CHALMERS

A Model-Based Approach to Computer Vision and Automatic Control using MATLAB Simulink for an Autonomous Indoor Multirotor UAV

Master of Science Thesis

Niklas Ohlsson
Martin Ståhl

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2013
Report No. EX014/2013

THESIS FOR THE DEGREE OF MASTER IN SCIENCE

A Model-Based Approach to Computer Vision and Automatic Control using MATLAB Simulink for an Autonomous Indoor Multirotor UAV

Niklas Ohlsson
Martin Ståhl

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2013

A Model-Based Approach to Computer Vision and Automatic Control using Matlab Simulink for an Autonomous Indoor Multirotor UAV

Niklas Ohlsson
Martin Ståhl

Master of Science Thesis in collaboration with Combine Control Systems and MathWorks
Report No. EX014/2013
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
The prototype multirotor with the letter B attached to its back, enabling tracking of its true position during flight using a ceiling mounted camera

Chalmers Reproservice
Göteborg, Sweden 2013

A Model-Based Approach to Computer Vision and Automatic Control using Matlab Simulink for an Autonomous Indoor Multirotor UAV

Niklas Ohlsson and Martin Ståhl
Department of Signals and Systems
Chalmers University of Technology

Abstract

A prototype autonomous UAV platform featuring computer vision based navigation for use in GPS denied environments is presented. The UAV is based on a hexacopter platform from 3DR Robotics which has been equipped with a PandaBoard ES single-board computer and a downward facing webcam. Position estimation is performed using template and feature point matching image analysis techniques and object recognition has been implemented using invariant moment descriptors. Position control has been achieved using cascaded PD control, generating attitude setpoints sent to the hexacopter on-board computer over Ethernet.

Image analysis, control and decision making algorithms have been developed using model-based design techniques with automatic code generation in MATLAB Simulink. A multirotor model has been obtained with system identification methods and a camera model and emulator have been developed and used to emulate a camera video feed for image analysis algorithms development and verification. A 3D-visualization environment has been developed and used for assessment of the simulated system performance and behavior. Model accuracy is considered high, image analysis and control algorithm parameters tuned in simulation give similar flight behavior during actual test flights.

The UAV prototype is capable of limited time hovering above a play mat floor surface. Insufficient hexacopter altitude and yaw control performance achieved by the hexacopter computer does, however, affect position estimation in a negative way and arguably making it perform unsatisfactory. The template matching position estimation technique is functional but image feature point matching methods should be considered in future development for improved position estimation robustness to hexacopter yaw and altitude change.

KEYWORDS: Autonomous, Multirotor, UAV, Quadcopter, Hexacopter, Image Analysis, Position Estimation, Template Matching, Feature Point Matching, Model-Based Design, Matlab, Simulink, Automatic Control, Computer Vision, Object Recognition, System Identification, Code Generation, PandaBoard, ArduCopter, Camera Model, GNC, Guidance, Navigation, Control, Simulation

Sammanfattning

En autonom UAV-prototyp med bildanalysbaserad navigering för användning i miljöer utan GPS-täckning presenteras. UAV:n baseras på en hexakopterplattform från 3DR Robotics som har utrustas med en PandaBoard ES enkortsdator och en nedåtriktad webbkamera. Positionsestimering görs med hjälp av template matching och feature point matching bildanalysmetoder och objektigenkänning har implementerats med hjälp av invarianta momentdeskriptorer. Positionsreglering har uppnåtts med hjälp av kaskadkopplade PD-regulatorer vilka genererar vinkelbörvärden som skickas till hexakopterns dator med hjälp av Ethernet.

Bildanalys-, reglering- och autonomialgoritmer har utvecklats med hjälp av modellbaserade utvecklingsmetoder och automatisk kodgenerering i MATLAB Simulink. En multirotormodell har tagits fram med hjälp av systemidentifieringsmetoder och en kameramodell och simulator har utvecklats och använts för att emulera en kameravy för användning vid utveckling och verifiering av bildanalysalgoritmer. En 3D-visualiseringsmiljö har utvecklats och använts för utvärdering av systemprestanda och beteende under simulering. Modellens noggrannhet anses vara hög, bilanalys- och reglerparametrar kalibrerade i simulering ger snarlikt beteende under provflygning och simulering.

UAV-prototypen är kapabel till tidsbegränsad hovring ovanför en lekmatta. Otillräcklig bäring- och höjdreglering, utförd av hexakopterns dator, påverkar dock positionsestimeringen negativt och bidrar till dess ringa prestanda. Template matching är en funktionell metod för positionsestimering men feature point matchingmetoder bör tas i beaktning vid framtida vidareutveckling för förbättrad robusthet och tålighet mot ändring av bäring och höjd.

Preface

This master's thesis was conducted during the winter and spring of 2012–2013 at Combine Control Systems AB in Lund and Gothenburg, Sweden in conjunction with Chalmers University of Technology, department of Signals and Systems, division of Automatic Control, Automation and Mechatronics in Gothenburg, Sweden.

We would like to send thanks to our supervisors Erik Silfverberg and Simon Yngve at Combine Control Systems for their enthusiastic support and shown interest throughout the project. We would also like to thank our examiner at Chalmers, docent Torsten Wik, for his guidance and help. Special thanks goes out to Daniel Aronsson at Mathworks for weekly project meetings, support and help with software trial licenses throughout the duration of the project.

It has been a privilege for us to work with a startup project such as this and we would like to wish our successors good luck with their improvements and further development.

Watch us present this thesis at Chalmers here:
http://www.youtube.com/watch?v=ofLQw3X9-MU.

Niklas and Martin
Gothenburg, June 2013

Contents

List of Figures

List of Tables

1 Introduction

Model-based control systems design is a concept where mathematical process models are used to simulate and verify system performance before building physical prototypes. Model-based software design simplifies conventional development using an intuitive block diagram environment and automatic code generation. It can be argued that this method reduces coding errors and the need for programming skills, thus enabling the engineer to focus on his or her area of expertise. It is also said to reduce development time (Fleischer et al., 2009). Model-based development with code generation is claimed to be a promising approach when developing image analysis algorithms towards embedded platforms (Doblander et al., 2005).

MathWorks recently added native support for code generation in Simulink for small single-board computers powerful enough to perform advanced image analysis algorithms. These platforms are available at consumer price levels and no extra code generating MATLAB toolboxes are required.

1.1 Background

We are becoming used to seeing autonomous vehicles operate on land, at sea and in the air. Search and rescue operations are examples of new applications for unmanned aerial vehicles (UAV) where they help to, for example, locate missing people (Waharte and Trigoni, 2010). Operating outdoors, these UAV rely on GPS for positioning. However, research is being conducted on alternative positioning methods (Leishman et al., 2012; Lange et al., 2009) to enable the use of these vehicles in GPS-denied environments, such as close to buildings or indoors.

An alternative way for positioning without GPS is the use of computer vision where image analysis algorithms estimate the UAV position using on-board cameras. Velocity can be estimated using optical flow algorithms and specialized sensors (Horn and Schunck, 1981) and this is attempted with success by Kim and Brambley (2007). Relative position can be estimated using standard camera equipment and object recognition, or using image feature point tracking algorithms as examined by Lange et al. (2009).

1.2 Scope

This thesis employs model-based design using MATLAB Simulink in the development of an indoor autonomous multi-rotor prototype UAV, featuring computer vision based navigation and object recognition to be run on a PandaBoard ES single-board computer. The tasks include development of computer vision based position estimation and position control algorithms, implemented using automatic code generation. Autonomous flight behaviors include takeoff, hover and land.

1.3 Purpose

The UAV prototype is to be used for technology demonstration as well as to show the potential of model-based design methods. The purpose also includes exploration of position estimation techniques using computer vision and the study of algorithms for object recognition.

2 Model-Based Design Method

In a competitive market, actors in technology and engineering industries seek to reduce cost and time consumption for their development processes (Ahmed, 2010). Engineers and managers are thus seeking new methods to improve in these respects. Model-based design strategies have been used through the design phase of development projects for some time. However, the method is now becoming used even beyond the design phase, all the way to production.

2.1 Basic Concept

The idea of model-based design is to reduce development time and cost by eliminating the need for early product prototypes and by narrowing down possible design alternatives early in the process (West, 2009; Fleischer et al., 2009). For instance, consider a mechatronic embedded control system: A model of the plant is developed where physical parameters, such as electric motor windings or gear ratios, can be evaluated and optimized. An existing system can be modelled using system identification tools. Controllers can then be designed and tested in the simulated environment and the complete system can thus be verified and validated before a physical prototype is built.

Modern model-based design methods often incorporate automatic code generation, reducing the gap between controller design and actual software implementation. Also, human coding errors are said to be reduced and MathWorks recently claimed that the expected development time for a hybrid powertrain was reduced by more than 70 % using model-based design methods at General Motors (GreenCarCongress, 2009).

2.2 Simulink and Code Generation

The concept of model-based design in MATLAB Simulink was used throughout this project. The multirotor platform was modelled using physical modeling and system identification methods. Computer vision, autonomy and control algorithms were designed in the Simulink environment and were tested using a plant model. A camera simulator, emulating the image captured by the actual camera equipment, was also developed and used.

The algorithms mentioned above were implemented with Simulink blocksets. The code was automatically generated and downloaded to a PandaBoard ES single-board computer where it was compiled and run. A Simulink-native debugging mode called External Mode was used to monitor signals, variables and camera video feed in real-time using a wireless network connection.

3 System Description

A hexacopter platform from 3DR Robotics was selected because of its hardware availability, ready-to-fly delivery and open source software. Included were motors, speed controllers and the ArduPilot Mega 2.5+ (APM) control unit running the ArduCopter software. An RC-radio was used for manual flight control and a radio telemetry kit was used for data logging as well as ArduCopter configuration using the Windows software APM Mission Planner.

The hexacopter can be flown in different modes using manual control. In the standard mode, hexacopter attitude (roll and pitch), yaw rate and thrust is manipulated with a standard flight RC-radio unit. The APM estimates attitude, altitude and yaw using on-board sensors.

To enable autonomous flight, the hexacopter platform was enhanced with a second on-board computer, the PandaBoard ES. In order to send and receive measurements and control setpoints between this computer and the APM, Ethernet communications hardware was added. Also, a downwards facing Logitech C310 USB camera was mounted on the hexacopter and connected to the PandaBoard. Computer vision and position control algorithms are executed on the PandaBoard using measurements from the APM and attitude setpoints are sent back to the APM. A complete system overview can be seen in Figure 1.

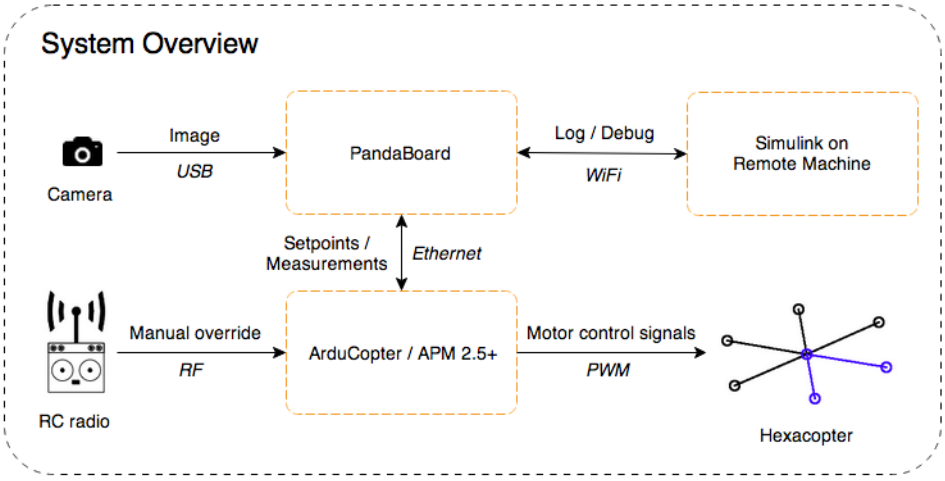
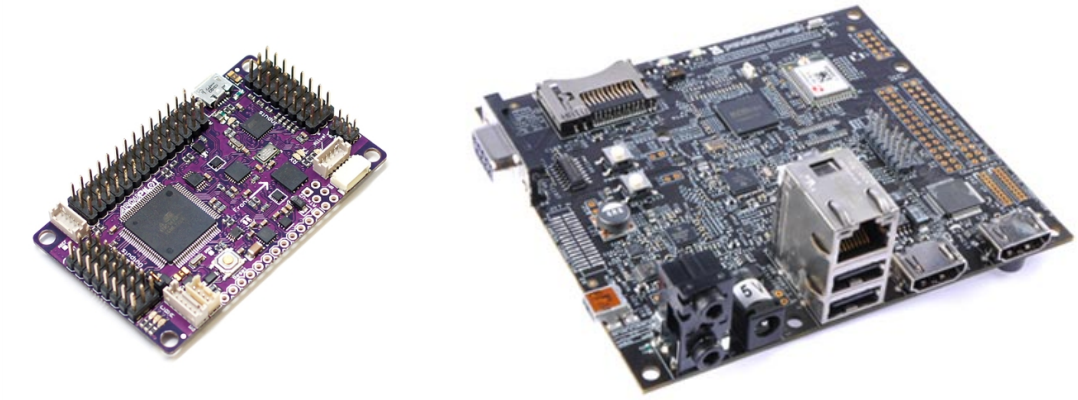


Figure 1: Flowchart of the complete system.

3.1 APM Autopilot

The ArduPilot Mega 2.5+ autopilot features an ATmega2560 microprocessor along with power electronics, flash data storage and programming logics. The onboard inertial measurement unit (IMU) features a three-axis accelerometer, a gyroscope and a magnetometer. These sensors are used for attitude and yaw estimation. A barometer and an external sonar sensor enables altitude measurements and an external GPS receiver is used for outdoor position measurements. The APM hardware is seen in Figure 2(a).



(a) APM (b) PandaBoard

Figure 2: APM 2.5+[1] and PandaBoard ES[2].

3.2 ArduCopter Software

The ArduCopter is an autopilot open source software project for helicopter and multi-rotor platforms (DIY Drones, 2013). It shares parts of its code base with the parent project ArduPilot, made for motorized airplanes and gliders. Both the ArduCopter and ArduPilot softwares are meant to run on the ArduPilotMega hardware series.

The basic functionality of the ArduCopter software includes reading sensors, receiving RC radio input and stabilizing the hexacopter during flight using PID control (Figure 3). Attitude angles are estimated using complementary filtering.

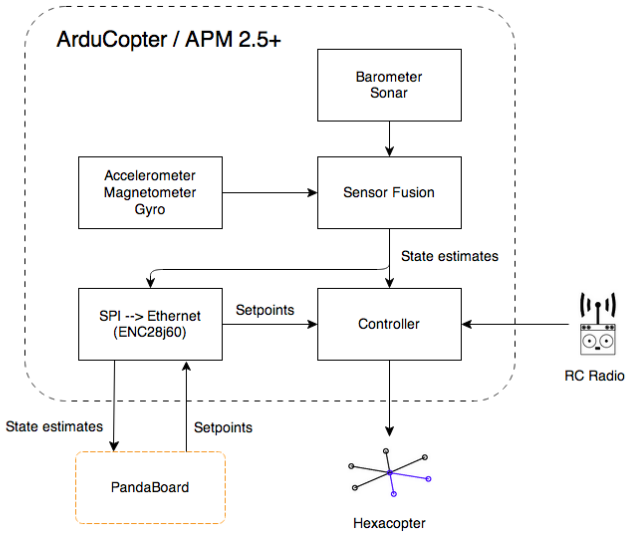


Figure 3: ArduCopter software layout.

[1]http://www.sgdrone.com/
[2]http://www.ixbt.com/

Several control loops are used to stabilize the hexacopter during flight. For basic flight stabilization, two cascaded PID controllers per axis are used to control attitude (seen in Figure 4). A similar approach is used for altitude and yaw control.

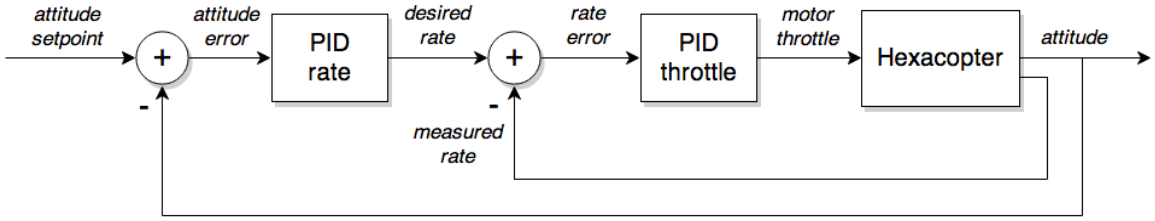


Figure 4: ArduCopter attitude controller layout.

3.3 PandaBoard Hardware

The computer platform used to execute image analysis and position control algorithms is the PandaBoard ES, seen in Figure 2(b). It was selected because of native Simulink code generation support (MathWorks, 2013), its small form factor supporting hexacopter mounting and its USB camera connectivity capabilities. It features a Dual-core ARM Cortex-A9 processor (Pandaboard.org, 2013) and a Linux distribution supplied by MathWorks is run from an SD memory card. A wireless network connection is used for programming and data logging using Simulink. Power is supplied from the hexacopter 12.6V LiPo battery and an external 5V switching regulator.

3.4 PandaBoard Simulink Model

The software run on the PandaBoard was designed in a Simulink model and an overview can be seen in Figure 5. The Vision block performs image analysis algorithms, measuring hexacopter movement over ground and recognizing pre-defined objects. This data is fed to the Autonomy block where a state machine is used to decide what flight behavior to activate. Behavior and Vision measurements are routed to the GNC block (Guidance, Navigation and Control) where the current hexacopter position is estimated, position error is calculated and attitude control setpoints are generated using PD controllers. These are sent to the APM autopilot through the Communication block.

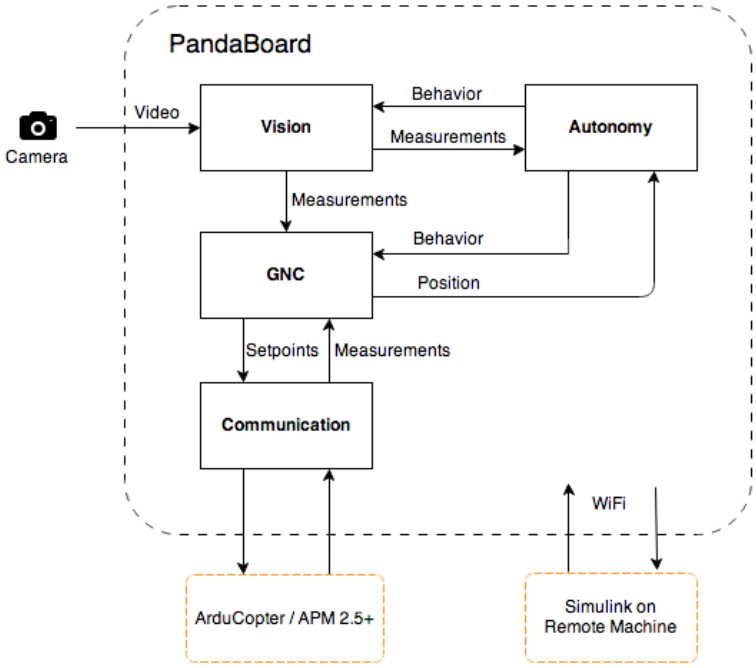


Figure 5: PandaBoard software layout.

3.5 Andon Light

Andon is Japanese for paper lantern and in lean production theory, an Andon signal is a light indicating the state of a workstation, for example working, waiting for parts or broken down (Slack et al., 2010).

An Andon light was designed, built and mounted on the hexacopter in order to indicate the state of the software and communication. Three colored LED lights indicate the current level of autonomy and status of the Ethernet communication. A bleeper is used to warn the user of critical errors in software or communication, requiring user intervention by manual RC control.

4 Modeling and Simulation

For image analysis development and position controller design, a model of the complete UAV system was developed in Simulink. This included an attitude model of the hexacopter platform, obtained using system identification. This was extended with yaw and position models and a virtual camera, used to emulate the image sent from the camera. The models put together, referred to as the hexacopter model, enabled system verification and validation prior to actual flight testing.

4.1 Multirotor Position Model

Work has previously been done on developing mathematical models of multirotor platforms (Andersson et al., 2010; Luukkonen, 2011). Since the ArduCopter software already implements attitude control for flight stabilisation, a dynamic model featuring motor dynamics, body inertias and propeller aerodynamics was omitted here. Instead, a position model was derived using physical modeling and an attitude model was obtained using system identification. For illustration and simplification purposes, a quadrotor platform is shown here. This model, however, is compatible with a hexacopter configuration.

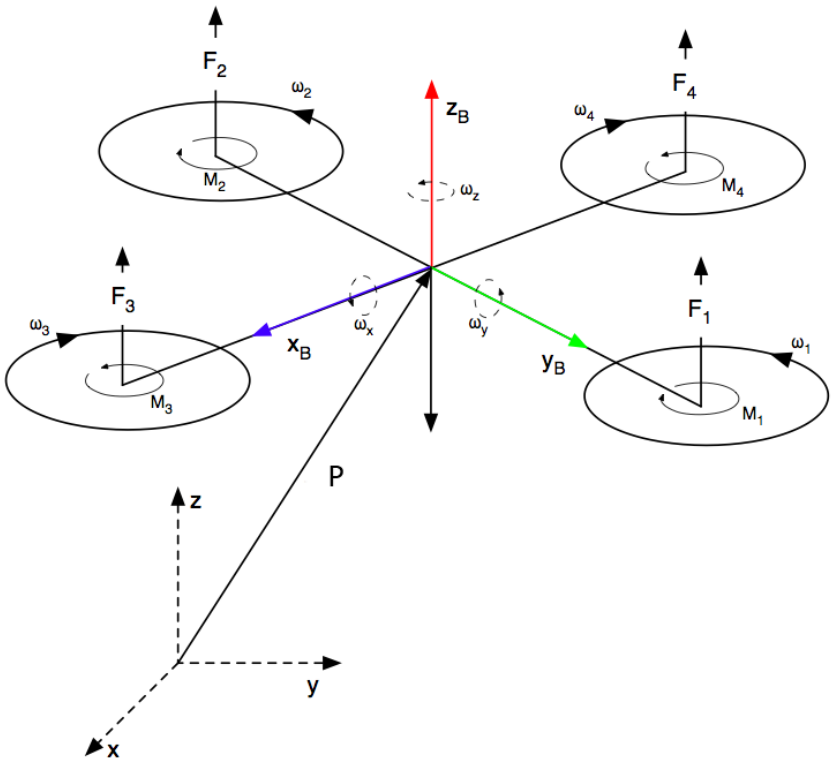


Figure 6: Multirotor position and angles.[3]

[3]Andersson et al. (2010)

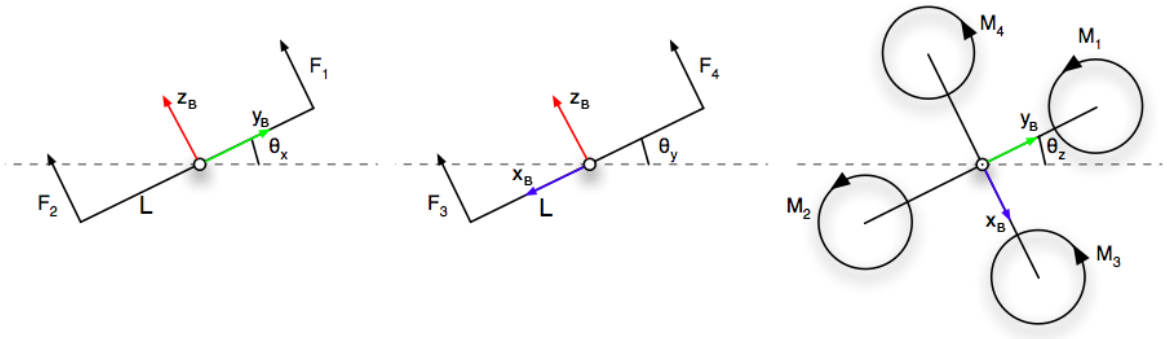


Figure 7: Multirotor angles $\boldsymbol{\theta}$.[4]

The multirotor position $\boldsymbol{P}$ (Figure 6, Equation (1)) is defined in the world inertial frame (x, y, z). The multirotor orientation can be described in this frame using angles $\boldsymbol{\theta}$ (Figure 7) or in the body-fixed frame (x_B, y_B, z_B) as $\boldsymbol{\theta}_B$ (Equation (2)). The body fixed angles θ_B are referred to as pitch, roll and yaw. Also, the pitch and roll angles are together named attitude.

$$\boldsymbol{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \text{latitude} \\ \text{longitude} \\ \text{altitude} \end{bmatrix} \tag{1}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \qquad \boldsymbol{\theta}_B = \begin{bmatrix} \text{roll} \\ \text{pitch} \\ \text{yaw} \end{bmatrix} = \begin{bmatrix} \theta_{Bx} \\ \theta_{By} \\ \theta_{Bz} \end{bmatrix} \tag{2}$$

An xy position model was derived by double integration of Newton's second law of motion, using multirotor thrust vector $\boldsymbol{F}(t)$, mass m and the gravitational acceleration vector $\boldsymbol{g}$.

$$\ddot{\boldsymbol{P}}(t) = \frac{\boldsymbol{F}(t)}{m} + \boldsymbol{g} \tag{3}$$

$$\boldsymbol{P}(t) = \iint \left(\frac{\boldsymbol{F}(t)}{m} + \boldsymbol{g} \right) \mathrm{d}t^2 \tag{4}$$

Figure 8 illustrates how different values of θ_x affect multirotor velocity v_y. Assuming constant motor thrust $F_1 = F_2$ and $F_3 = F_4$, a resulting total thrust $F_{hover} = gm$ can be used. This assumes that a constant total thrust is required to keep the multirotor hovering at a constant altitude. Using this simplification and the assumption that the x and y axis models are independent of one another, a simplified position model is expressed as

$$p_{x,y}(t) = \iint \frac{F_{hover}}{m} \sin\left(\theta_{y,x}(t)\right) \mathrm{d}t^2. \tag{5}$$

[4] Andersson et al. (2010)

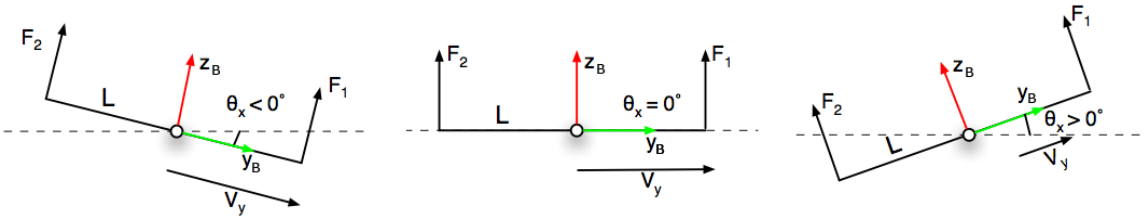


Figure 8: Multirotor angle θ_x affect acceleration and thus velocity in the y direction.[5]

Equation (5) is now expressed in the Laplace domain by first linearising using $\sin x \approx x$ for small x and a final position model $G_p(s)$ is thus achieved.

$$p_{x,y}(s) = \frac{F_{hover}}{s^2 m}\theta_{y,x}(s) \tag{6}$$

$$G_p(s) = \frac{p_{x,y}(s)}{\theta_{y,x}(s)} = \frac{F_{hover}}{s^2 m} \tag{7}$$

In order to set pitch and roll angles according to world frame angles θ_y, θ_x, a rotation of these angles by θ_z is required (see Figure 7). This was done using rotational matrix R_z as

$$\begin{bmatrix}\text{pitch}\\ \text{roll}\end{bmatrix} = \begin{bmatrix}\theta_x\\ \theta_y\end{bmatrix} R_z \tag{8}$$

where

$$R_z = \begin{bmatrix}\cos\theta_z & -\sin\theta_z\\ \sin\theta_z & \cos\theta_z\end{bmatrix}. \tag{9}$$

4.2 Multirotor Attitude Model

The model describing the roll and pitch angles was derived using system identification. This method is a process of identifying mathematical models of physical systems (Ljung, 1988). Non-parametric or parametric identification methods can be used. The former method includes transient and frequency analysis, while parametric identification involves testing defined model structures and finding their parameters.

Black box modelling refers to the case where little or no information of the system is available (Falcone, 2010). Grey box modelling, on the other hand, refers to the case where a system model structure is known, but some parameters are unknown or uncertain.

As the multirotor is controlled using angular setpoints sent to the ArduCopter software, a model describing the dynamics of both the physical system and the PID attitude controllers was used for simulation and position controller design purposes.

In the case of modeling the hexacopter attitude from setpoint to actual angle, it was known that two cascaded PID controllers were embedded (seen in Figure 4) (DIY Drones,

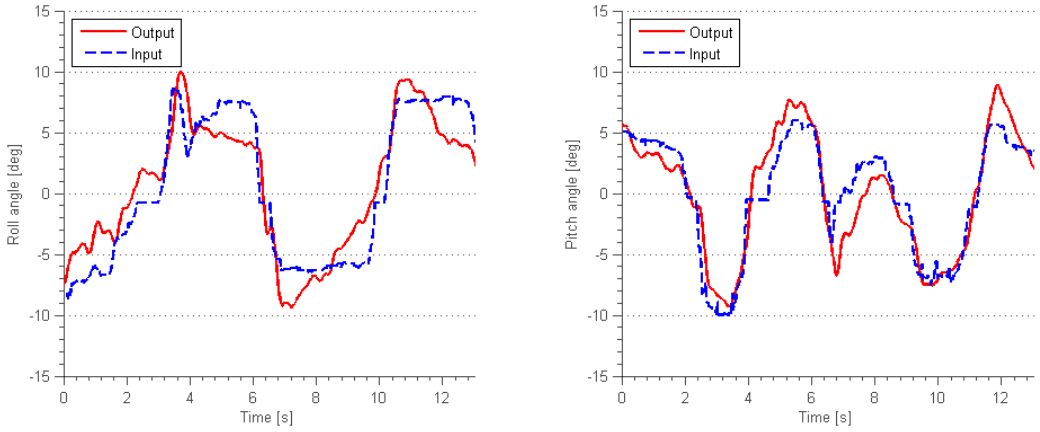
[5] Andersson et al. (2010)

2013). The model achieved through system identification is thus a representation of this closed loop system including electric, mechanical and aerodynamic properties.

The general process model

$$G_a(s) = K_p \frac{1 + sT_z}{(1 + sT_{p1})(1 + sT_{p2})} e^{-sT_d} \tag{10}$$

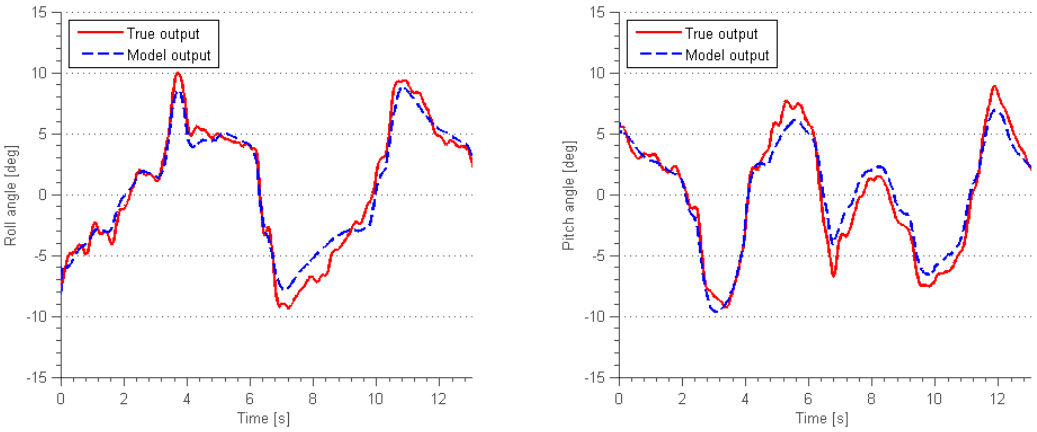
was used for the identification with a static gain K_p, a zero at $-1/T_z$, two poles at $-1/T_p$ and a time delay of T_d. This second order system was assumed to be sufficient to describe the process dynamics. The excitation signal, sent as input to the system using RC-radio, was manually created to mimic a piecewise constant telegraph signal. Input and output data logged from test flights (see Figure 9) show a nearly static relationship between input and output. This is expected since the system has feedback control. However, it can be observed that the step response is decreasing for constant nonzero input, motivating the identification of a dynamic system model since a static gain model could have problems representing the actual plant behaviour with accuracy sufficient for simulation and position control design.



(a) Roll (b) Pitch

Figure 9: Roll/Pitch input and output.

Using MATLAB System Identification Toolbox, a process model was obtained. Model output fit percentage for different number of poles, zeros and delay is presented in Table 2 and 3 in Appendix A. The best fit was obtained using two poles, one zero and a delay, as presented in Equation (10). The identified roll and pitch models were tested against validation input and output data (Figure 10). The model step responses are seen in Figure 11. The model parameters can be seen in Table 4 and 5 in Appendix A.



(a) Roll (b) Pitch

Figure 10: Roll/Pitch model output with verification data.

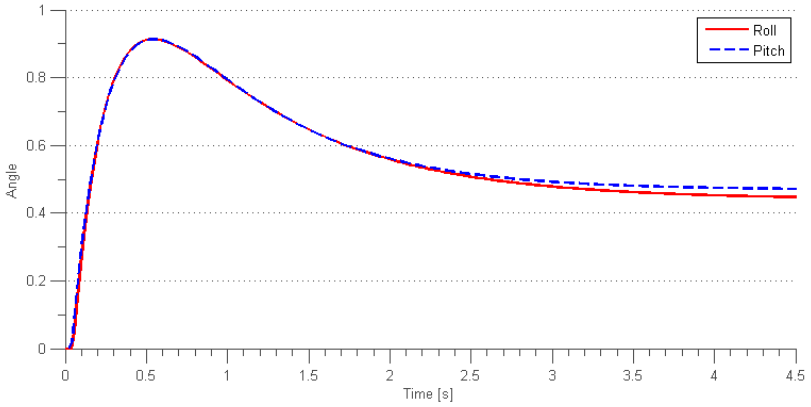


Figure 11: Identified roll and pitch model step responses.

4.3 Camera Model

A model of the Logitech camera was developed in order to emulate its image capturing behaviour and to test image analysis algorithms. The model describes the relationship between pixel values and world coordinates. Figure 12 illustrates a camera positioned above ground, pointing downwards.

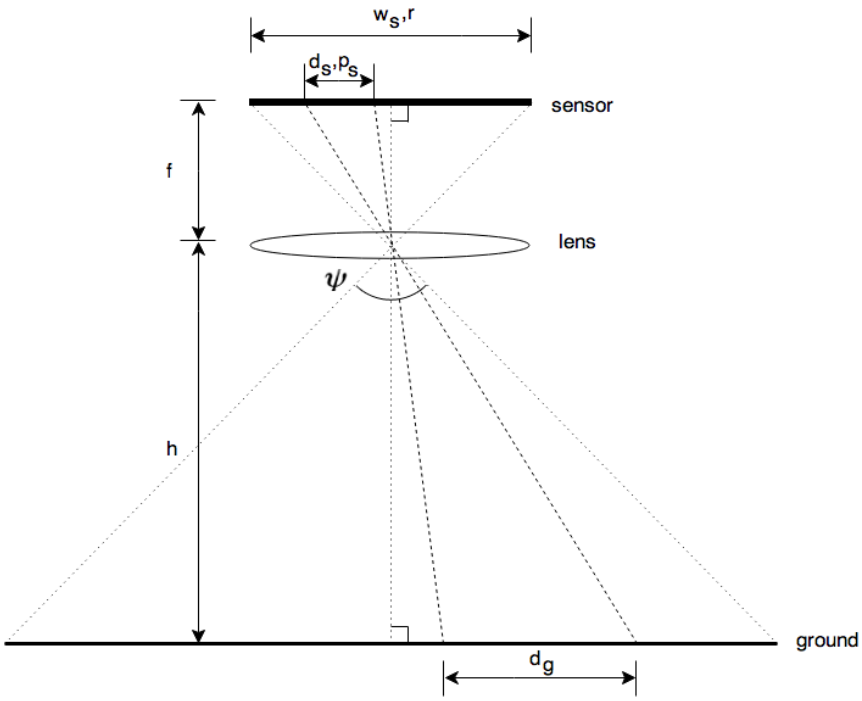


Figure 12: Camera model.

The relationship between the ground and sensor distances d_g and d_s can be found using camera height h, focal length f and similarity of triangles.

$$\frac{d_s}{d_g} = \frac{f}{h} \tag{11}$$

Then, a relationship between d_s and the corresponding number of pixels p_s is found using sensor width w_s and corresponding pixel resolution r.

$$d_s = \frac{p_s w_s}{r} \tag{12}$$

Sensor width w_s can be found using known camera parameters angle of view ψ and the focal length f.

$$\frac{w_s/2}{f} = \tan\left(\frac{\psi}{2}\right) \tag{13}$$

Combining Equation (12) and (13) with (11), a final expression for ground distance is obtained as

$$d_g = \frac{2hp_s}{r} \tan\left(\frac{\psi}{2}\right). \tag{14}$$

The camera mounted on the hexacopter will always point downwards along the z_B axis (Figure 6). Position measurements from image analysis algorithms would thus have to be compensated for the current hexacopter attitude. As illustrated in Figure 13, an object positioned directly underneath the camera will not always appear in the center of the acquired image.

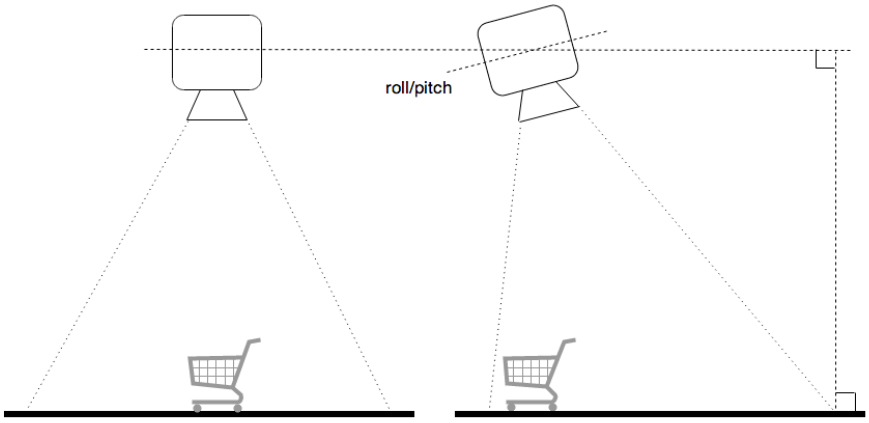


Figure 13: Camera attitude affects position measurements.

Assuming that a measurement compensation can be done with respect to the roll θ_{Bx} and pitch θ_{By} angles independently, an approximation of the relationship between attitude and camera pixel x and y coordinate p_s was formulated using camera angle of view ψ (Equation (15) and (16)). This approximation is valid for small attitude angles and is based on the measure of pixels per angle of view in the center of the image.

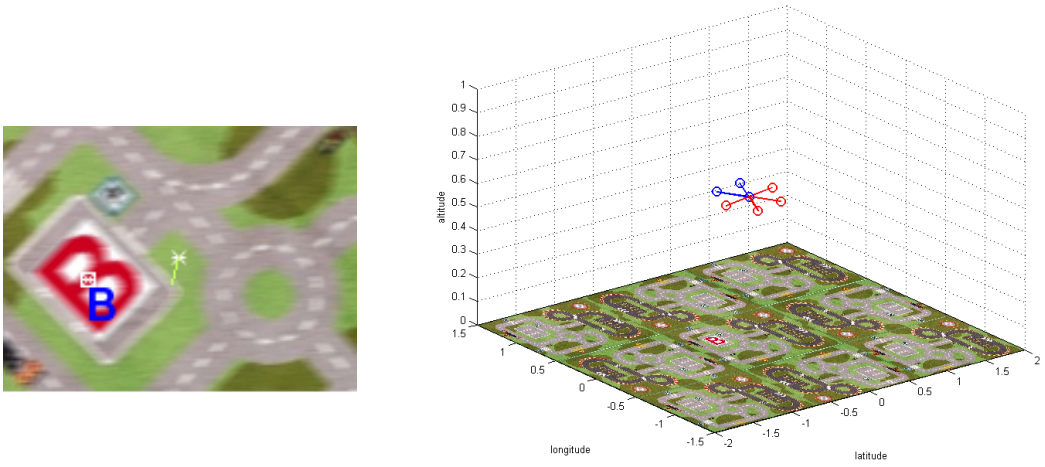
$$\Delta p_{s,x} = \frac{r\,\theta_{By}}{\psi} \tag{15}$$

$$\Delta p_{s,y} = \frac{r\,\theta_{Bx}}{\psi} \tag{16}$$

4.4 Simulation Environment

For simulation purposes, an algorithm emulating the camera view was developed. It produces a cutout of a floor image which is translated, scaled and rotated according to the hexacopter model output. It also features motion blur derived from current velocity and shutter speed. The emulated image was used for simulation of the PandaBoard software and the image was presented using a video window, seen in Figure 14(a). Here, information such as position and attitude setpoints was overlayed. The floor was assumed to be a play mat.

In order to visualize the simulated hexacopter movement and behavior, a 3D visualization was developed and this can be seen in Figure 14(b). A floor image is displayed together with a hexacopter representation, animated to display position, altitude, attitude and yaw calculated by the hexacopter model.



(a) Emulated camera view

(b) 3D visualization of simulation

Figure 14: Emulated camera view and 3D visualization of simulation.

5 Computer Vision

The Vision block (Figure 15) was designed to measure hexacopter movement over ground and to recognize objects. The active algorithm is selected using current behavior (Section 7) and the output is a pixel offset relative to the center of the image.

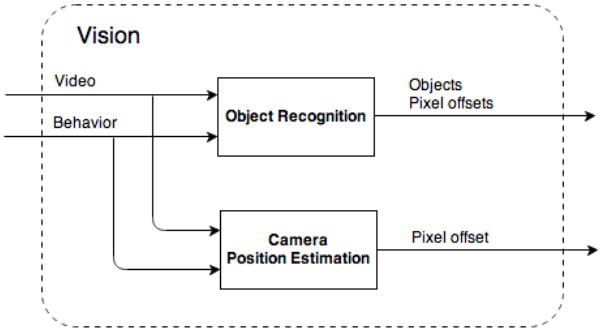


Figure 15: Vision block layout.

5.1 Object Recognition

The hexacopter was equipped with the capability to recognize letters. This enables it to trigger certain predefined behaviors, such as landing and following of objects, when different letters are recognized. To limit the scope of this task, only red capital letters A and B are detected using a classifier with two features. However, most of the theory presented here applies to an infinite amount of objects. The object recognition algorithms were implemented without using MATLAB Computer Vision Toolbox.

A general layout for a computer vision system aimed at object recognition can be seen Figure 16. The process can be divided into three stages: low-level, intermediate and high-level processing (Mehnert, 2012). Preprocessing can be image sharpening, color space conversion and scaling while segmentation is the process of extracting areas or objects of interest in the image. Representation and description extract metrics from the segmented objects, no longer describing them as images. Recognition and interpretation match the detected object metrics to those of known objects. All processing stages use information of the object to be recognized, here denoted knowledge base.

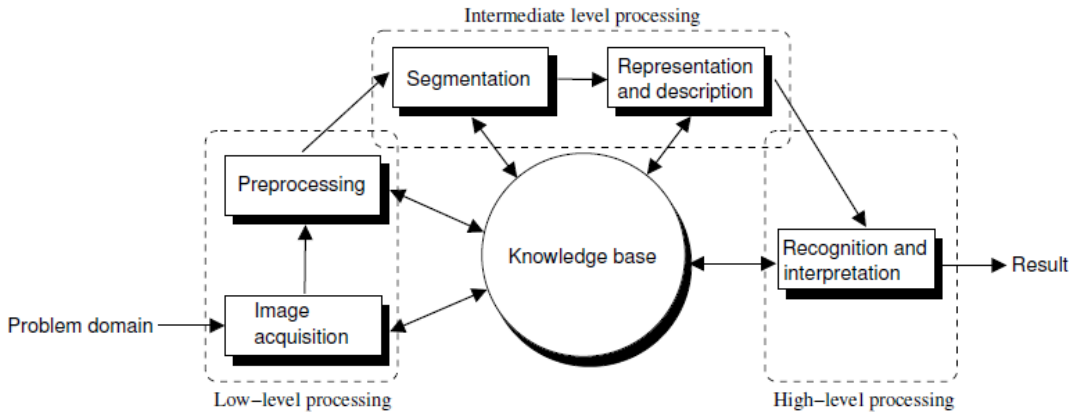
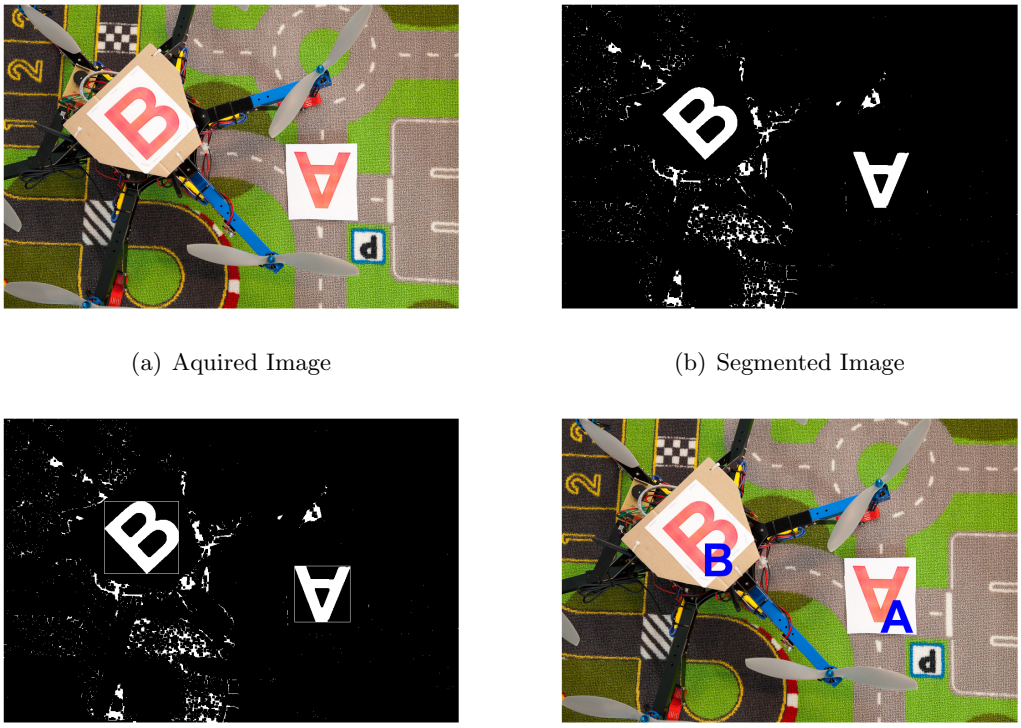


Figure 16: Typical stages of object recognition.[6]

The acquired image in Figure 17(a) is converted to HSV color space and is segmented by the color red to yield the binary image shown in Figure 17(b). It was noted that an inaccurate object segmentation due to e.g. illumination affects the recognition process negatively. Assuming the texture of the floor does not have large areas of red, object detection is done by selecting the largest connected red regions. These are then passed on to feature extraction and classification. In Figure 17(c), the two largest regions are highlighted for display purposes and their bounding boxes are drawn. The regions are classified and the results are displayed in Figure 17(d).

[6]Mehnert (2012)



(a) Aquired Image

(b) Segmented Image

(c) Objects Detected with Bounding Boxes

(d) Objects Recognized

Figure 17: Different stages and results of object recognition.

5.1.1 Segmentation

The image acquired from the camera is obtained in the RGB color space where a color is defined in three chromaticities (channels). Red, green and blue additive primaries are combined to form different colors (Gonzalez and Woods, 2006). Even though it is possible to define the range of RGB primaries making up a specific color, this is somewhat non-intuitive (Chen et al., 2008).

The HSV color space is an alternative to RGB. Instead of red, green and blue channels, color in HSV is represented with hue, saturation and value. Here, definitions of specific colors are closer to human perception since the intensity information is decoupled from color (Sonka et al., 2008), see Figure 18. Red can, for example, be roughly defined as {hue; saturation; value} = {[330°,30°]; [0.4,1]; [0.3, 1]}. For segmentation, HSV has been shown to give better results and it has an easily invertible transform to RGB (Chen et al., 2008).

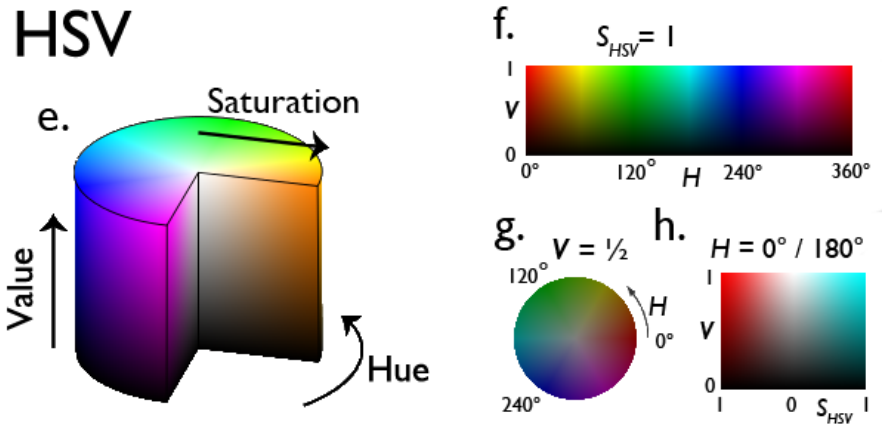


Figure 18: HSV Color Space [7]

5.1.2 Description

Moments are a type of regional descriptors and it is a statistical property of a normalized grey-level image seen as a probability density function of a 2D random variable. The binary image from segmentation is used where the non-zero values are assumed to be regions (objects). A raw moment in its general form of order (p + q) is not invariant to translation, scaling or rotation (Sonka et al., 2008). For an image, it is given by

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y) \tag{17}$$

where x and y are the regional pixel coordinates and $I(x, y)$ is the intensity value of the image at position (x, y). In binary images, $I(x, y)$ is either zero or one.

Moments can be written as central moments

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \tag{18}$$

taken about the mean of the distributions, instead of about zero, in order to achieve translation invariance (Weisstein, 2013). Here, $(\bar{x}, \bar{y})$ is the center of gravity coordinate for the region and can be obtained using

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \text{and} \qquad \bar{y} = \frac{M_{01}}{M_{00}} \tag{19}$$

where M_{00} corresponds to the area of the object, as seen in Equation (17).

[7] $http://en.wikipedia.org/wiki/HSL\_and\_HSV$

To reduce calculation time and complexity, the central moments are expressed with the raw moments using the binomial transform, as shown in one dimension by Weisstein (2013). Equation (20) is expanded up to the third order.

$$\mu_{pq} = \sum_{m}^{p} \sum_{n}^{q} \begin{pmatrix} p \\ m \end{pmatrix} \begin{pmatrix} q \\ n \end{pmatrix} (-\bar{x})^{(p-m)} (-\bar{y})^{(q-n)} M_{mn} \tag{20}$$

$$\begin{aligned}
\mu_{00} &= M_{00} \\
\mu_{10} &= 0 \\
\mu_{01} &= 0 \\
\mu_{11} &= M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10} \\
\mu_{20} &= M_{20} - \bar{x}M_{10} \\
\mu_{02} &= M_{02} - \bar{x}M_{01} \\
\mu_{21} &= M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2 M_{01} \\
\mu_{12} &= M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2 M_{10} \\
\mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2 M_{10} \\
\mu_{30} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2 M_{01}
\end{aligned} \tag{21}$$

Scale invariance can be achieved using the normalized unscaled central moments (Sonka et al., 2008) as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}}. \tag{22}$$

A set of seven moments that are invariant to rotation mirroring (except for a minus sign), scaling and translation is derived from the second and third order normalized unscaled central moments (Gonzalez and Woods, 2006), originally shown by Hu (1962).

$$\phi_1 = \eta_{20} + \eta_{02} \tag{23}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{24}$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{25}$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{26}$$
$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \tag{27}$$
$$(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{28}$$
$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{29}$$
$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2)] - \tag{30}$$
$$(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{31}$$

In order to reduce the dynamic range and simplify interpretation (Gonzalez and Woods, 2006), the Hu set of moments (Equation (23) through (31)) is transformed using

$$\phi_n = \text{sgn}(\phi_n) \log_{10}(|\phi_n|) \tag{32}$$

where $\text{sgn}(\phi_n)$ is used to retain the sign of the moment in order to determine whether the region is mirrored or not. For recognition purposes, this is not relevant and instead,

$$\phi_n = |\log_{10}(|\phi_n|)| \tag{33}$$

is used to further reduce the dynamic range and make classification more straightforward.

5.1.3 Feature Selection

Features ϕ_1 to ϕ_7 from Equation (23) through (31) were extracted for every letter in class A and B in the full training set shown in Figure 19 to form the feature space $\mathbf{\Phi} = [\phi_1 \ \phi_2 \ \dots \ \phi_7]^{\text{T}}$. To make every feature equally important, the mean and variance were removed, neutralizing dominating features that would otherwise give a non-optimal classifier (Mehnert, 2012).

(a) Class A (b) Class B

Figure 19: Training set for letters A and B.

Given the scope of a two dimensional classifier, two features from the feature space needed to be selected to form a space $\boldsymbol{x}$ in Euclidean $\mathbf{R}^2$. These features should preferably separate class A from B with the largest possible distance between them. Visually inspecting every combination of features in the feature space would not only be inaccurate but it would also require looking at a number of graphs, equal to

$$\binom{\dim\mathbf{\Phi}}{\dim\boldsymbol{x}} = \frac{7!}{2!(7-2)!} = 21. \tag{34}$$

The Bhattacharyya distance can be used as a measure of class separability (Choi and Lee, 2000) and it is defined as

$$D_B = \frac{1}{8}(\mu_1 - \mu_2)^{\mathrm{T}}\Sigma^{-1}(\mu_1 - \mu_2) + \frac{1}{2}\ln\left(\frac{\det\Sigma}{\sqrt{\det\Sigma_1 \det\Sigma_2}}\right) \tag{35}$$

where μ_i and Σ_i are the mean vector and covariance matrix, respectively, for class i and

$$\Sigma = \frac{\Sigma_1 + \Sigma_2}{2} \tag{36}$$

The Bhattacharyya distance was measured for every combination of features in $\mathbf{\Phi}$ and the greatest class separation of $D_B = 0.3$ (seen in Figure 20(a)) was achieved using feature space $\boldsymbol{x} = [\phi_1 \ \phi_3]^{\mathrm{T}}$. Choi and Lee (2000) states that knowing the Bhattacharyya distance gives an estimate classification error of

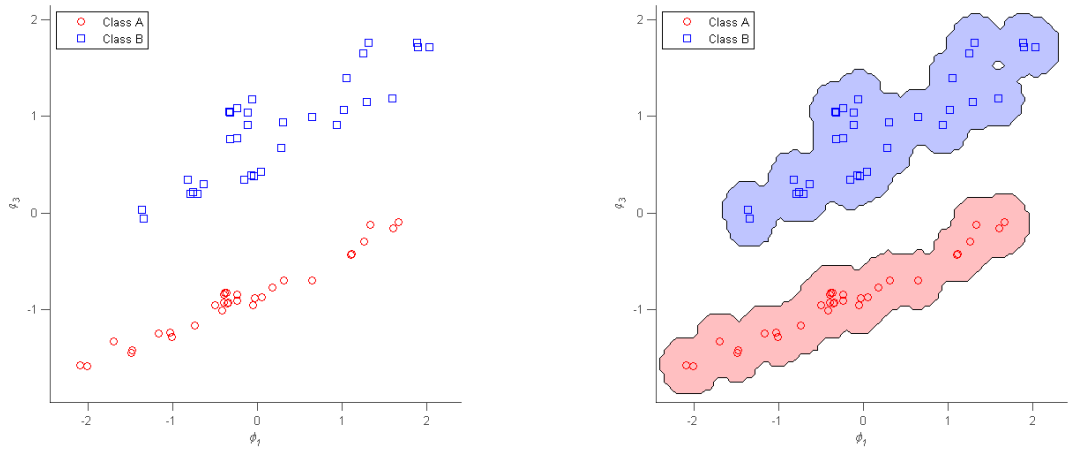
$$\hat{\varepsilon} = 40.219 - 70.019D_B + 63.578D_B^2 - 32.766D_B^3 + 8.7172D_B^4 - 0.91875D_B^5, \tag{37}$$

here equal to 24%.

5.1.4 Recognition

Recognition based on matching uses a trained prototype feature space for each class. An unknown object to be classified should be matched to the class to which it has the closest metric. The minimum distance classifier is the most basic approach to this problem (Gonzalez and Woods, 2006). It works by temporarily placing the unknown sample in the feature space and classifying it to the class of which has the smallest combined (Euclidean) distance to the sample. This approach works well when the distance between means is large compared to the spread of each class. In practice however, this is uncommon except if the system designer has control over the nature of input to the system.

Here, as can be seen in Figure 20(a), the distance between means was in the same order as the spread of the classes. Regarding input control, the scope did allow for choosing a suitable font for the selected feature space.



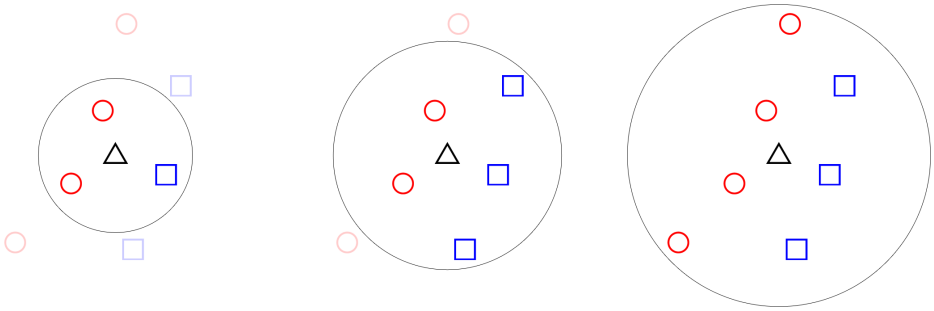
(a) Selected feature space is $\boldsymbol{x} = [\phi_1\ \phi_3]^{\mathrm{T}}$. The squares are the letter B and the circles the letter A.

(b) Precalculated classification map. The top region is letter B and the bottom region is letter A.

Figure 20: Features selected and classification map.

Like the minimum distance classifier, the k-Nearest Neighbors algorithm (kNN) places the unknown sample in the prototype feature space and then computes the distance to every sample in that space. Unlike the minimum distance classifier, kNN makes its decision based solely on the k-nearest samples and not the entire set. It can be argued that this reduces the effect of the spread of the classes compared to the difference in mean.

The kNN algorithm is explained using Figure 21. The unknown sample, here in the form of a triangle, is temporarily placed in the feature space. Figure 21(a) shows the classification when looking at k = 3 nearest neighbors, two circles and one square, resulting in the sample being classified as a member of the class represented by a circular markers. When k = 5, the same unknown sample is classified as the square class. This changes once again when k = 7 when the sample is deemed to be a member of the circles.



(a) k = 3 (b) k = 5 (c) k = 7

Figure 21: kNN algorithm visualization.

Here, the impact of k on the final classification map (Figure 20(b)) was reduced by discarding one neighbor and picking another if the distance between the unknown sample and the neighbor was greater than d. This distance was selected small so as not to have overlap

between classes, causing a potential misclassification and thus making the hexacopter behave erratically. Classification robustness was improved by requiring the object to be recognized n times as the same class before being classified or discarded.

The classification map was pre-calculated in order to reduce the algorithm computation time on the PandaBoard. The map can be seen in Figure 20(b). If an unknown sample is within the red region, it is classified as a letter A and if it is within the blue region, it is classified as a B. A sample outside the colored regions is discarded, classifying this object as neither an A nor B.

5.2 Position Estimation

Here, different methods for hexacopter position estimation using image analysis algorithms, implemented with MATLAB Computer Vision Toolbox, are presented. The approach is motivated by the fact that GPS measurements most of the time are unavailable indoors. Also, relaying on inertial measurements (i.e. gyro and accelerometers) alone will accumulate a drift in the position estimate (Huster and Rock, 2001).

5.2.1 Feature Point Matching

One way of estimating relative motion between two images is the use of feature points. These are obtained in the two images and are then matched to each other. A geometric translational and/or rotational transformation describing the relation between the two sets of points is then estimated.

The feature points can be found using corner detection (Rosten and Drummond, 2006) or SURF (Speeded-Up Robust Features) (Bay et al., 2006). The first method involves finding corners in the image, assuming the same set of corners will be found in the next image frame. SURF is designed to find feature points invariant to image scaling and rotation. The two sets of points are matched with each other using correlation and a geometric transform is estimated.

The geometric transformation assumes a non-reflective similarity transform, thus detecting point translation, scaling and rotation. This makes the method somewhat robust to hexacopter yaw and altitude change. The fact that the point matching method implemented here uses correlation to match the points did however limit the rotation and scaling robustness. If these effects are known and their magnitude can be measured, however, they can be compensated for.

5.2.2 Template Matching

One method for finding the relative translational motion between two images is the use of template matching. Here, a small part of the current image is matched with a reference image, representing a known position. The best match location represents the distance that the image has been translated, or in other words, the hexacopter has moved. A region of interest is selected around the previous match location, assuming a maximum hexacopter velocity. This reduces the number of match metric calculations required. The template match metric is calculated using a sum of absolute differences.

Tests showed that the algorithm is sensitive to image rotation and scaling, reducing the likelihood of finding a valid template match. Thus, an algorithm triggering re-acquisition of the template image was developed. Re-acquisition is performed when a change in hexacopter

yaw or altitude larger than a threshold is detected since these changes reduce the probability of finding a good template match. Re-acquisition is also performed when a large position change is measured, indicating a bad template match location, and when the template is moving outside the current image field of view.

5.2.3 Algorithm Evaluation

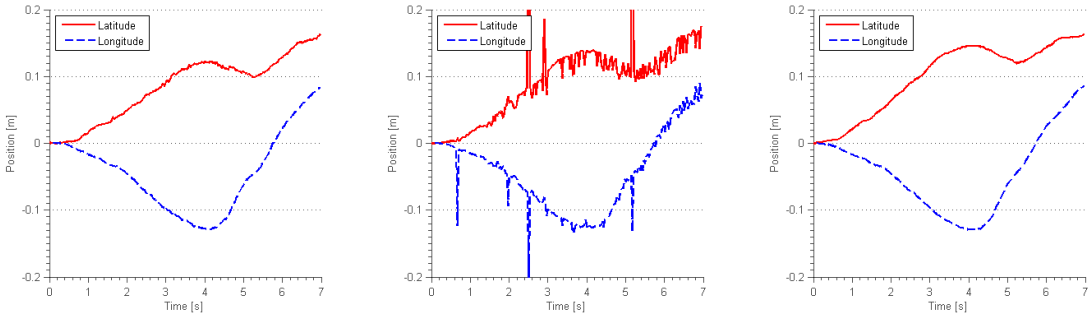
The different approaches to motion estimation mentioned above were evaluated using two different floor textures, marble floor tiles and a play mat from IKEA (Figure 22). A sample video featuring translational movement captured at an altitude of 1 meter was used and the estimated pixel translation obtained from the image analysis algorithms was recalculated to a position in meters using the camera model derived in Section 4.3. The video was captured so that a small center part of the image is always in frame.

(a) Marble floor (b) Play mat

Figure 22: The two floor textures used for algorithm evaluation.

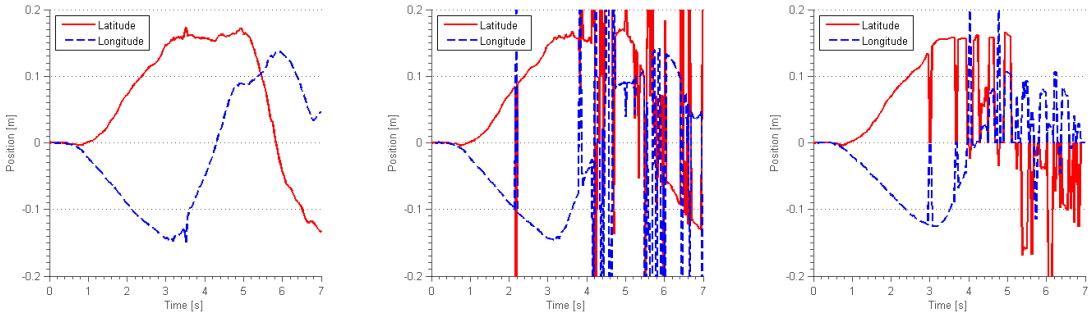
Figures 23 and 24 show the position estimated using the three different motion estimation algorithms. For comparison reasons, none of the algorithms are enhanced with the aforementioned image re-acquisition methods. By visual inspection, one can conclude that the SURF method outperformed both alternative algorithms even though true camera position is not measured. However, the MATLAB implementation of SURF does currently not support automatic code generation and was therefore rejected. One alternative way of implementing SURF is the use of the open source computer vision library OpenCV (Itseez, 2013) in combination with MATLAB S-functions.

The remaining two algorithms both had problems using the play mat. Template matching, however, performed better than Corner detection on the marble floor and was therefore selected for further development. Using template re-acquisition, template matching was made capable of adequate position estimation using the play mat (Figure 25).



(a) SURF (b) Corner detection (c) Template matching

Figure 23: Position estimate 1 meter above marble floor.



(a) SURF (b) Corner detection (c) Template matching

Figure 24: Position estimate 1 meter above play mat.

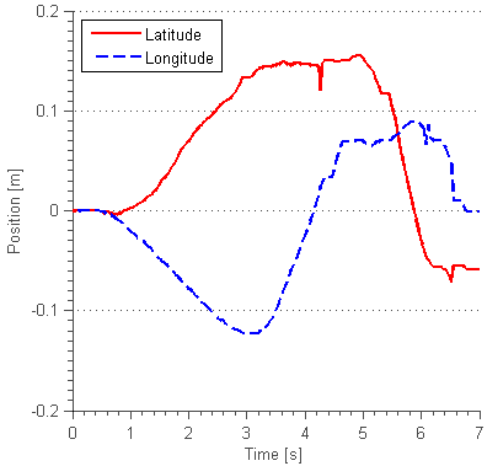


Figure 25: Position estimate 1 meter above play mat using template matching and template re-acquisition algorithms.

6 Guidance, Navigation and Control

It is in aerospace engineering common to group on-board flight control systems into three subsystems: guidance, navigation and control (GNC) (Kim et al., 2006). Each one of these systems can feature more or less complex functionality and can be developed independent of each other as long as their interfaces are well defined. The structure also enables future extension and enhancement of functionality and performance. A GNC subsystem block structure was implemented in Simulink for use on the hexacopter (see Figure 26).

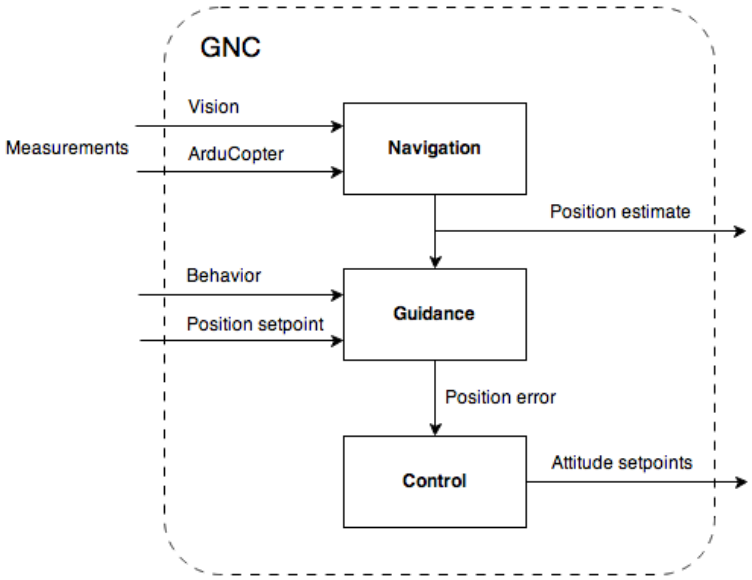


Figure 26: Guidance, Navigation and Control layout

6.1 Navigation

The Navigation block is responsible for the position estimation. To do this, vision measurements are recalculated to world coordinates using camera model equations (Section 4.3) and attitude, heading and altitude measurements sent from the APM. Position offset measurements from the Vision block are stored in order to keep track of the hexacopter's absolute position.

6.2 Guidance

The Guidance block calculates a position error based on coordinates set by the user or by the Autonomy block which is described in Section 7.

6.3 Control

The Control block calculates attitude control signals based on the current position error received from the Guidance block. The control output is sent to the ArduCopter and is calculated using cascaded PD and P controllers, one for velocity and the other for attitude (see Figure 27). It can be argued that this control structure provides more straightforward tuning than a single PD controller.

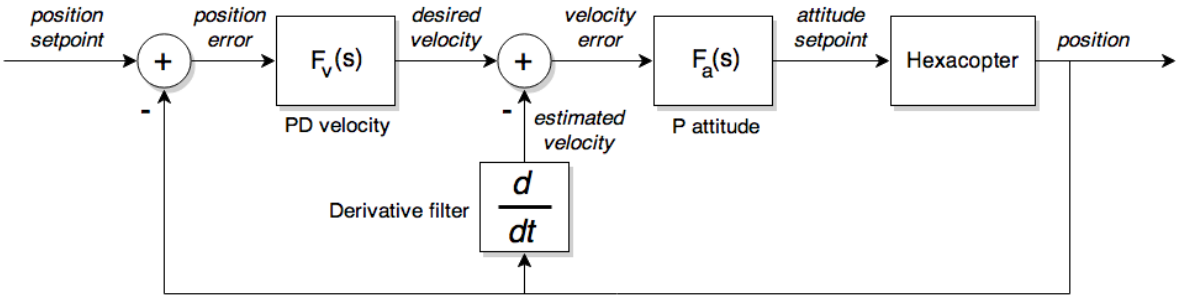


Figure 27: Position control layout. During simulation, the Hexacopter block is represented by attitude and position models $G_a(s)$ and $G_p(s)$

Continuous controllers were designed, tuned and analysed for stability using the hexacopter model derived in Section 4. Using position p and setpoint r_p, controllers $F_v(s)$ and $F_a(s)$ and the hexacopter attitude and position models $G_a(s)$ and $G_p(s)$, a closed-loop transfer function $H_{rp}(s)$ was derived from the control layout in Figure 27 as

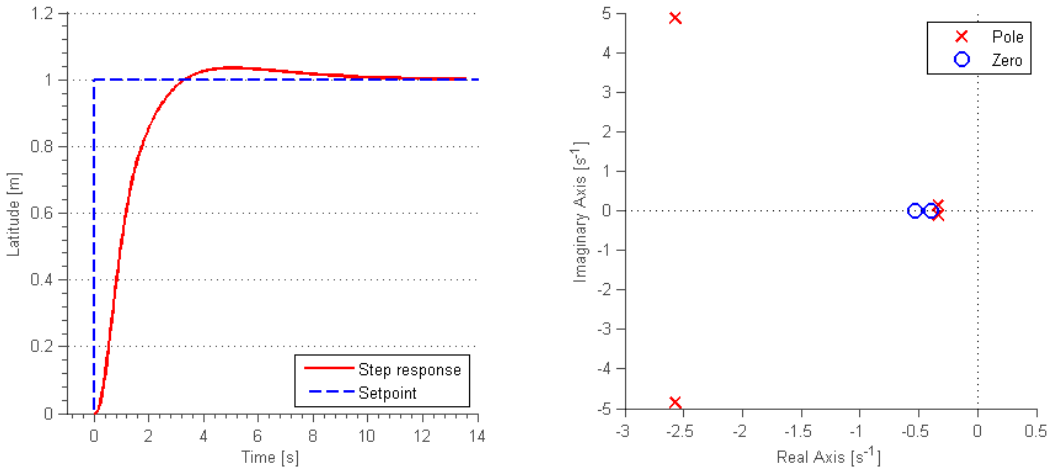
$$p(s) = \left((r_p(s) - p(s)F_v(s) - sp(s)\right) F_a(s)G_a(s)G_p \tag{38}$$

$$H_{rp}(s) = \frac{p(s)}{r_p(s)} = \frac{F_v(s)F_a(s)G_a(s)G_p(s)}{1 + F_v(s)F_a(s)G_a(s)G_p(s) + sF_a(s)G_a(s)G_p(s)} \tag{39}$$

The PD controller transfer function is described as

$$F(s) = K_p + sK_d \tag{40}$$

where the velocity controller static gain K_p was tuned to produce a 1 m/s velocity for every meter of position error. K_d was tuned to reduce position overshoot. The attitude controller was designed as a simple P-controller, tuned to produce an attitude angle of 10 degrees for a velocity setpoint of 1 m/s. The step response of $H_{rp}(s)$ and a plot of the system poles and zeros can be seen in Figure 28. As seen here, all poles are found in the left half plane, indicating a stable system.



(a) Step response

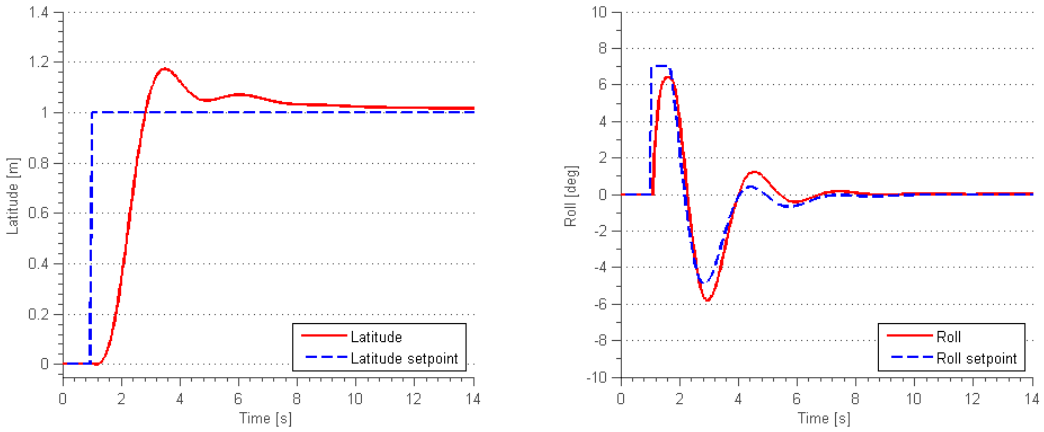
(b) Pole-zero plot

Figure 28: $H_{rp}(s)$ step response and pole-zero plot.

In order to implement the aforementioned continuous PD and P controller structure in the Simulink model running in discrete time on the PandaBoard, a discrete transfer function $F(z)$ was formulated. The derivative is obtained using Backward Euler and a derivative filter with coefficient N.

$$F(z) = K_p + K_d \frac{N}{1 + NT_s \frac{z}{z-1}} \tag{41}$$

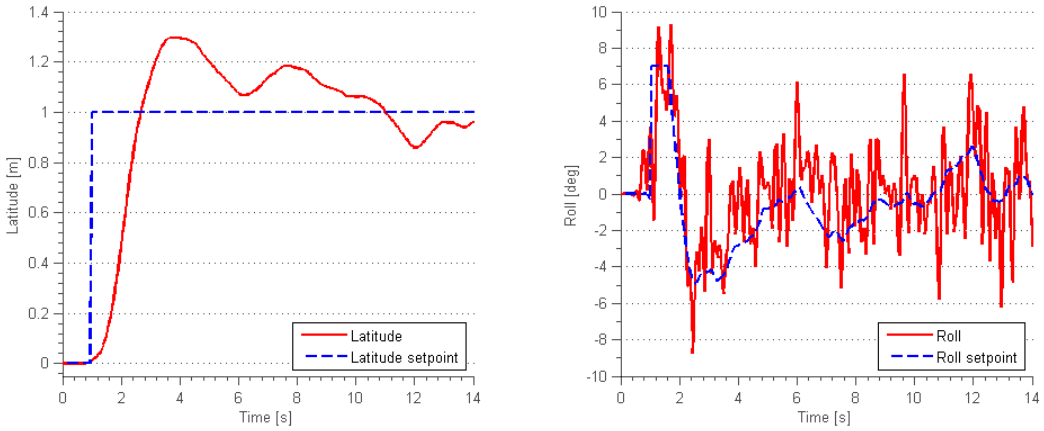
The discrete controllers were simulated using the hexacopter plant model and the resulting position step response and roll angle can be seen in Figure 29. Using a sample time of 50 ms, determined by image analysis execution time on the PandaBoard, the stability margin was decreased, producing large position overshoot. This was handled by increasing K_d. A comparison of the continuous and discrete time controller parameters is presented in Appendix B.



(a) Latitude (b) Roll

Figure 29: Latitude step and corresponding roll angle.

In order to simulate process noise, the hexacopter roll angle model was subjected to additive zero-mean Gaussian noise with a 2 degree standard deviation. The resulting step response and roll angle is presented in Figure 30.



(a) Latitude (b) Roll

Figure 30: Latitude step and corresponding roll angle subjected to noise.

7 Autonomy

Autonomous flight and behavior requires a method for decision making. Numerous approaches to machine learning and artificial intelligence has been attempted in robotics (Wahde, 2011). Here, a simple brain process was developed using select behaviors (Table 1).

Table 1: List of autonomous behaviors.

Behaviour	Description
Hover	Maintain current position and altitude
Manouver	Fly to location set by user
Search	Maintain position, run object recognition
Follow	Follow detected object
Takeoff	Start from ground, take off to user selected altitude
Land	Descend, land and stop motors
Idle	Stay on ground

The activation of these behaviors (i.e. decision making) is governed by a state machine (Figure 31) implemented using MATLAB Stateflow. The transitions between different behaviors are activated using data such as altitude, battery level, object recognition status and user input.

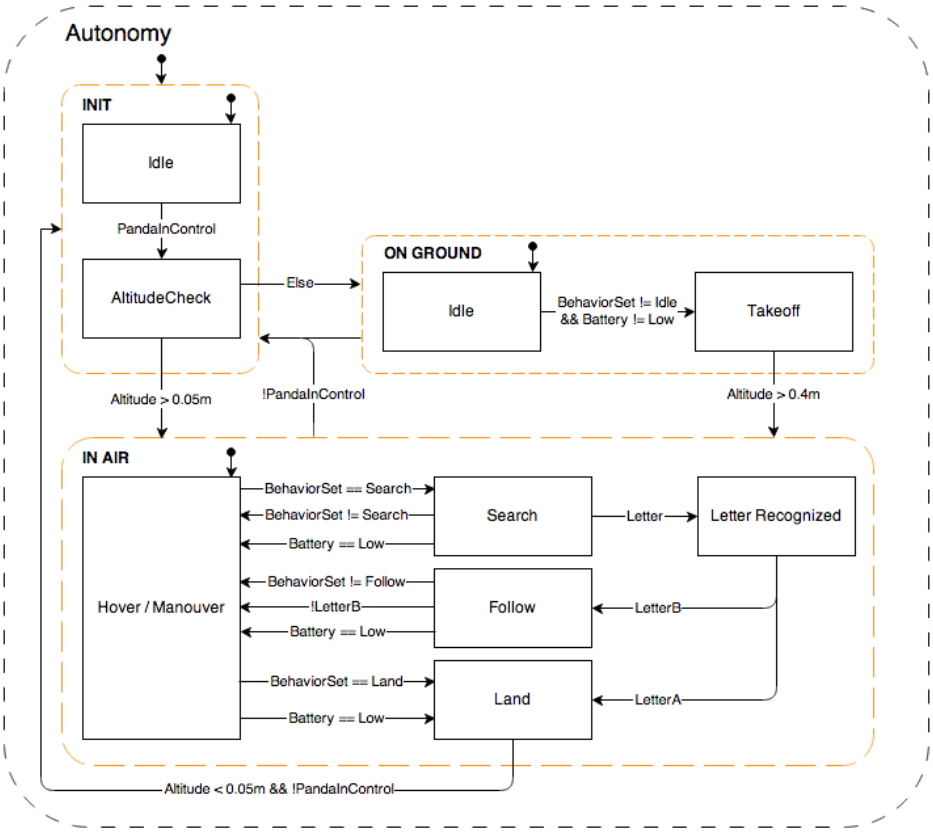


Figure 31: Autonomy state machine.

8 Communication

The PandaBoard native support in Simulink for communication is currently limited to User Datagram Protocol (UDP) over IP-networks (MathWorks, 2013) even though the PandaBoard has both I^2C and UART pins available (Pandaboard.org, 2013). With no documented success of implementing these in Simulink found, the ArduCopter software and hardware was modified to support UDP over Ethernet.

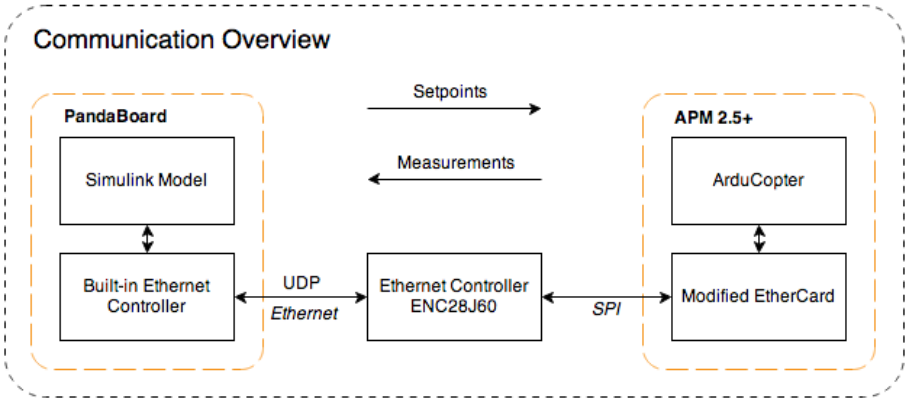
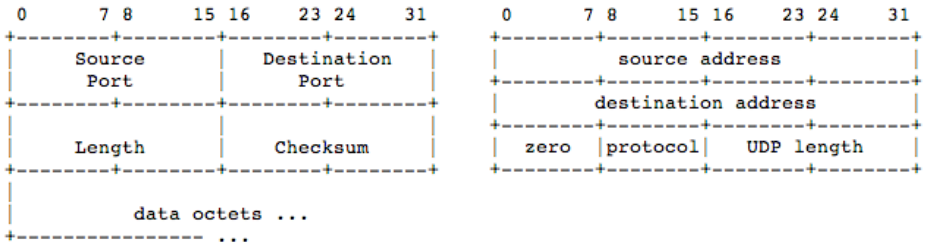


Figure 32: Communication overview.

8.1 User Datagram Protocol

UDP is used as a way for applications to communicate using a small set of protocol mechanisms. It does not guarantee delivery or packet order as the Transmission Control Protocol (TCP) does. UDP requires the Internet Protocol (IP) to be used as the underlying protocol (Postel, 1980).

An UDP packet consists of a source port, destination port, length, checksum and the data (see Figure 33(a)). The last part of the IP packet contains the source address, destination address, UDP protocol flag, the length of the UDP header and data (see Figure 33(b)).



(a) User Datagram Header Format (b) Part of the IP protocol

Figure 33: UDP and IP protocol structure.[8]

[8] http://tools.ietf.org/html/rfc768

8.2 ArduCopter Modifications

The APM hardware was extended with a Ethernet ENC28J60 controller from Microchip. It is a 10Base-T stand-alone controller with a Serial Peripheral Interface bus (SPI) (Microchip Technology Inc., 2012). EtherCard, a driver written i C++ specifically for the Microchip controller and compatible with the Arduino platform (JeeLabs, 2012) was modified and implemented in the ArduCopter code base. The documentation of the code is unsatisfactory, requiring extensive efforts when extending the software. A state machine was developed to be called at 50Hz to send and receive data over UDP, do sanity checks, propagate setpoints and update the status of communication and autonomy to the Andon light.

9 System Verification and Validation

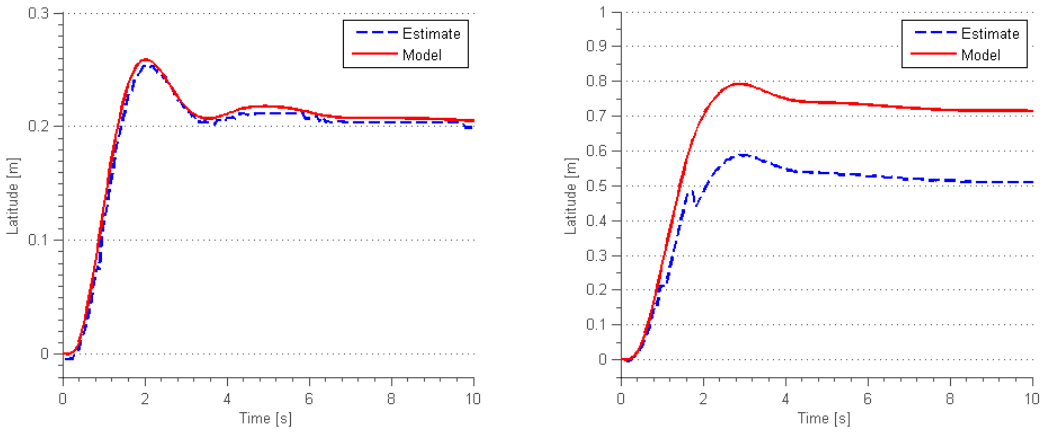
The hexacopter autonomous hovering capability was evaluated by simulating the complete system and by performing test flights using the same vision and controller software parameters. Disturbances observed during test flights were introduced in simulation to assess the accuracy of the hexacopter model.

9.1 Simulation

The UAV system performance was assessed in Simulink simulation using the hexacopter model, the camera simulator and the actual vision and control algorithms to be run on the PandaBoard. The system hover and maneuvering capabilities were tested using a latitude step setpoint and a play mat floor image. Hexacopter yaw and altitude were assumed constant, being controlled by the ArduCopter's internal controller. The position, estimated using template matching, was compared with the hexacopter model position output.

The result of a 0.2 and 0.5 meter position step setpoint is seen in Figure 34. The 0.5 meter step causes the template to move outside the image frame, forcing the Vision block to acquire a new template. The algorithm currently used to do this has the consequence of bad position estimation during the samples required to determine a new template. This explains the notches seen in the estimated position curve and the error in position steady state value.

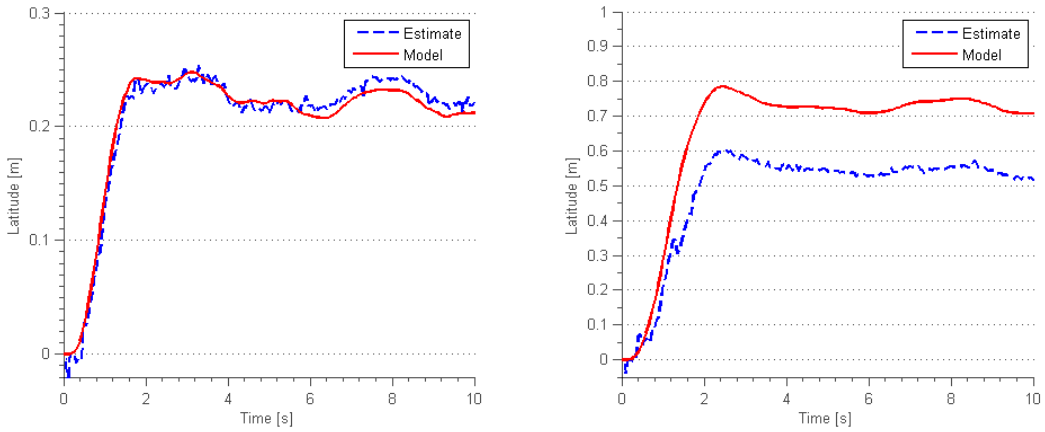
Additive zero-mean Gaussian noise with a 2 degree standard deviation was applied to the roll and pitch angles to evaluate step position control with process disturbances (see Figure 35).



(a) 0.2 meter step

(b) 0.5 meter step

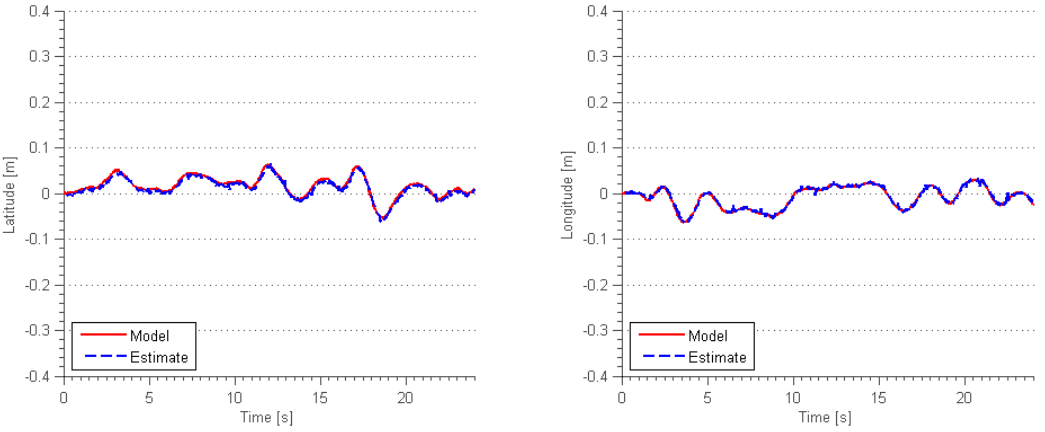
Figure 34: Latitude position step simulation, estimated position and model output.



(a) 0.2 meter step (b) 0.5 meter step

Figure 35: Latitude position step simulation, estimated position and model output. White noise disturbance on roll, pitch, altitude and yaw.

To test hover performance, the hexacopter model was subjected to additive zero-mean Gaussian noise with a 2 degree standard deviation on the roll and pitch angles. The position setpoint was set to zero. Figure 36 shows estimated and actual position deviation during hover.



(a) Latitude (b) Longitude

Figure 36: Position from hover simulation with roll and pitch angles subjected to additive white noise.

9.2 Test Flight

The position estimation and hovering capabilities of the hexacopter were verified using a camera mounted in the ceiling pointing straight down (see a cutout of the camera view in Figure 38). In the test, the hexacopter took off and hovered autonomously. The letter B was mounted to the back of the hexacopter to enable tracking of its true position. This

was achieved using video acquired from the ceiling camera, object recognition theory from Section 5.1 and the camera model from Section 4.3. Maneuvering capability assessment, using a position step setpoint, was omitted in favour of hovering tests.

Figure 37: Hexacopter ready for system evaluation with letter B mounted on its back to enable tracking using ceiling camera.

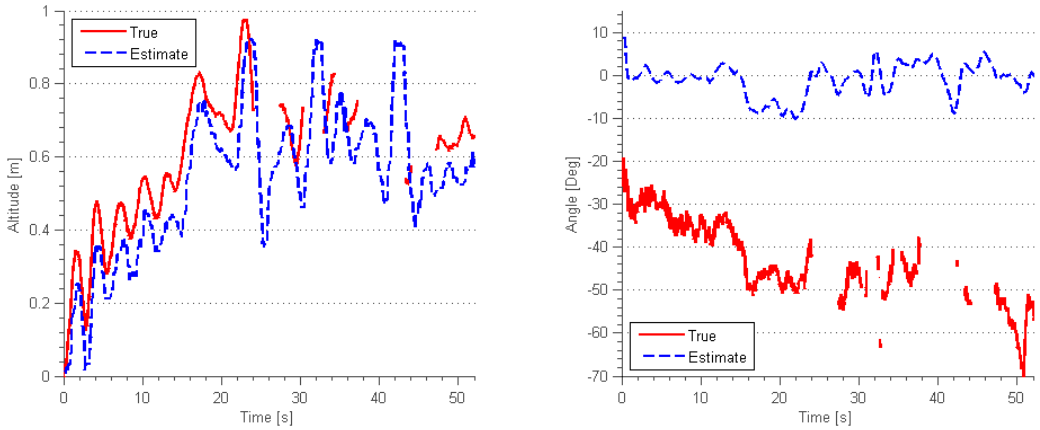
Figure 38: Cutout of the view from ceiling mounted camera. The marker X represents the hexacopter estimated position and the small B its true position.

In every image frame of the video, the letter B was recognized and its bounding box (BBox) was used to estimate the hexacopter's true position and yaw. These values were compared to the hexacopter's own onboard position estimate and yaw measurement.

The altitude of the hexacopter from ground (Figure 39(a)) was estimated using the known dimensions of the letter B, its BBox and Equation (14) solved for h.

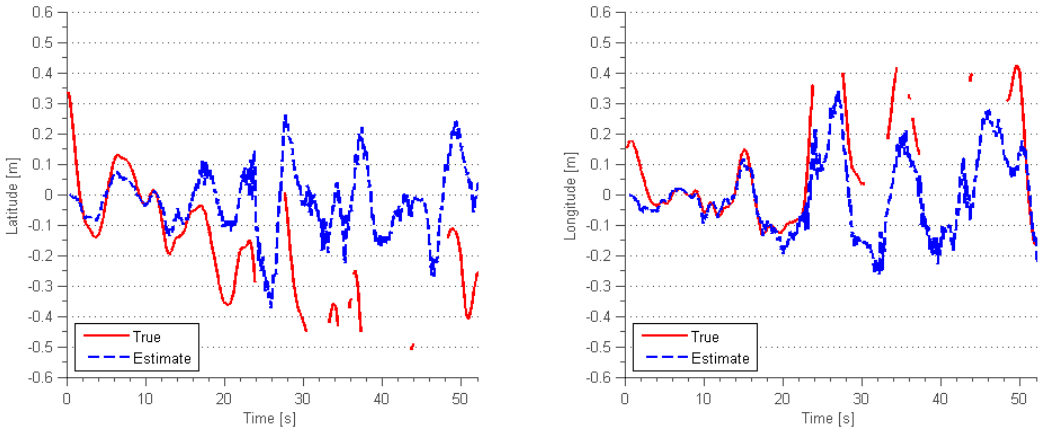
Latitude and longitude position (Figure 40) were estimated using the coordinate of the BBox, the distance from the camera and the camera model equations presented in Section 4.3. The coordinate system was rotated to compensate for the ceiling camera's orientation offset from north.

The yaw was estimated in Figure 39(b) using the orientation of the BBox, the offset angle of the letter about z_B on the hexacopter and the orientation of the ceiling camera angle offset from north.



(a) Altitude, 0.6 m setpoint (b) Yaw, 0 degrees setpoint

Figure 39: Altitude and yaw estimates together with true values obtained using data from ceiling mounted camera. The gaps in data is caused by the hexacopter flying outside the camera view.



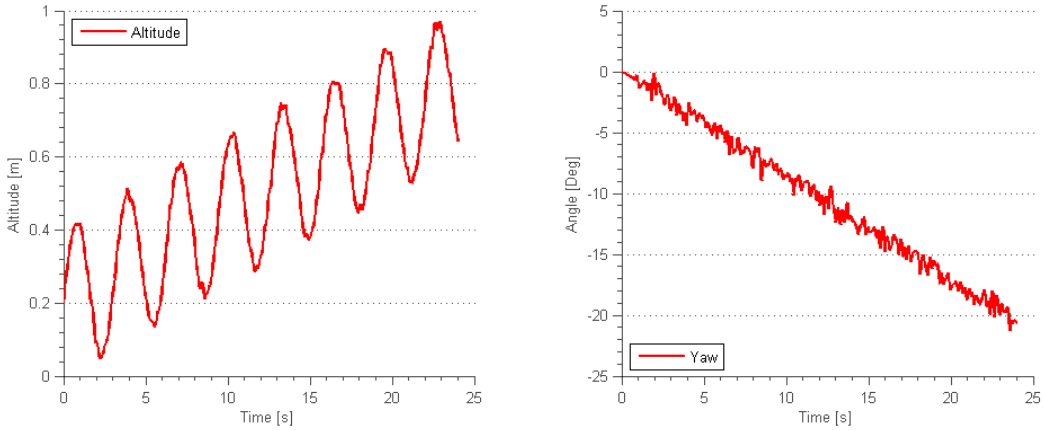
(a) Latitude (b) Longitude

Figure 40: Hexacopter latitude estimation and true position in flight obtained from ceiling mounted camera, 0 m position setpoint. The gaps in data is caused by the hexacopter flying outside the camera view.

During some test flights, the Simulink model running on the PandaBoard locked up or caused lag in the communication for brief periods of times. This causes the hexacopter to drift in its estimation and therefore physical position. However, none of these symptoms affected the results seen in Figure 40.

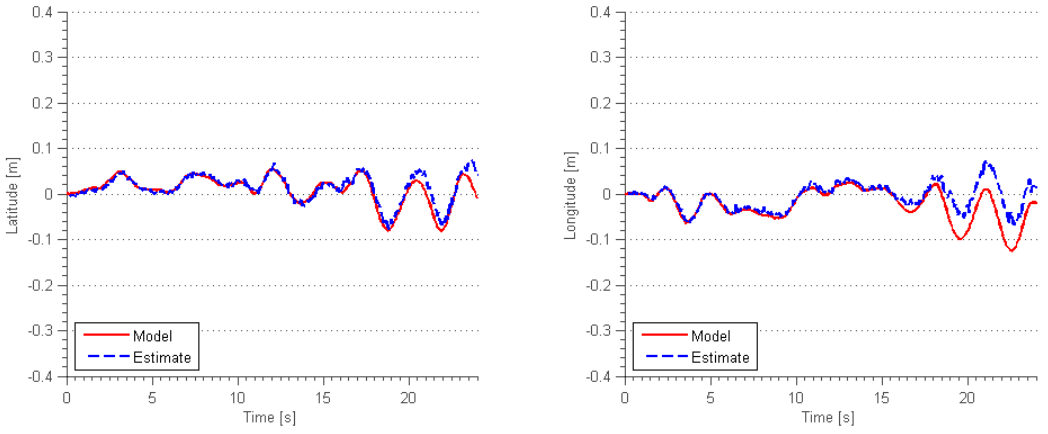
9.3 Simulating Disturbances

During test flight, observations were made to identify what kind of process disturbances were present. It was noted that the altitude was subjected to sinusoidal and drifting disturbances and that the yaw angle was decreasing over time. Disturbance signals were created to mimic this behavior (see Figure 41). In order to examine model accuracy, new hover simulations were conducted with these disturbances present and the results can be seen in Figure 42-44.



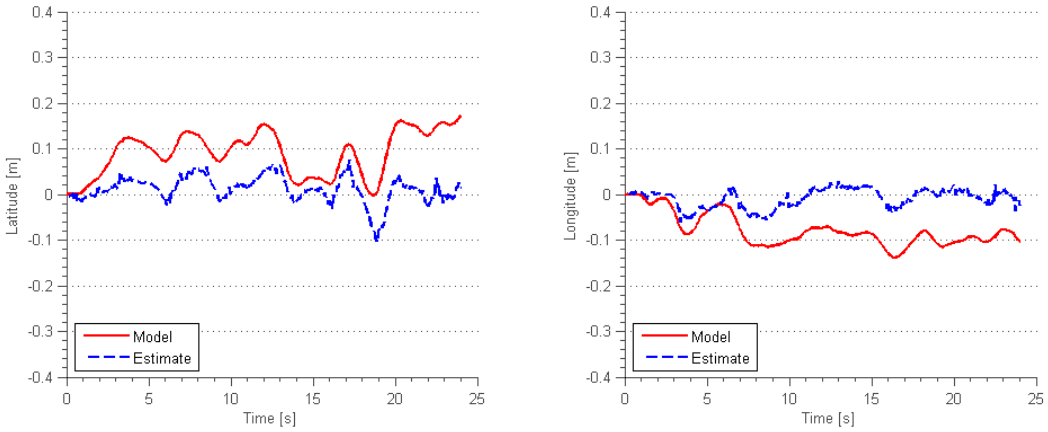
(a) Altitude (b) Yaw

Figure 41: Altitude and yaw disturbance.



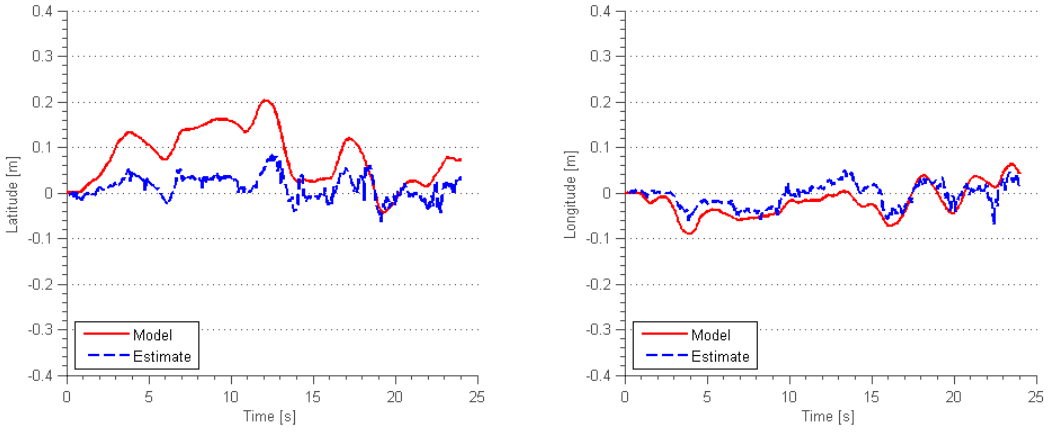
(a) Latitude (b) Longitude

Figure 42: Position from hover simulation with roll and pitch angles subjected to additive white noise, yaw subjected to ramp and white noise disturbances.



(a) Latitude (b) Longitude

Figure 43: Position from hover simulation with roll and pitch angles subjected to additive white noise, altitude subjected to a sinusoidal and white noise disturbance.



(a) Latitude (b) Longitude

Figure 44: Position from hover simulation with roll and pitch angles subjected to additive white noise, yaw subjected to a ramp and white noise disturbance, altitude subjected to a sinusoidal and white noise disturbance.

10 Discussion

A fully autonomous UAV platform requires the proper operation of several subsystems including vision, navigation and control. It can be argued that verification of the complete system is an extensive task, requiring thorough testing of subsystems as well as the complete system, both in simulation and in test flight.

The results obtained in this study are somewhat inconclusive. Thorough testing of the subsystems was put aside in favor of complete system performance. There are however several findings made that can serve as guidelines for future work in the UAV field of study.

In general, the results point out that an autonomous UAV can indeed navigate in an indoor environment using computer vision based navigation, provided that the surfaces of the room are textured. This is in line with the findings of Leishman et al. (2012) and Lange et al. (2009). Position hold is accomplished over a time period in the order of a half minute but bad altitude and yaw hold achieved by the ArduCopter software in its current configuration leads to less than satisfactory position estimation using the template matching image analysis method.

10.1 Model-Based Design

In general, results obtained in simulation matched the actual test flight results. The camera simulator, featuring motion blur and attitude dependence, provided imagery similar to that of the camera mounted underneath the hexacopter during flight and thus, image analysis algorithm development and testing was simplified. The hexacopter model, derived using system identification and physical modelling, enabled offline position control tuning and the parameters obtained here gave actual flight behavior similar to that of the simulation. The 3D-visualization environment provided an intuitive way of assessing actual hexacopter flight performance.

The position model developed using physical modeling seems to have performed well. Future work could perhaps improve accuracy by using system identification to develop this model using position data from the ceiling camera and attitude angles measured by the ArduCopter.

As mentioned, a UAV system consists of multiple complex subsystems. Complete system verification and assessment could be considered difficult and time consuming to achieve by only looking at one subsystem at a time. A holistic approach is a necessity, rendering the model-based design method helpful in the strive for reduced development time and cost while maintaining or improving quality.

10.2 Position Estimation

Simulation results indicate that the use of a downward facing camera for position estimation is arguably a method suited for navigation, providing a floor texture not varying over time and featuring enough detail and contrast is present. Template matching is an intuitive image analysis approach to the problem. Using feature point extraction and matching methods, such as SURF, do however show great promise and should provide a more general solution to the problem, enabling mapping using algorithms such as SLAM (Lee and Song, 2009).

During test flights, template matching performed poorly at low altitudes but the use of a wide angle camera could solve this problem. There were also symptoms of bad position

estimation when the hexacopter exhibited rapid roll or pitch movements, seen during rapid descents. This could be caused by motion blur in the image as well as asynchronous sampling of angles and image, causing bad attitude compensation. A physical camera stabilization rig or a faster shutter time should reduce the amount of motion blur appearing here, arguably affecting the image analysis results in a positive way. Template re-acquisition performed during hexacopter position movement produced an error which could be handled with the use of a constant velocity model.

A method for velocity estimation that could be evaluated for position measurement purposes in future work is the optical flow algorithm. This approach has been attempted in the multirotor UAV community (Kim and Brambley, 2007) and specialised sensors, similar to those present in optical computer mice, can be used to decrease the computational effort.

10.3 Control

Analysis of the control system was complicated by non-deterministic delays introduced by the Simulink UDP block and by the fact that the Linux kernel run on the PandaBoard is non-realtime. More deterministic and stable communication could perhaps be achieved by using low-level protocols such as I^2C or UART. The continuous control system stability analysis presented in Section 6.3 can be improved by instead performing the analysis using the discrete controller.

The plant model derived using system identification should be considered valid only for inputs similar to those tested. A white noise signal ought to be applied in order to excite as many nodes of the system as possible. Here, a telegraph signal was used instead. This input was, however, similar to control inputs used during manual position step control.

The attitude step response, achieved using system identification, does have a quite unexpected appearance with a large overshoot. It is however reasonable to think that the ArduCopter attitude controllers are tuned in a way to produce an intuitive flight behavior, disregarding the characteristics of the step response. Another factor affecting control performance and system identification procedures is turbulence arising from flight in confined indoor spaces.

11 Conclusion

A hexacopter UAV prototype featuring computer vision based navigation developed using model-based design methods has been presented. A number of conclusions are drawn:

- Computer vision based navigation is a navigational strategy suited for use in GPS-denied environments, such as indoors. Using a downward facing camera, floor texture is a factor to take into account when choosing and tuning image analysis algorithms for motion estimation.
- The template matching strategy is applicable for position estimation purposes, shown in simulation and test flights, assuming the floor texture has regions of contrast and structure and that it is non-varying during flight. Improvements must, however, be made in order to achieve reliable navigation. Feature based algorithms, such as corner detection and especially SURF, are promising.
- Object recognition of letters was performed in a satisfactory manner using invariant moments as regional descriptors and a kNN classifier. A good object segmentation is important for proper object recognition. A color segmentation approach using HSV was proven successful.
- Position control was achieved using two cascaded PD controllers, generating a desired velocity and attitude setpoint respectively. The control structure enabled intuitive tuning of parameters and a simple algorithm implementation.
- The model-based design approach using a plant model, achieved through physical modeling and system identification, enabled early verification and validation of control and image analysis algorithms. System behaviour and performance was visualized and assessed in 3D. The plant, position and camera models were all proven sufficiently accurate based on test flights successfully conducted with tuning parameters being left unaltered from simulation. Automatic code generation greatly simplified the process of implementing algorithms developed in Simulink to the PandaBoard. However, not all Simulink blocksets and MATLAB functions are supported for code generation.
- The PandaBoard ES single-board computer features performance adequate for execution of image analysis algorithms at sufficient resolution. The PandaBoard Simulink blocksets does, however, only support Ethernet communications and not low level protocols, such as I^2C and UART.
- The ArduCopter software is open source and can thus be modified according to needs. However, the code documentation is unsatisfactory, requiring extensive effort to extend its functionality.

Bibliography

Ahmed, S. (2010). Model-Based Design Takes Flight. *Electronics Weekly*, 1(2439):44.

Andersson, T., Arver, J., Johansson, P., Karlsson, A., Linder, J., and Lindkvist, S. (2010). Robust Control System for Quadrocopter. Technical report, Chalmers University of Technology, Gothenburg. SSY-225 Design Project in Systems, Control and Mechatronics.

Bay, H., Tuytelaars, T., and Gool, L. (2006). SURF: Speeded Up Robust Features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg.

Chen, T.-W., Chen, Y.-L., and Chien, S.-Y. (2008). Fast image segmentation based on K-Means clustering with histograms in HSV color space. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 322–325.

Choi, E. and Lee, C. (2000). Feature extraction based on the Bhattacharyya distance. In *Geoscience and Remote Sensing Symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 International*, volume 5, pages 2146–2148 vol.5.

DIY Drones (2013). ArduCopter. `https://code.google.com/p/arducopter/`. [Online; accessed April 16 2013].

Doblander, A., Gösseringer, D., Rinner, B., and Schwabach, H. (2005). An Evaluation of Model-Based Software Synthesis from Simulink Models for Embedded Video Applications. *International Journal of Software Engineering & Knowledge Engineering*, 15(2):343 – 348.

Falcone, P. (2010). Modeling and Simulation. Lecture notes distributed at Chalmers University of Technology.

Fleischer, D., Beine, M., and Eisemann, U. (2009). Applying Model-Based Design and Automatic Production Code Generation to Safety-Critical System Development. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 2(1):240–248.

Gonzalez, R. C. and Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

GreenCarCongress (2009). General Motors Developed Two-Mode Hybrid Powertrain With MathWorks Model-Based Design; Cut 24 Months Off Expected Dev Time. `http://www.greencarcongress.com`. [Online; accessed April 9 2013].

Horn, B. K. P. and Schunck, B. G. (1981). Determining Optical Flow. *Artificial Intelligence*, 17:185–203.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187.

Huster, A. and Rock, S. M. (2001). Relative position estimation for intervention-capable AUVs by fusing vision and inertial measurements. In *Proceedings of the 12th International Symposium on Unmanned Untethered Submersible Technology*.

Itseez (2013). OpenCV. `opencv.org`. [Online; accessed May 7 2013].

JeeLabs (2012). EtherCard Readme on GitHub. `https://github.com/jcw/ethercard#readme`. [Online; accessed 2 May 2013].

Kim, J. and Brambley, G. (2007). Dual Optic-flow Integrated Navigation for Small-scale Flying Robots. In *Australasian Conference on Robotics and Automation (ACRA)*.

Kim, J.-H., Sukkarieh, S., and Wishart, S. (2006). Real-Time Navigation, Guidance, and Control of a UAV Using Low-Cost Sensors. In *Field and Service Robotics*. Springer Berlin Heidelberg.

Lange, S., Sunderhauf, N., and Protzel, P. (2009). A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-denied Environments. In *International Conference on Advanced Robotics. ICAR 2009.*, pages 1–6.

Lee, Y.-J. and Song, J.-B. (2009). Visual SLAM in Indoor Environments Using Autonomous Detection and Registration of Objects. In *Multisensor Fusion and Integration for Intelligent Systems*, pages 301–314.

Leishman, R., Macdonald, J., McLain, T., and Beard, R. (2012). Relative navigation and control of a hexacopter. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4937–4942.

Ljung, L. (1988). *System Identification Toolbox For Use with MATLAB*. MathWorks, Natick, MA.

Luukkonen, T. (2011). Modelling and Control of Quadcopter. Technical report, School of Science, Espoo, Finland.

MathWorks (2013). PandaBoard Support from Simulink. `http://www.mathworks.se/academia/pandaboard/`. [Online; accessed April 11 2013].

Mehnert, A. (2012). Image Analysis. Lecture notes in Image Analysis distributed at Chalmers University of Technology.

Microchip Technology Inc. (2012). ENC28J60 Data Sheet Stand-Alone Ethernet Controller with SPI Interface. `http://ww1.microchip.com/downloads/en/DeviceDoc/39662e.pdf`. [Online; accessed 2 May 2013].

Pandaboard.org (2013). PandaBoard ES. `http://pandaboard.org/content/pandaboard-es`. [Online; accessed April 11 2013].

Postel, J. (1980). RFC 768. `http://tools.ietf.org/html/rfc768`. [Online; accessed 30 April 2013].

Rosten, E. and Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In Leonardis, A., Bischof, H., and Pinz, A., editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer.

Slack, N., Chambers, S., and Johnston, R. (2010). *Operations management*. Financial Times, Prentice Hall, Harlow.

Sonka, M., Hlavac, V., and Boyle, R. (2008). *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering.

Waharte, S. and Trigoni, N. (2010). Supporting Search and Rescue Operations with UAVs. In *International Symposiom on Robots and Security*.

Wahde, M. (2011). Autonomous Agents. Lecture notes in Autonomous Agents distributed at Chalmers University of Technology.

Weisstein, E. W. (2013). Central Moment, MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/CentralMoment.html`. [Online; accessed 23 April 2013].

West, B. (2009). Model-Based Design. *ECN: Electronic Component News*, 53(10):26–27.

A System Identification Parameters

Table 2: Roll model output fit percentage for different number of zeros and poles. With and without delay, left and right table respectively. Best overall fit in bold.

z \ p	0	1	2
0	61	51	57
1	-	75	**82**

z \ p	0	1	2
0	61	61	61
1	-	72	61

Table 3: Pitch model output fit percentage for different number of zeros and poles. With and without delay, left and right table respectively. Best overall fit in bold.

z \ p	0	1	2
0	63	41	16
1	-	46	**78**

z \ p	0	1	2
0	63	64	64
1	-	64	57

Table 4: Roll model parameters.

Parameter	Value
K_p	0.44
T_z	2.53
T_{p1}	0.88
T_{p2}	0.21
T_d	0.05

Table 5: Pitch model parameters.

Parameter	Value
K_p	0.47
T_z	2.14
T_{p1}	0.74
T_{p2}	0.24
T_d	0.04

B Control Parameters

Table 6: Continuous velocity PD parameters.

Parameter	Value
K_p	1
K_d	0.2

Table 7: Discrete velocity PD parameters.

Parameter	Value
K_p	1
K_d	1.8

Table 8: Continuous attitude P parameters.

Parameter	Value
K_p	10

Table 9: Discrete attitude P parameters.

Parameter	Value
K_p	10