of influences like distortions, light effects, noise and so far. So the real video approach can help to fine tune filters and to optimise the image processing, but should not be used for the initial prototyping steps of development. Another important aspect in the real video approach is that the trajectory should be measured in the real world with a reliable measurement configuration. This has to be great deal more precise as the expected precision of the image processing to prevent misjudgements of the evaluated results.

All in one, the synthetic video approach is suitable for the first prototyping simulations and the real video for the final fine tuning. The challenge thereby is the crossing level from synthetic to real video. That means the synthetic video simulation has to reach as much as possible the real video, before the exchange of the video sources.

[19] The synthetic video architecture is derived from video mosaicking memo, presented at http://www.mathworks.com/help/toolbox/vision/

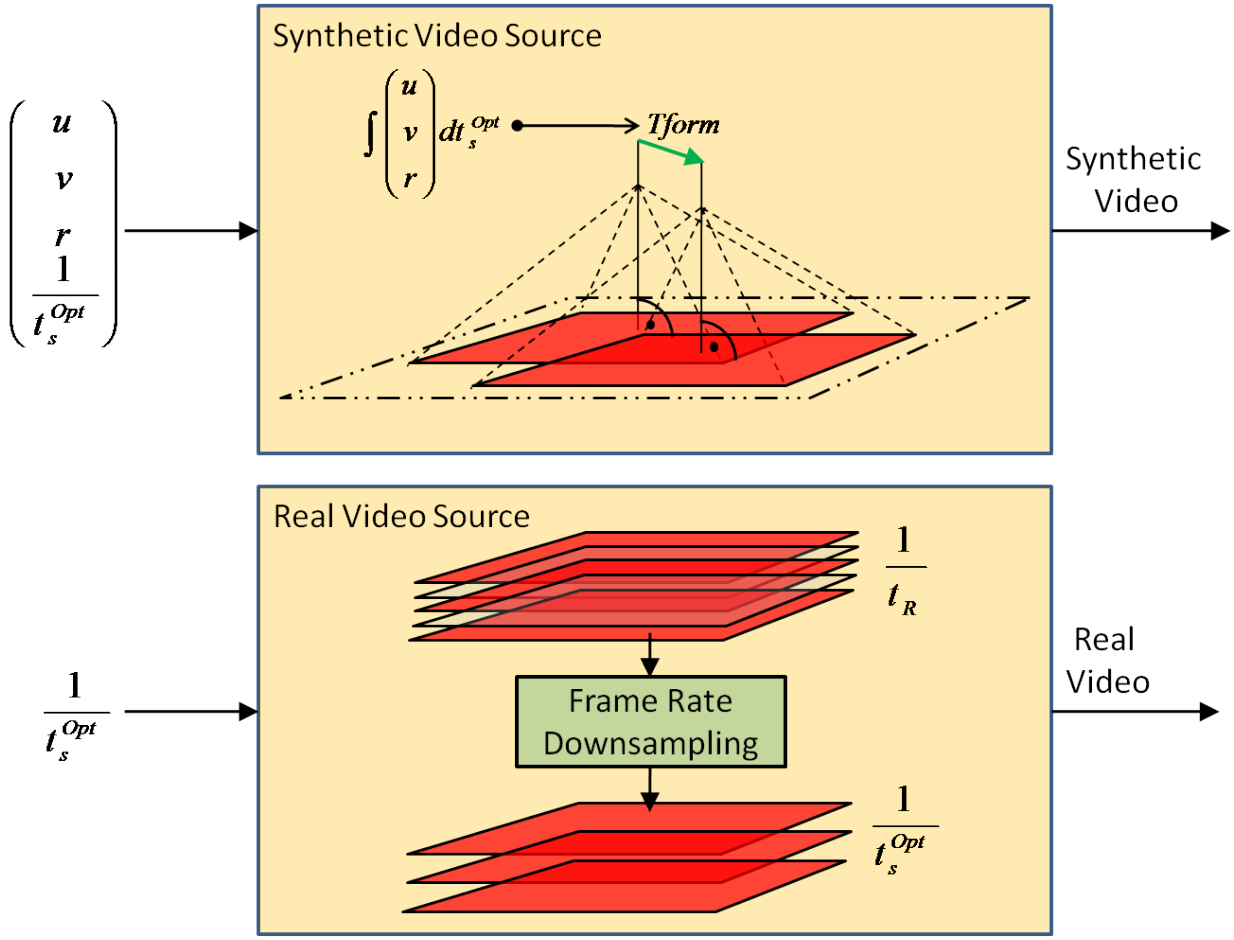


Figure 4.17: Synthetic and Real Video Generation

4.4.2 Calculations

The interpretation of the optical flow is executed in this project here in two different ways. These approaches, mentioned in the architecture overview 4.16 as calculations, provide the same output with different input representations. The first approach, presented by Termtanasombat [Naw10], is called Flow Number Calculation (FNC) and determines the quantity of the optical flow by calculating the average of optical flow over the magnitude of the velocity vectors. A similar calculation is visualised in figure 4.18, with the difference that the velocity vectors of the raw optical flow are represented as complex numbers. Further the average

calculation is executed with the accumulation of the average of the imaginary and real part in each direction. This resulting vectors are symbolised with ω_i with the index corresponding to the direction and axis. Thereby, the assumptions in chapter 4.2.2, just allow the mentioned movement set which does not contain the movements in Z_B direction and a corresponding Focus Of Expansion (FOE) [20].

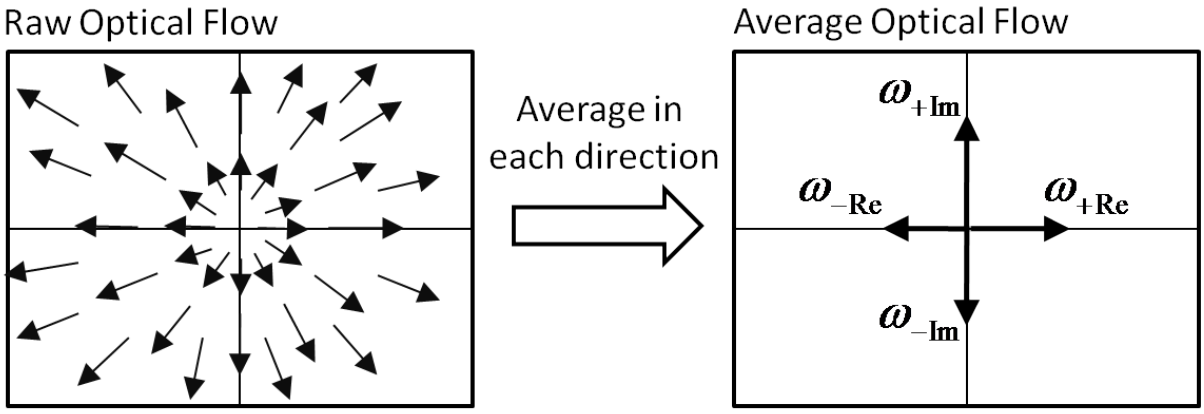


Figure 4.18: Averaging Process of Raw Optical Flow Field

The defined DOF allow to calculate the desired velocity values of movement from the average optical flow. This procedure is shown in figure **??**. There we can see a combination of rotational and translational movement as average optical flow. First, to determine the rotational movement, the smallest magnitude of the velocity vectors is the used as indicator. The idea behind that is, that the rotation around the centre of the image is represented in each average velocity vector. Additionally, just a rotational movement can influence in parallel all four average velocity vectors. So the separation of the translational and rotational movements can be executed with subtract the constant rotational offset, represented in the circle, from each average velocity vector. The complement results a translational combination of vectors in X_B and Y_B direction. A problem of this described inter-

[20] Such a FOE is shown e.g. in the middle of the raw optical flow in figure 4.18

pretation of the rotational component, is the missing information of the rotation direction. Consequently the direction of rotation has to be adopted from the yaw gyroscope sensor.

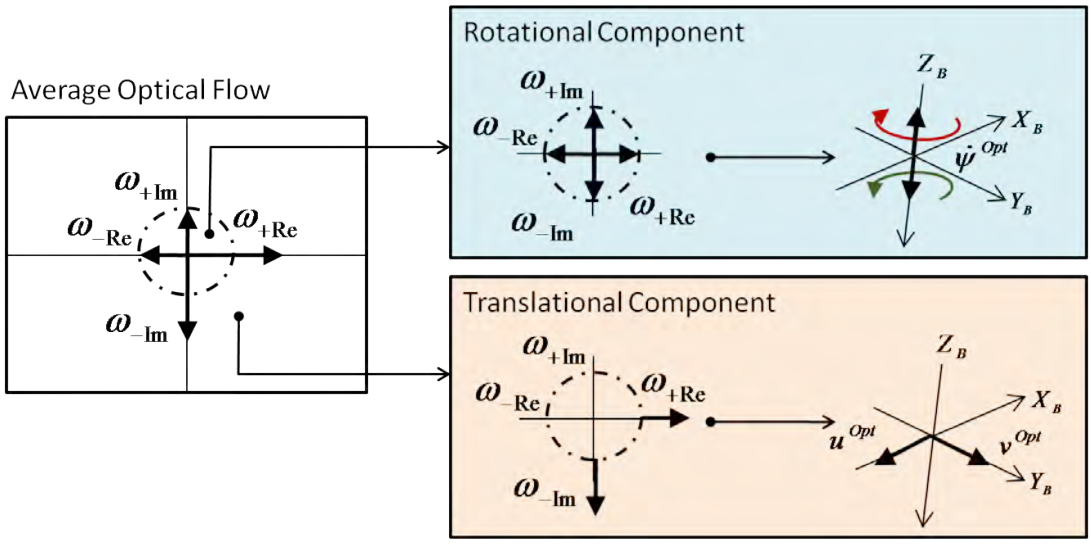


Figure 4.19: Determination of rotational and translational component with the Flow Number Calculation approach

Another approach to determine the desired translation and rotation velocities is to determine the Transformation Matrix (Tform) between two images in an image sequence. In this project here, the tracked features of both images are represented as two dimensional points. An algorithm, which estimates the movement of the points, called Random Sample Consensus (RANSAC) provides the determination of the Tform in each sample step. Furthermore this matrix contain information of rotation and translation which allows the determination of these values with a reverse calculation shown in figure 4.20. Combined with the known sample time of the image sequence, the determined angle and translation positions can be transformed to velocities. The important characteristic of the Tform calculation, is given with the determinable direction of the rotation movement.

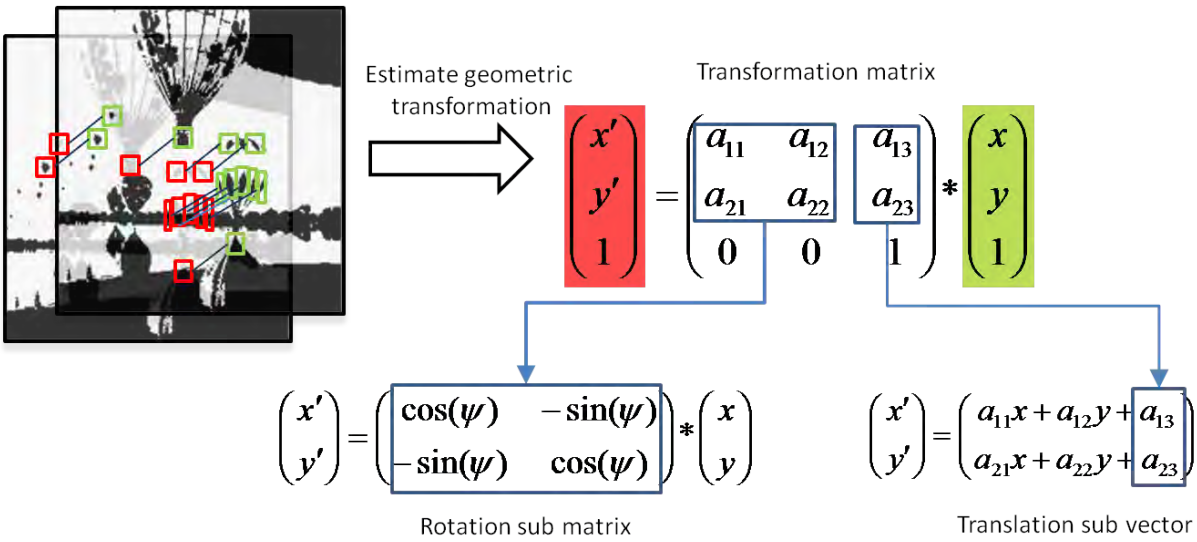


Figure 4.20: Determination of rotational and translational component with the Transformation Matrix approach

4.5 Optical Movement Detection Architectures

	MATLAB/Simulink-Native IP	OpenCV IP
Pros	Support of automated target code generation	Provides exact configuration interfaces
	Easy to configure and to maintain	Big amount of algorithms
	Easy to learn	Easier to unify to special implementations causes C++ object oriented paradigm
Cons	Limited amount of algorithms	Flexibility of configuration and maintenance is related programmed architecture
	Supports not detailed configuration	Does not support MDB paradigm. Code must be written manually
	Limited unification of algorithms possible	Difficult to learn. Complex architecture

5 Experimental Results and Analysis

5.1 Analysis of Image Processing

This section introduces the specific configured test environment and the corresponding results of the analysed topics, related to the motion detection. Furthermore the testing analysis is separated in three domains of variation, presented in figure 5.1. The basic idea thereby is to create a 2D-trajectory by using a stimuli as evaluation source of the results. This stimuli can be executed in three types of motion, presented in the stimuli domain. Basically the stimuli is given as rotational or translational velocity type, which is also provided as output value from the image processing. Additionally, to test the reaction behaviour of the a image processing configuration, it is important use velocity steps, or further, acceleration inputs. Such tests show how fast the changed value can be reached by the image processing and gives a measure for the time impact. Another domain which can be separated for the tests here is the algorithm and calculation domain. As visualised this segmentation can be structured as a tree with n nodes which contain the image processing algorithms and calculations as leaf nodes. As mentioned, in this project here the algorithm and calculation constellation is reduced to the two visualised combinations, for reasons of algorithm support of the used simulation framework.

Finally, the regarded aspects in relation to this project of the video source behaviour, are summarised in the video domain. To provide a realistic environment,

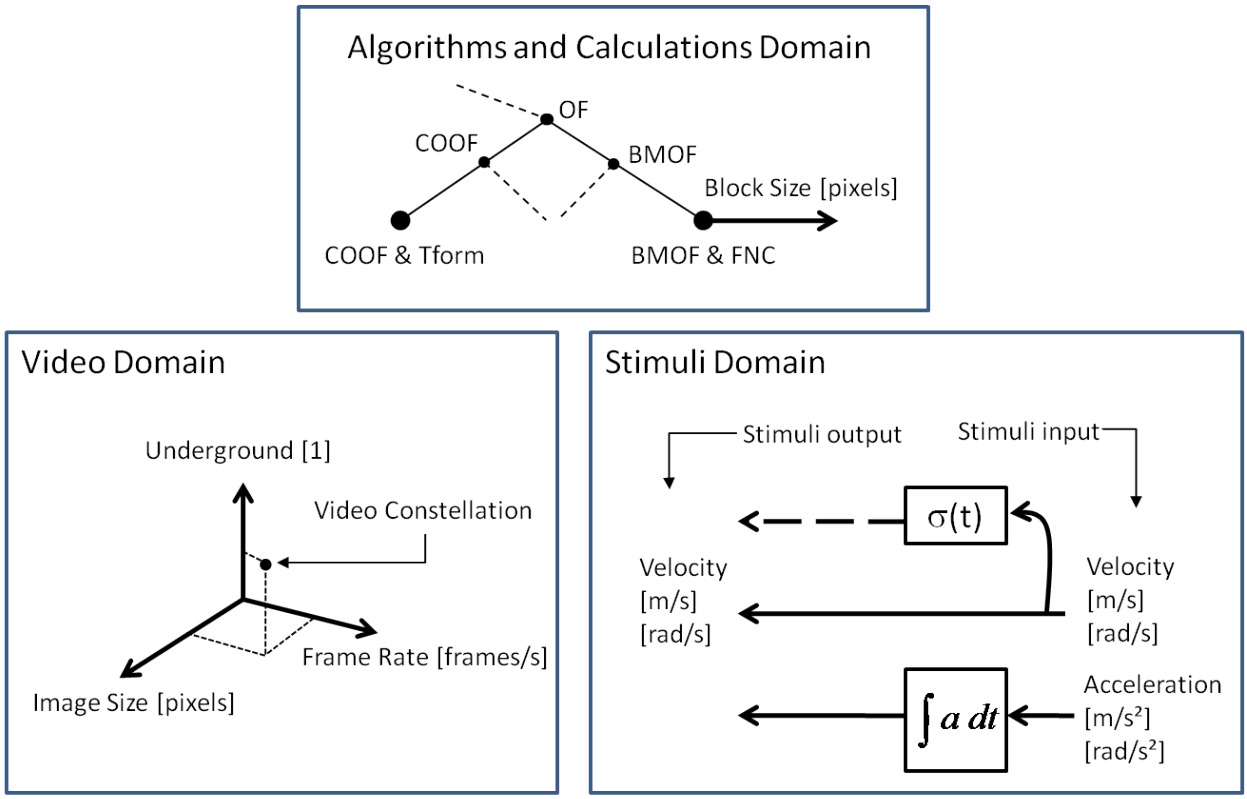


Figure 5.1: Image processing analysis domains in focus of this project

the chosen ranges of experimental analysis are derived from the data-sheets and publications [1], presented in **??**. Afterwards, the components and their corresponding decision criteria of the video domain are presented and discussed in the following points.

[1] See [28, p.35], [19, p.6], [5, p.22], [1], [17, p.4]

Camera Name	FPS	Connection	Image Size
firefly MV 1.3”	15	Wifi (802.11g)	640x480
Micron MT9V022 CMOS	60	general-purpose I/O	752x480
Panasonic BL-C131A	20	Wifi (802.11g)	320x240
uEye UI-122xLE	80	USB-Cable	752x480

Table 5.1: Camera examples derived from literature

- Frame Rate: The frame rate is related to the maximum velocity in the regarded DOF, which can be detected by the camera system. In relation to the image size, algorithm and calculation and the stimuli it can be determined which constellation to the Frames per Second (FPS) can provide satisfactory results. Generally the ideal constellation is, regarding the frame rate, to reduce as much as possible the processed images per second. By doing this, the calculations as well as the transmission complexity from quadrocopter to the base-station are reduced. The variations which will be executed here in the experimental analysis are 10, 20, 40, 60, 80 FPS.

- Image Size: Similar to the frame rate, the image size can also influence the maximum velocity which can be detected by the camera system. Important thereby is the behaviour of the objects, moving relative to the camera system. A big image window size allows to observe a bigger amount of points similarly. Furthermore, in a constellation of similar conditions, points are projected for a longer period to a big image window. In contrast to that, points stay for short time interval into the focus of a small image. So in such cases a higher image processing sample rate is desired to provide the same quality of results. In the other hand, if we also assume the same conditions of image composition, a bigger image window size requires a higher data size which also influences the transmission and the calculation of the

image processing. Based on the example cameras presented in **??**, the following image size variations in the corresponding experiments are 320x240, 640x480, 768x576, 800x600, 1024x768 [height x width].

- Underground: The underground variation is a special behaviour of experimental analysis related to this project here. Based on the fact that the presented algorithms of optical movement detection focus different aspects of the image, it can be assumed that the motion detection behaviour will be different by varying the underground. So the undergrounds presented here, focus two characteristics. As visualised in 5.2 the first underground (amorphous), provides amorphous groups of different contrasts. In difference to that the second (segmented amorphous) provides additionally segmentations like edges and corners. Finally the third (cornered) has mainly a cornered and edged structure. This variation of amorphous and corned structures provide a challenge to the algorithms which work with corner detection and block matching.

5.1.1 Aspects of Reproducibility and Reliability

This chapter introduces the methods, which were executed to proof the correct function of the image processing simulation. As first step, the sources which influence the test scenarios were analysed and evaluated. Thereby it is important to clarify if there elements inside these, which generate noisy or randomly values. In case of the real and synthetic video source this is not the case, because the real video never changes in the course of execution, and the synthetic video generates the same output with the corresponding same input. For proofing this reproducibility, and further the correctness of the synthetic video, two experiments

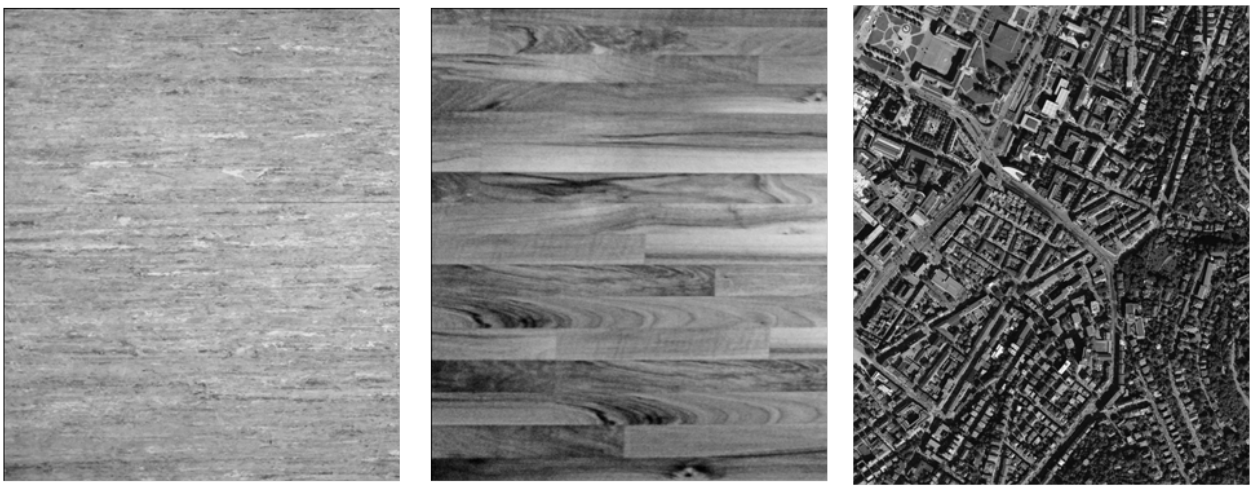


Figure 5.2: Different undergrounds with amorphous, segmented amorphous and cornered structure

were executed several times. These experiments are visualised in figure 5.3. As shown, an special underground for measuring the movement is installed for the rotational and translational test. Furthermore the movements $\Delta\psi$, Δx and Δy are tested to the corresponding simulation time Δt. The result can be compared with the configured stimuli, and it can be evaluated if the velocity input was realised correctly. These test cases executions show that the synthetic video environment is reliable and reproducible.

5.1.2 Test Scenarios and Expectations and Results

As presented in the figure 5.1, the amount of variance is enormous. So the optical movement detection algorithms and corresponding calculations are tested in several scenarios which focus a specific behaviour with a corresponding expectation of the result. The goal thereby is to provoke expected characteristics, or to demon-

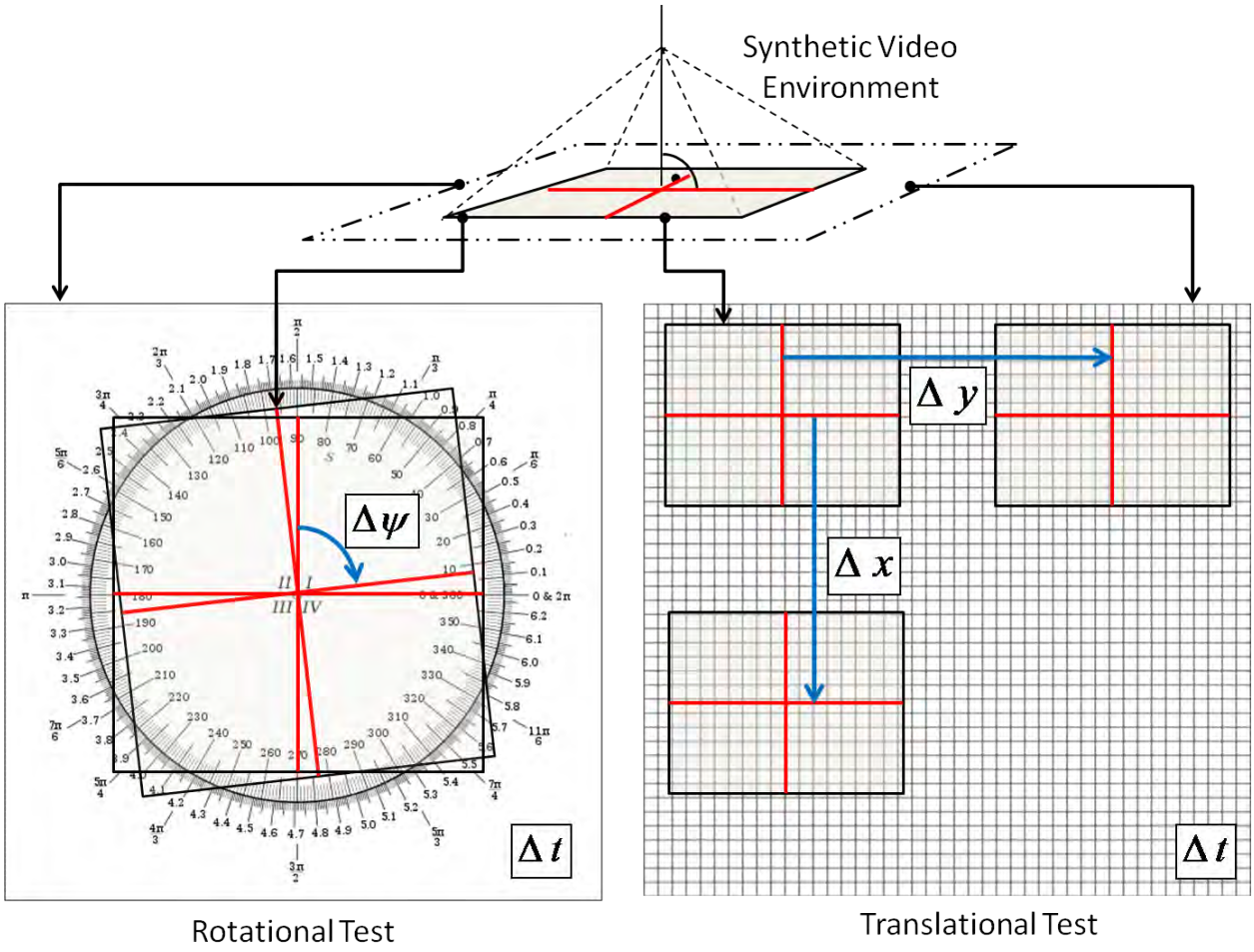


Figure 5.3: Image Processing Test Plan

strate that the expectation are not fulfilled by the result of the corresponding test scenario. For a better overview the test plan 5.4 shows which components of the related domains of variation. Each scenario is driven by a stimuli data which contains the tree mentioned components of the stimuli domain, presented in 5.1. Beside this each test scenario focus separately the rotational and translational behaviour.

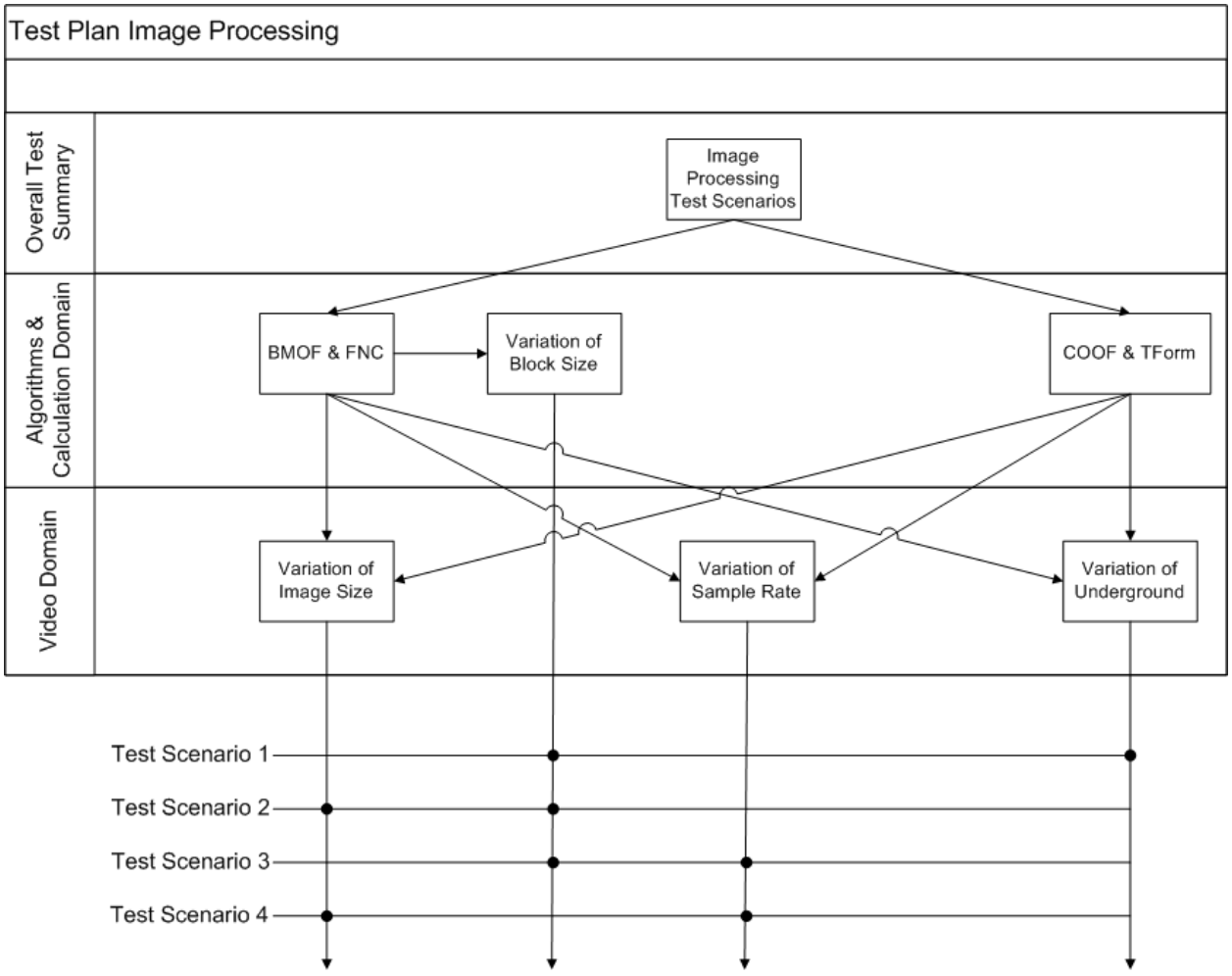


Figure 5.4: Image Processing Test Plan

Test Scenario 1: Variation of Underground

As mentioned, in the description of the underground item, a flight scenario can be influenced by the underlying underground image. So this test scenario focus the behaviour of the different algorithms in relation to the underground. To provide suitable results for investigation, the configurations of the used algorithms must also been varied. In this case it is interesting to vary the Block Matching Optical Flow (BMOF) algorithm's block size, which can have a impact to the result. This scenario bases on the expectation that the structure of the underground influences

the quality of movement detection in relation to the used algorithm. Furthermore as expected, the Corner Detection Optical Flow (COOF) algorithm should show the best results in the cornered underground, because the build in corner tracker can find a bigger amount of features. In the other hand, the BMOF should operate satisfactory in the amorphous structure. The reason for this expectation is that the similarity measure of blocks is not as hight as in the corned structure, because of the amorphous behaviour. This measure should influence rate of the correct matched blocks. Finally the segmented amorphous structure should expectably demonstrate that both algorithms work moderate on this structure, but not as good as in the expected more advantageous structure.

Results of Test Scenario 1

Starting with the translational analysis of test scenario 1, the result presented in 5.5 shows the characteristic of the COOF algorithm. The expected best case behaviour for this algorithm, is proved and is as expected the cornered structure. Interesting to see is that the segmented amorphous nearly shows the same results as the best case. The amorphous underground affects as expected some outlines (See arrow No.2 5.5), probably because the found feature amount is not big enough or some features raise errors because of the same structure. In the other hand this assumption could be the reason for the apparent better detection of the acceleration stimuli part with the cornered structure (See arrow No.4 5.5). Another unexpected behaviour, which is probably related with the sample time, is that the complete output curves react with a delay of 100ms (See arrow Nr.1 and Nr.3 5.5).

The results of the BMOF are visualised in 5.6. Thereby the unexpected but in-

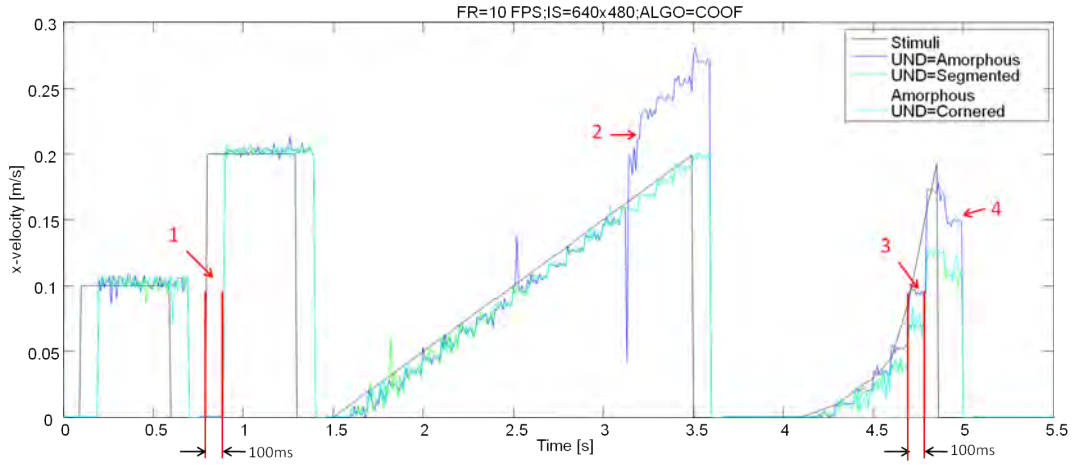


Figure 5.5: Result of Test Scenario 1: Translation analysis of COOF

teresting behaviour was found, that the BMOF algorithm reaches the limit of maximum velocity, in this special case $0.1m/s$, with equal conditions faster as the COOF algorithm. Furthermore the expected best suited underground is the amorphous structure, but just in the aspect of precision (See arrows No.1, 5.6). But this noise behaviour could also be an limitation of the simulation, caused by multiple calculations of the same sampled image sequence (See 5.1.3)(See arrow No.2 and No.3). The unexpected behaviour is that the BMOF algorithm has a better offset behaviour, in relation to its block size, with the cornered structure. This means that the BMOF algorithm does not normalise the detected velocities the blocks, to a relative level which affect the problem that a course grained block segmentation has a lower offset in the equal environment as a fine grained segmentation.

Afterwards the experimental results of the rotational behaviours show enormous differences in several aspects. First, the diagram (a) in figure 5.7 visualises the rotational COOF algorithm behaviour in a test with a maximum gain of $2 * s\pi rad/s$. As mentioned, the COOF algorithm can detect the rotational direction. This information is reflected in the sign of the detected velocity. As visualised, the negative

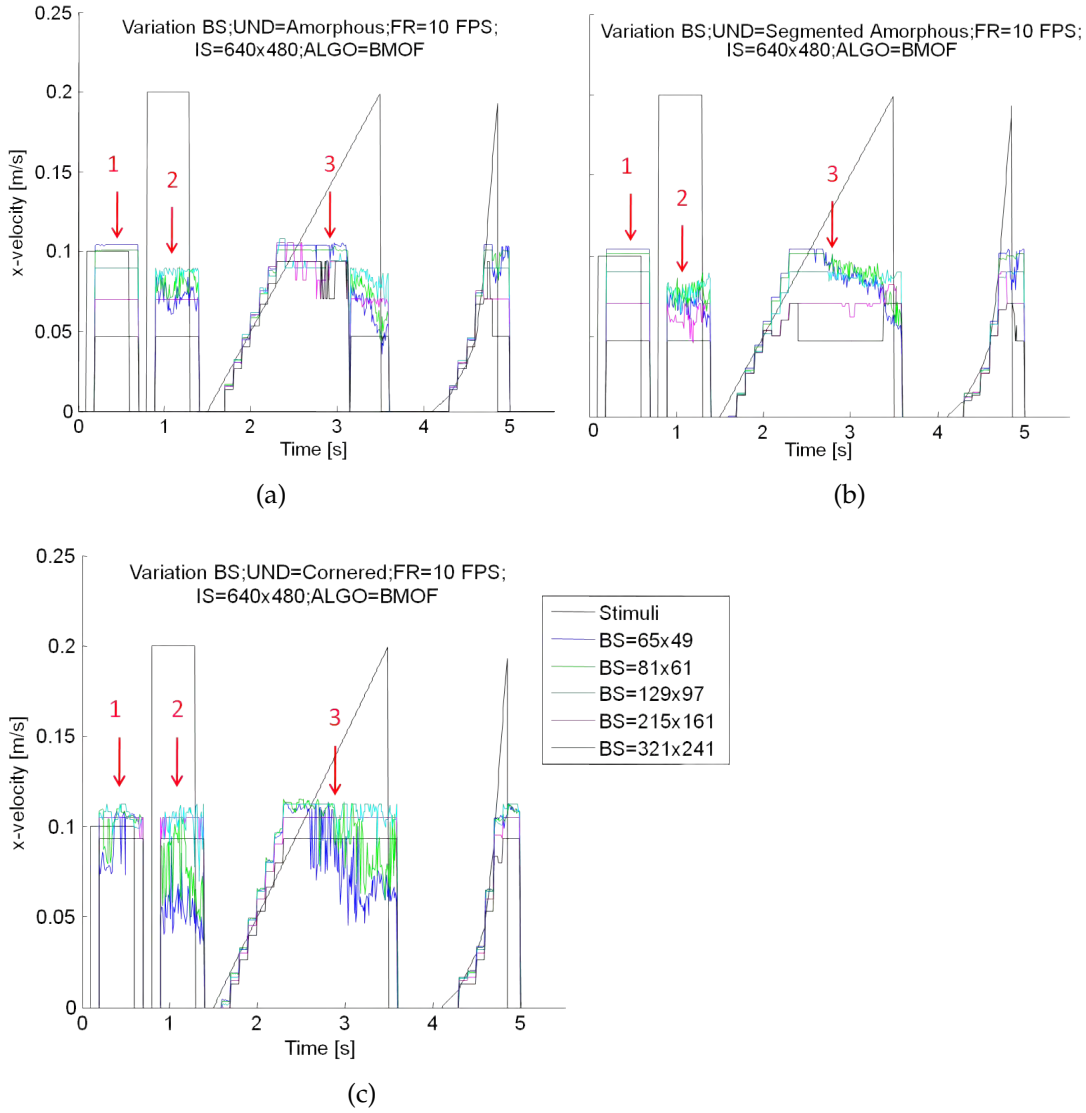
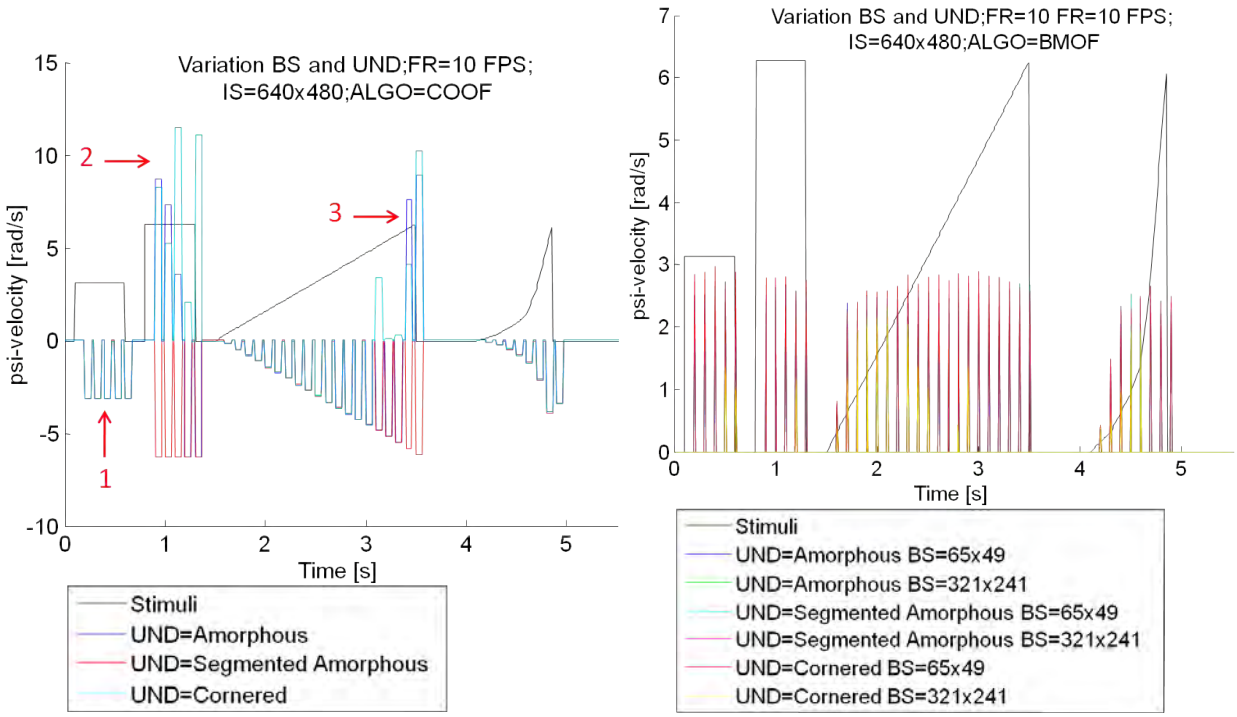


Figure 5.6: Result of Test Scenario 1: Translation analysis of BMOF

direction of the rotational behaviour shows a counter-clockwise rotation. As we can see, the first pulse, with a gain of π is nearly error-free detected (See arrows No.1 5.7). In contrast to that, the COOF algorithm shows problems with reaching the doubled gain of π (See arrows No.2 and No.3 5.7). These errors possibly based on the combination of a rotational movement and the low sample rate. Such effect is also discussed and presented in the stroboscopic torque measurement in chapter 4.1.2. Unexpected is in this case that, the COOF algorithm also show er-

rors with the amorphous and cornered structure, but provides the best matchings with the segmented amorphous structure. The reason could be a big amount of dissimilar forms, bases on the amorphous and cornered structure. The diagram (b) shows the result of the BMOF algorithm, which is under no circumstances useful. In this case it is obvious, that the is low sample rate to the corresponding hight rotational velocity affects a big amount of error matchings.



(a) (b)

Figure 5.7: Result of Test Scenario 1: Rotation analysis of BMOF and COOF

Because of the poor rotational behaviour of the BMOF, the rotational experiment of this algorithm was executed with a sample rate of 80 FPS. The result is the first satisfactory configuration for the rotational behaviour of this algorithm. Returning to the focus of this test scenario, the underground structure which showed the best results is the segmented amorphous with the fine grained configuration of the blocks (See arrow No.1 5.8). This behaviour is argumentative, because it

behaves like the COOF algorithm. Possibly the mentioned reason of the highest dissimilarity of forms can also be the characteristic of best matches.

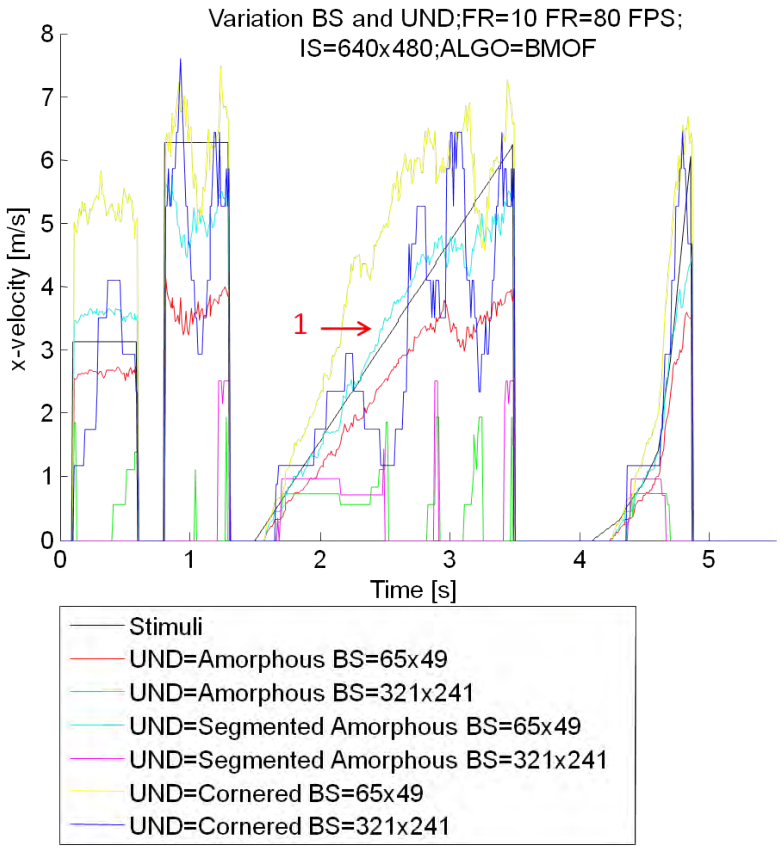


Figure 5.8: Result of Test Scenario 1: Rotation analysis of BMOF with conformed frame rate

Test Scenario 2: Variation of Image Size

The image size of the captured images is related to the data size these and further on the transmission time to the base station. So this test sequence focus the variation of usual image sizes, presented in the introduction of this value and based on the presented cameras in **??**. As is well known from the first test scenario, the block size of the BMOF also is varied in this test case to analyse the corresponding relation of block and image size. The underground which was chosen

in this scenario, is the segmented amorphous structure, which was not changed over the test execution. The goal and expectation of this test case is to demonstrate the relation of the image size and the maximum translational or rotational speed which can be detected. Also interesting is to investigate, if the precision of movement detection is related to the image size. This expectation bases on the assumption that more points can be captured in a bigger image and this leads to a more exact determination of movement. Another interesting characteristic is, which algorithm can provide a better performance in aspects of precision and maximum speed, with the same image size. The expectation is that the COOF algorithm will show better results, and will not be related as much, as the BMOF algorithm, to the image size. This assumption bases on the fact that COOF takes similarly all tracked features into account in each step, and is not dependent on a limited search area as is the case with the BMOF algorithm.

Test Scenario 3: Variation of Sample Rate

The Sample Rate variation test scenario, based on MATLABS/Simulink's time variant simulation of dynamic systems, provides the perspective to analyse the algorithm behaviour in relation to the presented rates in the introduction. As mentioned, similar to the image size the frame rate also has impact on the load factor of the image processing and transmission spectrum. So in this scenario the introduced algorithms should be investigated with the result to determine the limits of maximum speed operation, and to figure out which algorithm, with corresponding configuration, provides the best result related to the same frame rate. The expected result of this scenario is, that both algorithms will reach a limit, but the COOF will work with with higher velocity conditions. Based on

the mentioned expectation in the test scenario 2, the limited search region of the BMOF will increase the error possibility with ascending speed. In the other hand the BMOF algorithm should show a more precise processing in lower speed and frame rate, because with increasing successful block matches, the possibility of error matches is decreased. This expected characteristic could be demonstrated with jigsaw puzzle example, in which also the possibility to find the right piece in a region increases in relation to the amount of already located pieces.

Test Scenario 4: Variation of Image Size reciprocal to the Sample Rate

5.1.3 Limits of simulation environment

5.2 Analysis of Control behaviour

6 Conclusions and Further Work

7 Appendix

7.1 Moment of inertia

The physical modelling and simulation of a rigid body, needs to be described with the dynamic behaviour of that. So the quadrocopter UAV can be described with the EOM described in chapter 3.1.2 in relation to the moment inertias which describe the resistance of the UAV rigid body against changes in rotational directions. Such moment inertia is described as a $3x3$ matrix, also called tensor, for a rigid body with 6 DOF. The assumption that the B-frame originates in the point of mass of the quadrocopter [1], the matrix is simplified to a matrix with diagonal entries (Ixx, Iyy, Izz)[2].

$$I = \begin{pmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{YX} & I_{YY} & I_{YZ} \\ I_{ZX} & I_{ZY} & I_{ZZ} \end{pmatrix} = \begin{pmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{pmatrix} \tag{7.1}$$

7.2 Mathematical description of distortions

Distortions can occur if the distance of image points to the centre of projection varies or if the centre of projection is not in the centre of the image. These kinds

[1] See chapter 3.2

[2] The derivation of the moment inertia and the corresponding pictures are derived from [16, pp.24 -33 Determination of the moment of inertia]

of distortions are defined as radial x_dr, y_dr and decentring distortions x_dd, y_dd, and represent the components of the complete distortion of images x_d, y_d.

The following formulas[3] describe the possible distortion of images and show that the complexity of an image equalisation is proportional to the image points e.g. the image dimensions. Thereby the values x_n, y_n describe a point of an image which has to been equalized. The distance radius to the centre of the image 7.3 is a important component and is used in combination of the distortion coefficients $k_{cn} : n \in \{1, 2, 3, 4, 5\}$ to eliminate the mentioned components of distortion.

$$\begin{aligned} x_d &= x_{dr} + x_{dd} \\ y_d &= y_{dr} + y_{dd} \end{aligned} \tag{7.2}$$

$$r = \sqrt{(x_n^2 + y_n^2)} \tag{7.3}$$

$$\begin{aligned} x_{dr} &= (1 + k_{c1} * r^2 + k_{c2} * r^4 + k_{c5} * r^6) * x_n \\ y_{dr} &= (1 + k_{c1} * r^2 + k_{c2} * r^4 + k_{c5} * r^6) * y_n \end{aligned} \tag{7.4}$$

$$\begin{aligned} x_{dd} &= 2 * kc3x_n * y_n + k_{c4} * (r^2 + 2 * x_n^2) \\ y_{dd} &= k_{c3} * (r^2 + 2 * y_n^2) + 2 * k_{c4} * x_n * y_n \end{aligned} \tag{7.5}$$

These calculations can be executed initial for the camera calibration, and further for the real-time equalization for the trajectory movement of the camera. Popular calibration processes transform the distortions to a value in which they can

[3]This formulas are derived from the formulas presented in [18, p.30 Mathematical description of distortions]

be considered as equalized. Such a calibration process is described by Tsai et al. [Tsa05, Camera Calibration by Tsai] and extended by Zhang [Zha99, Visual Servoing with Dynamics], and describes a iterative way for the determination of the coefficients k_{cn}. Thereby a squared image is used as reference for the camera and the algorithm, described in figure 7.1[4], is executed iteratively. Real-time equalization processes mostly are implemented with an array of distortion coefficients and can effective determine these with a lookup operation[5].

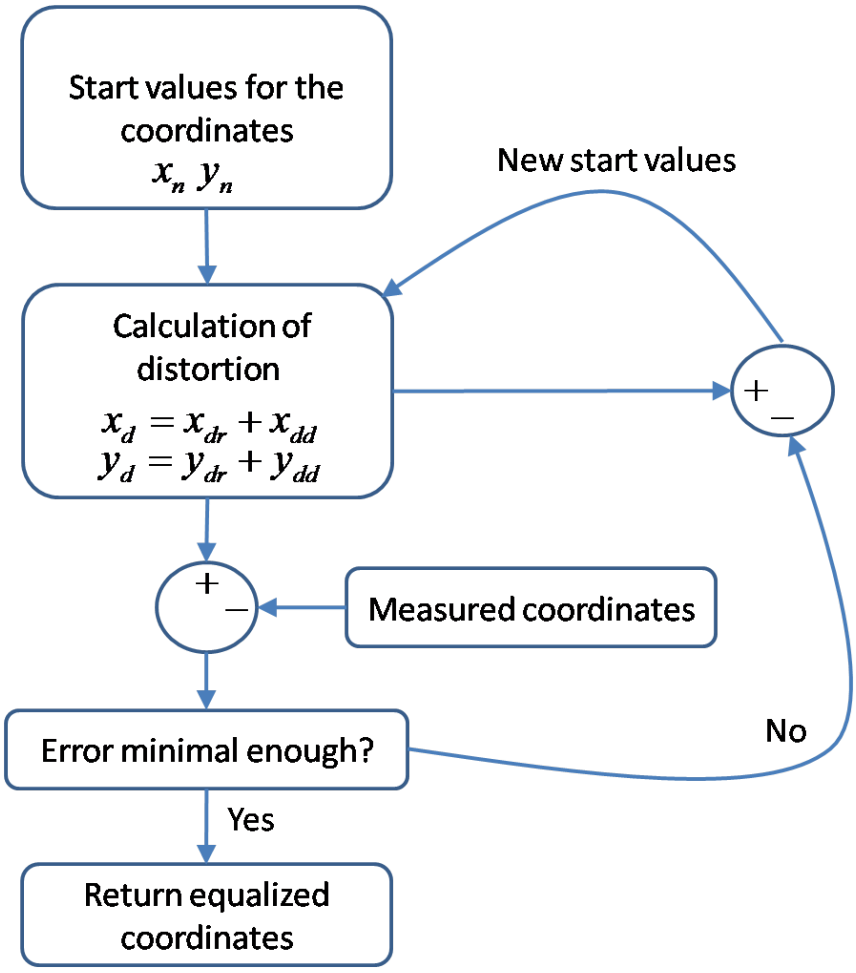


Figure 7.1: Equalization Algorithm of distorted images

[4]This picture extends the scheme picture presented in [18, p.32 Equalization of a captured image]
[5]See, Distortion Correction with FPGA [Gri03, pp. System Design]

Test Scenario	FPS	Image Size	Algorithm	Underground
1	Translation=10 FPS, Rotation=80FPS	640x480	COOF, BMOF(BS Variation)	Amorphous, Segmented Amorphous, Cornered
2	Translation=10 FPS, Rotation=80FPS	320x240, 640x480, 768x576, 800x600, 1024x768	COOF, BMOF(BS Variation)	Segmented Amorphous
3	10, 20, 40, 60, 80 FPS	640x480	COOF, BMOF(BS Variation)	Segmented Amorphous
4	10, 20, 40, 60, 80 FPS	320x240, 640x480, 768x576, 800x600, 1024x768	COOF, BMOF(81x61)	Segmented Amorphous

Table 7.1: Image Processing Test Scenario Configuration Values

7.3 Characteristics of Sensor Noise

The captured sensor noise of the IMU can be characterized with the sample variance of the readout random sample. Thereby the Bias-corrected Sample Variance(S^2_{N-1}) (See 7.6) can be used to determine the variance of a random set of values. This variance can be used as characteristic for noise generators, as well it was used in focus of this projects' simulation.

$$S^2_{N-1} = \frac{1}{N-1} * \sum_{i=1}^{N} (x_i - \bar{x}^2) \tag{7.6}$$

7.4 Image Processing Test Scenario Configuration

Values

The derived S^2_{N-1} values are presented in table 7.2. These values are derived from a list of measures captured over 11 seconds in an interval of 100 milliseconds.

$S^2_{N-1}(\ddot{x}_e)$	$S^2_{N-1}(\ddot{y}_e)$	$S^2_{N-1}(\ddot{z}_e)$	$S^2_{N-1}(p)$ *Roll*	$S^2_{N-1}(q)$ *Pitch*	$S^2_{N-1}(r)$ *Yaw*
2.178 m/s^2	0.576 m/s^2	6.860 m/s^2	0.003 rad/s	0.001 rad/s	0.001 rad/s

Table 7.2: Derived Bias-corrected Sample Variances of sensor captured values

8 References

[1] *MT9V022: 1/3-Inch Wide-VGA Digital Image Sensor*.

[2] Model Based Design, The MathWorks, Inc. http://www.mathworks.co.uk/model-based-design/, September 2010.

[3] Antoine Beyeler, Jean-Christophe Zufferey, Dario Floreano. optiPilot: control of take-off and landing using optic flow. Technical report, Laboratory of Intelligent Systems, École Polytechnique Fédérale de Lausanne (EPFL), 2009.

[4] Barry Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):pp.61–72, 1988.

[5] Beau J. Tippetts. Real-Time Implementation of Vision Algorithms For Control, Stabilization, and Target Tracking, for a Hovering Micro-UAV. Master's thesis, Department of Electrical and Computer Engineering, Brigham Young University, August 2008.

[6] Bernhard Kleiner. Kameragestuetztes inertiales Navigations-system mit Sensorfusion. Technical report, Fraunhofer Institute for Manufacturing Engineering and Automation IPA, 2009.

[7] Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

[8] Daniel Mellinger, Nathan Michael and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. *International Symposium on Experimental Robotics, Delhi, India*, December 2010.

[9] David Morin. *Introduction to classical mechanics: with problems and solutions*. Cambridge University Press, 2007. ISBN 978-0-521-87622-3 First Published Edition 2008.

[10] Douglas Eastman, Paul Lambrechts and Arkadiy Turevskiy. Design and Verification of Motion Control Algorithms Using Simulation. *The MathWorks, Inc.*, 2009.

[11] Erdinc Altug, James P. Ostrowski, Robert Mahony. Control of a Quadrotor Helicopter Using Visual Feedback. *IEEE International Conference of Robotics and Automation, Washington, DC* , 2002.

[12] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, First Edition edition, 2008.

[13] George Koharchik. Image Processing with OpenGL and Shaders. http://www.linuxjournal.com/content/image-processing-opengl-and-shaders, February 2011.

[14] Jean-Bernard Hayet. Exploration strategies for simultaneous localization and mapping. http://www.cimat.mx/ jbhayet/temas.php, February 2011.

[15] J.L. Barron, D.J. Fleet and S.S. Beauchemin. Performance of Optical Flow Techniques. Technical report, Dept. of Computer Science, University of Western Ontario, Queen's University, 1994.

[16] Joachim Schwab, Sreekanth Sundaresh, Nina Vetlugina, Marc Weber, Joerg Fiedrich and Dionysios Satikidis. Project Quadrocopter, Automotive Systems Master Software Based Automotive Systems ASM2-SB / SS 2010 Documentation. Technical report, University of Applied Sciences Esslingen, Department of Information Technology, June 2010.

[17] John Stowers, Andrew Bainbridge-Smith, Michael Hayes and Steven Mills. Optical Flow for Heading Estimation of a Quadrotor Helicopter. Technical report, University of Canterbury, New Zealand, 2009.

[18] Martin Fach. *Navigation und RAhrensegelung eines Luftschiffes mittels optischer, inertialer und GPS Sensoren*. PhD thesis, Institut für Flugmechanik und Flugregelung, University of Stuttgart, 2008.

[19] Michael Bloesch, Stephan Weiss, Davide Scaramuzza and Roland Siegwart. Vision Based MAV Navigation in Unknown and Unstructured Environments. Technical report, Autonomous Systems Lab, ETH Zuerich, 2010.

[20] Nirupama Bulusu, John Heidemann and Deborah Estrin. GPS-less Low Cost Outdoor Localization For Very Small Devices. Technical report, University of Southern California / Information Sciences Institute, 2002.

[21] Pratap Rudra. *Getting started with MATLAB 7 // a quick introduction for scientists and engineers*. Oxford University Press, Inc., 2006. ISBN 978-0-19-517937-8.

[22] Reinhard Keller, Hermann Kull, Walter Lindermeier, Werner Zimmermann. System Dynamics, Modelling, Simulation, Control engineering, July 2007. VZ1.2.

[23] Richard C. Dorf, Robert H. Bishop. *Modern Control Systems*. Prendice-Hall, Inc., ninth edition edition, 2001.

[24] Roger R. Pressman. *Software Engineering, A Practitioner's Approach*. McGraw-Hill, Inc., fifth edition edition, 2001.

[25] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer Science+Business Media, LLC, 2008. ISBN 978-0-387-74314-1.

[26] Samir Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2007.

[27] Scott M. Ettinger, Michael C. Nechyba, Peter G. Ifju and Martin Waszak. Vision-Guided Flight Stability and Control for Micro Air Vehicles. Technical report, Intel Corporation, University of Florida, NASA Langley Research Center, March 2004.

[28] Spencer Greg Ahrens. Vision-Based Guidance and Control of a Hovering Vehicle in Unknown Environments. Master's thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 2008.

[29] Steven Kelly. Domain-Specific Modeling 76 cases of MDD that works, 2009.

[30] Steven Kelly and Juha-Pekka Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. John Wiley & Sons, Inc, 2008. ISBN 978-0-470-03666-2.

[31] Sven Lange, Niko Sünderhauf, and Peter Protzel. Autonomous Landing for a Multirotor UAV Using Vision. *Intl. Conf. on SIMULATION, MODELING*

and PROGRAMMING for AUTONOMOUS ROBOTS, Venice, Italy, November 2008.

[32] Sven Lange, Niko Sünderhauf, Peter Protzel. A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments. Technical report, Department of Electrical Engineering and Information Technology, Chemnitz University of Technology, 2009.

[33] Tommaso Bresciani. Modelling, Identification and Control of a Quadrotor Helicopter. Master's thesis, Department of Automatic Control, Lund University, October 2008.

[34] Zhaoyi Wei. *REAL-TIME OPTICAL FLOW SENSOR DESIGN AND ITS APPLICATION ON OBSTACLE DETECTION*. PhD thesis, Department of Electrical and Computer Engineering, Brigham Young University, 2009.

9 Bibliography

[ADX] *ADXRS610*.

[Aud01] Audsley N. C., Burns A., Richardson M. F., Wellings A. J. HARD REAL-TIME SCHEDULING: THE DEADLINE-MONOTONIC APPROACH. Technical report, Department of Computer Science, University of York, 2001.

[Bea08] Beau J. Tippetts. Real-Time Implementation of Vision Algorithms For Control, Stabilization, and Target Tracking, for a Hovering Micro-UAV. Master's thesis, Department of Electrical and Computer Engineering, Brigham Young University, August 2008.

[Ber80] Berthold K. P. Horn and Brian G. Schunck. Determining Optical Flow. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1980.

[BLC] *Brushless Controler BL-Ctrl V1.2*.

[Bri07] Brian Williams, Georg Klein and Ian Reid. Real-Time SLAM Relocalisation. Technical report, Department of Engineering Science, University of Oxford, 2007.

[Bru81] Bruce D. Lucas, Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, pages pp.121–130, 1981.

[Dan07] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, Daniela Rus. Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz. *IEEE International Conference of Robotics and Automation, Roma, Italy* , April 2007.

[Dav71] David G. Luenberger. An Introduction to Observers. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. AC-16, NO. 6*, December 1971.

[DP73] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, pages 112–122, October 1973.

[Erd02] Erdinc Altug, James P. Ostrowski, Robert Mahony. Control of a Quadrotor Helicopter Using Visual Feedback. *IEEE International Conference of Robotics and Automation, Washington, DC* , 2002.

[Erd03] Erdinc¸ Altug, James P. Ostrowski, Camillo J. Taylor. Quadrotor Control Using Dual Camera Visual Feedback. *IEEE International Conference on Robotics and Automation, Taipei, Taiwan*, 2003.

[Gar08] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, First Edition edition, 2008.

[Gri03] Gribbon K.T., Johnston C.T. and Bailey D.G. A Real-time FPGA Implementation of a Barrel Distortion Correction Algorithm with Bilinear Interpolation. Technical report, Institute of Information Sciences and Technology, Massey University, Palmerston North, New Zealand, 2003.

[HCS] *MC9S12XDP512 Data Sheet, HCS12X Microcontrollers, Rev. 2.17*.

[HSE10] Hochschule Esslingen University of Applied Sciences. http://www.hs-esslingen.de, September 2010.

[Joa10] Joachim Schwab, Sreekanth Sundaresh, Nina Vetlugina, Marc Weber, Joerg Fiedrich and Dionysios Satikidis. Project Quadrocopter, Automotive Systems Master Software Based Automotive Systems ASM2-SB / SS 2010 Documentation. Technical report, University of Applied Sciences Esslingen, Department of Information Technology, June 2010.

[Joh86] John Canny. A Computational Approach to Edge Detection. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6*, November 1986.

[Joh09] John Stowers, Andrew Bainbridge-Smith, Michael Hayes and Steven Mills. Optical Flow for Heading Estimation of a Quadrotor Helicopter. Technical report, University of Canterbury, New Zealand, 2009.

[Jue09] Juergen Eckert, Falko Dressler and Reinhard German. An Indoor Localization Framework for Four-rotor Flying Robots Using Low-power Sensor Nodes. Technical report, University of Erlangen Dept. of Computer Science, February 2009.

[Kar10] Karl E. Wenzel, Andreas Masselli and Andreas Zell. Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. Technical report, Department of Computer Science, University of Tuebingen, 2010.

[LIS] *LIS3LV02DQ, MEMS INERTIAL SENSOR*.

[Mar08] Martin Fach. *Navigation und RAhrensegelung eines Luftschiffes mittels optischer, inertialer und GPS Sensoren*. PhD thesis, Institut für Flugmechanik und Flugregelung, University of Stuttgart, 2008.

[Mat11a] Matlab, The MathWorks, Inc. http://www.mathworks.com/products/matlab/, March 2011.

[Mat11b] The MathWorks, Inc. http://www.mathworks.com/, March 2011.

[Mic10] Michael Bloesch, Stephan Weiss, Davide Scaramuzza and Roland Siegwart. Vision Based MAV Navigation in Unknown and Unstructured Environments. Technical report, Autonomous Systems Lab, ETH Zuerich, 2010.

[Nat10] Nathan Michael, Daniel Mellinger, Quentin Lindsey and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. Technical report, GRASP Laboratory, University of Pennsylvania, 2010.

[Naw10] Nawarat Termtanasombat. Development of Hardware-in-the-Loop Facility for Optical Navigation with the Study of Insect Inspired Algorithm for Spacecraft Landing. Master's thesis, Department of Space Science, Lulea University of Technology, 2010.

[Ope10] OpenCV 2.0 C Reference. http://opencv.willowgarage.com/, 2010.

[Ped05] Pedro Castillo, Rogelio Lozano and Alejandro Dzul. Stabilization of a Mini Rotorcraft with Four Rotors. *IEEE Control Systems Magazine*, December 2005.

[Rei07] Reinhard Keller, Hermann Kull, Walter Lindermeier, Werner Zimmermann. System Dynamics, Modelling, Simulation, Control engineering, July 2007. VZ1.2.

[Ric01] Richard C. Dorf, Robert H. Bishop. *Modern Control Systems*. Prendice-Hall, Inc., ninth edition edition, 2001.

[Rob10] Robert Czech and Dionysios Satikidis. Project Quadrocopter, Analysis and development of control algorithms. Technical report, University of Applied Sciences Esslingen, Department of Information Technology, 2010.

[Sam04] Samir Bouabdalla, Pierpaolo Murrieri, Roland Siegwart. Design and Control of an Indoor Micro Quadrotor. Technical report, Autonomous Systems Laboratory Swiss Federal Institute of Technology, University of Pisa, 2004.

[Sam07] Samir Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2007.

[Seb08] Sebastian Blumenthal, Dirk Holz, Thorsten Linder, Peter Molitor, Hartmut Surmann and Viatcheslav Tretyakov. Teleoperated Visual Inspection and Surveillance with Unmanned Ground and Aerial Vehicles. Technical report, Department of Computer Science, University of Applied Sciences Bonn-Rhein-Sieg, Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), 2008.

[Sim11] Simulink, The MathWorks, Inc. http://www.mathworks.com/products/simulink/

March 2011.

[Sla09] Slawomir Grzonka, Giorgio Grisetti, Wolfram Burgard. Towards a Navigation System for Autonomous Indoor Flying. Technical report, Autonomous Systems Lab, Department of Computer Science, University of Freiburg, 2009.

[ST 05] ST Microelectronics. MEMS Inertial Sensor LIS3LV02DQ Rev 1 Datasheet, October 2005.

[Sve08] Sven Lange, Niko Sünderhauf, and Peter Protzel. Autonomous Landing for a Multirotor UAV Using Vision. *Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, Venice, Italy*, November 2008.

[Sve09] Sven Lange, Niko Sünderhauf, Peter Protzel. A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments. Technical report, Department of Electrical Engineering and Information Technology, Chemnitz University of Technology, 2009.

[Tom08] Tommaso Bresciani. Modelling, Identification and Control of a Quadrotor Helicopter. Master's thesis, Department of Automatic Control, Lund University, October 2008.

[Tsa05] Tsai R. Y. and Lenz R. K. . A new technique for fully autonomous and efficient 3D robotics hand-eye calibration. *Robotics and Automation. IEEE Transactions on Robotics and Automation, Vol. 5, No. 3, June 1989*, December 2005.

[Ved08] Vedran Sikiric. Control of Quadrocopter. Master's thesis, Royal Institute of Technology Sweden, School of Computer Science and Communication, 2008.

[Wei11] Weisstein, Eric W. Heaviside Step Function From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/HeavisideStepFunction.html , 2011.

[XBe] *XBee*.

[Zha99] Zhang H. and Ostrowsi J. P. Visual Servoing with Dynamics: Control of an Unmanned Blimb. Technical report, General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory; University of Pennsylvania, 1999.