Czech Technical University in Prague
Faculty of Electrical Engineering

Master Thesis

Visual servoing for a quadcopter flight control

Prague, 2012

Author: Michal Podhradský
Supervisor: Zdeněk Hurák

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Michal Podhradský**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Diploma Thesis: **Visual servoing for a quadcopter flight control**

Guidelines:

The goal of this thesis is to explore potentials of visual servoing techniques for flight control of a small indoor quadcopter (four-rotor helicopter). The particular tasks that should be addressed in this project are

1.) take-off/landing control,
2.) obstacle avoidance,
3.) ground velocity estimation.

The work will consist of a comprehensive survey of the state of the art in this domain followed by development of original solutions for a few selected flight control tasks and concluded by flight experiments with a real quadcopter. The work will be supervised jointly by Dr. Zdenek Hurak from Czech Technical University in Prague and Dr. YangQuan Chen from Utah State University.

Bibliography/Sources:

[1] P.I. Corke. Robotics, Vision and Control: Fundamental Algorithms in MATLAB. Springer, 1st ed., 2011.
[2] J. Dvořák. Micro Quadrotor: Design, Modelling, Identification and Control. Diploma thesis CTU in Prague, 2011.

Diploma Thesis Supervisor: Ing. Zdeněk Hurák, Ph.D.

Valid until the summer semester 2012/2013

[illegible]

prof. Ing. Michael Šebek, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 12, 2012

Declaration

I declare that I have created my Diploma Thesis on my own and I have used only literature cited in the included reference list.

In Prague, _______________ ____________________

signature

Acknowledgement

To all nice people on this planet who make love rather than war.

Abstrakt

V této práci jsou představeny možnosti použití metody *visual servoing* pro řízení bezpilotních prostředků s kolmým vzletem a přistáním (VTOL). Jsou zmíněny možné scénáře nasazení těchto prostředků, a také nejnovější používané *quadkoptéry* (bezpilotní čtyř rotorový vrtulník). Po nezbytném teoretickém úvodu jsou představeny návrhy řešení pro důležité letové úkony, zejména vzlet a přistání, vyhýbání se překážkám a odhad polohy bezpilotního prostředku.

Na závěr je představen autopilot pro kolmý vzlet a přistání, založený na metodě *visual servoing.* Tento autopilot je detailně teoreticky popsán, koncept je experimenátlně ověřen a poté implementován na skutečném quadrotoru vyvíjeném Dr. YangQuanem Chenem z Utah State University, Center for Self-Organizing Intelligent Systems.

Abstract

In this thesis the potential of visual servoing techniques for vertical take-off and landing (VTOL) UAV platforms is explored. An overview of UAV applications and state-of-the-art quadrotors is provided, backed up by the necessary theoretical background. Visual servoing based solutions of the important flight tasks (such as vertical take-off and landing, obstacle avoidance and attitude estimation) are presented.

A novel vision based autopilot for vertical take-off and landing is developed, experimentally verified and implemented on a medium-size outdoor quadrotor platform provided by Dr. YangQuan Chen from Utah State University, Center for Self-Organizing Intelligent Systems.

Contents

Nomenclature **viii**

1 Introduction **1**

1.1 UAV applications . . . 2

1.2 Quadrotors . . . 3

1.2.1 Indoor quadrotors . . . 4

1.2.2 Outdoor quadrotors . . . 5

1.3 Modelling of quadrotor dynamics . . . 6

1.3.1 General Moments and Forces . . . 7

1.3.2 Equations of motion . . . 9

1.3.3 State-space model . . . 9

1.3.4 Octorotor modelling . . . 11

2 Visual servoing in theory **12**

2.1 Visual servoing . . . 12

2.1.1 Image based visual servoing . . . 13

2.1.1.1 Perspective camera model . . . 14

2.1.1.2 Spherical camera model . . . 14

2.1.1.3 Depth estimation . . . 15

2.1.2 Position based visual servoing . . . 16

2.2 Visual odometry . . . 17

2.3 Visual SLAM . . . 19

2.4 Optical Flow . . . 21

3 Visual servoing in applications **23**

3.1 Vertical take-off and landing . . . 23

3.1.1 Landing spot selection . . . 24

3.1.2 Stabilization and descend 24
3.1.3 Vision-based autopilot 25
3.2 Ground speed estimation 27
3.3 Obstacle avoidance 28
3.4 Attitude estimation 29
3.5 Summary 30

4 VTOL autopilot **31**
4.1 Autopilot design 31
4.1.1 Landing autopilot 32
4.1.2 Take-off autopilot 33
4.1.3 Safety considerations 35
4.2 Optical Flow Estimation 36
4.2.1 Camera lenses 37
4.2.2 Focus of Expansion 39
4.2.3 Recommendation 45
4.3 Target detection 46
4.4 Hardware 47
4.4.1 Camera and the lens 48
4.4.2 On-board computer 49
4.4.3 Ultrasonic proximity sensor 49
4.5 Software 50
4.6 Summary 51

5 Flight tests and Conclusion **53**
5.1 Preliminary results 53
5.1.1 Calibration procedure 53
5.1.2 Improved target detection 54
5.1.3 Optical flow calculation 54
5.2 Conclusion 56

References **58**

A AggieVTOL quadrotor **I**

B Camera setup **IX**

C Panda board set-up **XIII**

D Landing pad design **XVI**

E Contents of the CD **XXI**

Nomenclature

Variable	Description
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle

Abbreviation	Description
AE	Attitude Estimation
FOE	Focus Of Expansion
FOOF	Fractional Order Optical FLow
FOV	Field Of View
GCS	Ground Control Station
CoG	Centre of Gravity
GSE	Ground Speed Estimation
HS	Horn-Schunck optical fow method
IBVS	Image Based Visual Servoing
IMU	Inertial Navigation Unit
LKT	Lukas-Kanade Tracker (optical flow method)
OA	Obstacle Avoidance
OF	Optical Flow
PBVS	Position Based Visual Servoing
V-SLAM	Visual Simultaneous Localisation and Mapping
VO	Visual Odometry
VS	Visual Servoing

Chapter 1

Introduction

The aim of this work is to explore the potential of visual servoing techniques for flight control of unmanned aerial vehicles (UAVs), which have an ability of vertical take-off and landing (VTOL). As we are talking mostly about VTOL platforms (quadrotors), the terms UAV and VTOL are to be used interchangeably. It is important to note that quadrotors and n-rotors are currently de facto only VTOL UAV in operation, although different concepts have been already proposed.[1] The potential applications for visual servoing are identified, thoroughly studied and the most promising ones are experimentally evaluated.

This work is organized as follows: In the beginning, a brief introduction of state of the art UAVs and their possible applications is provided, followed by a description of quadrotor dynamics in an extend necessary for this thesis. In Chapter 2 theoretical concepts of visual servoing are described, and in Chapter 3 these concepts are applied on the most promising VTOL tasks. As will be further explained, the most promising and most important application of visual servoing is take-off and landing, and whole Chapter 4 is dedicated to a development of a vision based VTOL autopilot. Chapter 5 shows results of real flight tests and evaluation of the applied visual servoing techniques.

When developing a solution for the selected flight tasks, first is a theoretical description of the problem. Second is a development of a software prototype in Matlab (so called "proof of concept" – POC) to prove the usability of suggested methods.[2] Finally (when applicable), the prototype is transferred into C++ program which can be run on-board on a real UAV.

[1] *http://www.gizmag.com/flying-wing-vtol-uav/13962/*

[2] Matlab is especially useful for POC as it allows user to focus on algorithms rather than on implementation issues.

1.1 UAV applications

Nowadays UAVs are already established in many areas of human life and we can expect that their presence will only increase. Naturally, more developed are fixed wing UAVs, firstly due to their relative simplicity comparing to VTOL platforms (also known as "rotary wings"), secondly due to much wider knowledge base for fixed wing air planes (wing profiles, air plane aerodynamic properties etc.), and lastly thanks to their inherently higher reliability (in case of a motor failure, fixed wing can still glide and safely land, whereas rotary wing cannot). However, VTOL platforms are catching up as is to be shown in the rest of this chapter.

Not surprisingly, first area of UAV's application is military. UAV drones Shadow[3] and Raven[4] serve American Army and are in operation in Iraq and Afghanistan. Both drones are exported to allied armies, for example Czech army uses them as a replacement for older UAVs Sojka and Mamok. UAVs in military are mostly used for area reconnaissance and surveillance.[5] So far none of VTOL platform has been used in regular army operation, although there is an ongoing extensive research in this area.[6] [7]

Second large area of application is security and civilian surveillance. Law enforcement, crime scene mapping and crowd surveillance are amongst the most promising topics.[8] As will be shown later, there are already VTOL platforms capable of fulfilling such tasks. It is worth noting that complicated urban environment, where will police and other authorities mostly use UAVs, naturally favours VTOL platforms that can hover on a spot and provide continuous monitoring of the area.

Big attention is paid to disaster awareness, response and monitoring. Using UAVs by fire fighters and rescue services undoubtedly brings benefit to civilians, unlike to the previous applications. Recently, Draganflyer X6 equipped with thermal and RGB cameras helped fire fighters in Grand Junction, Colorado to localize focus of fire and provided an overview of the whole area.[9] Another example is an ongoing project of Centre for Self Organizing and Intelligent Systems(CSOIS[10]) at Utah State University. The aim of this project is to monitor a large area hit by a disaster (for example an earthquake) using

[3]Codename RQ-7 Shadow, manufacturer AAI Corporation

[4]Codename RQ-11 Raven, manufacturer AeroVironment, Inc.

[5]Raven in operation: *http://www.youtube.com/watch?v=4p0VhNC0CLY*

[6]*http://vodpod.com/watch/2660163-robo-bugs-used-in-us-army-as-micro-air-vehicles*

[7]*http://robobees.seas.harvard.edu/*

[8]See *http://www.uavm.com/uavapplications/firepolice.html* for more details

[9]For more details and video footage see: *http://youtu.be/JpS21_5rvz8*

[10]*http://www.csois.usu.edu/*

fixed wing UAV platform. The flight tests are to be performed in New Zealand in July 2012.

Last large and promising area of applications for UAVs is Personal Remote Sensing. As computers transformed from large and heavy boxes into truly personal devices, similar development is expected to happen with remote sensing, once reliability and "trustworthiness" of UAVs increase and the costs decrease. Remote sensing of agricultural land, wetlands, riparian systems and other areas provide an excellent opportunity for both VTOL and fixed wing platforms, as can be seen in one of the previous CSOIS projects AggieAir[11]. VTOL platforms can be used for wide variety of civilian tasks, among the unusual ones we should mention "Taco delivery."[12]

The deployment of UAVs is currently (Spring 2012) limited by law restrictions, as Federal Aviation Administration (FAA) in the USA and its counterparts in different countries haven't developed a legal framework for UAV flights yet, which obviously are not Radio-Controlled (RC) flights and thus RC flight framework cannot apply. Considered regulations cover for example "pilot" licences for UAV operators, designated flight zones, plane identification[13] and so on. Once the legal framework and rules are settled down, fears from UAV misuse can be calmed, as for example no UAV will be allowed to fly on a certain area without a proper permission.

1.2 Quadrotors

Using only one propeller for propulsion of a rotary wing platform brings a problem how to counteract torque and how to control the attitude of the platform. This is the issue for all helicopters, and although the problem as well as a solution is well know, it is usually very complicated to develop cyclic and collective mechanisms small enough to fit on an UAV. Using two rotors eliminates unwanted torque, but the mechanical problems are still present. Three and four rotor platform are in favour, because they both eliminate unwanted torque (when the opposite propellers rotate in opposite directions) and are mechanically simple. They are using blades with fixed angle and changing their attitude using a different mechanism as shown in Section 1.3. Analogical to quadrotors

[11] *http://aggieair.usu.edu/index.html*

[12] http://www.aero-news.net/index.cfm?do=main.textpost&id=7af13aa2-f9e6-48cd-8f76-d20925cd1271

[13] i.e. tail number as used on civilian air planes, see *http://en.wikipedia.org/wiki/Aircraft_registration*

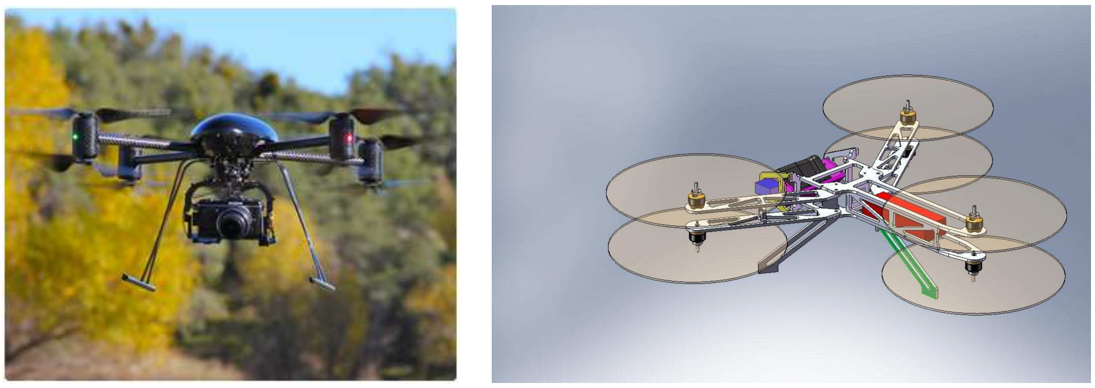


Figure 1.1: Left: Octorotor from Draganfly (*http://www.draganfly.com*)
Right: Hexarotor (*http://diydrones.com*)

are octorotors and hexarotors, with counter rotating propellers as is shown in Figure 1.1

Because propellers in general are more efficient at lower angular velocities, having more propellers allows octorotors and hexarotors run more efficiently while having the same nominal thrust as their quad and tri-rotor counterparts. Besides efficiency, having more propellers mean higher maximal thrust and thus a capacity to carry a heavier payload. In further text, when referring to quadrotors, also octorotors are meant, unless stated otherwise.

A typical quadrotor is equipped with an inertial navigation unit (3 accelerometers, 3 gyroscopes, 3 magnetometers) for attitude determination, a barometer (outdoor) or an ultrasonic proximity sensor (indoor) for altitude measurements and optionally they come with a camera or GPS receiver.

1.2.1 Indoor quadrotors

Quadrotors for indoor use cannot use GPS for absolute positioning and magnetometers provide noisy measurements due to disturbed local magnetic field. However they take benefit from absence of wind gusts, from relatively stable light conditions (useful for vision systems), and their mission duration is usually shorter than their outdoor counterparts. There are already many companies producing VTOL platforms, with comparable performance. For example Ascending Technologies GmBh,[14] whose platforms were used

[14] *http://www.asctec.de/*

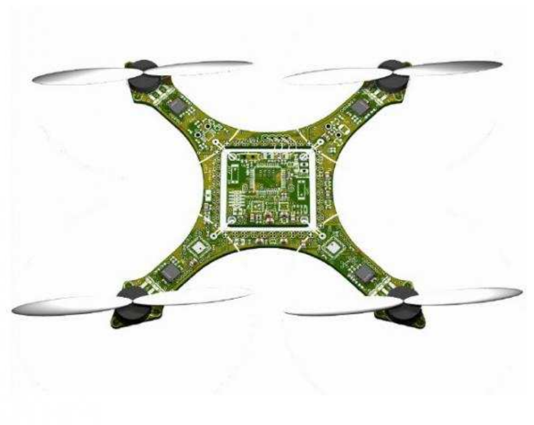


Figure 1.2: NANOQUAD. Prototype of NanoQUad (a very small quadcopter) developed at AA4CC by Jaromír Dvořák. Copyright(c) 2012 by AA4CC.

in 7th European Union Framework programme project sFly[15]. A small quadrotor shown in Figure 1.2 was developed at the Czech Technical University in Prague at AA4CC centre.[16]. Even smaller UAV platform is developed by Harvard university in Robobee[17] project, promising a small artificial insect only a few centimetres long.

Using external reference (for example VICON[18] motion capture system), an advanced control algorithms can be applied to control a lone quadrotor as well as whole swarm of them. State of the art consists of aggressive manoeuvres (flips, flight through windows, perching[19]), formation flying[20] and cooperative behaviour of quadrotor swarm (building a predefined structure[21])[38].

1.2.2 Outdoor quadrotors

Outdoor applications generally require quadrotors that are more durable, take higher payload and can fly on longer missions than their indoor counterparts[41]. Absolute positioning is provided via GPS receiver. Probably the best commercially available VTOL

[15] *http://www.sfly.org/*
[16] *http://aa4cc.dce.fel.cvut.cz/*
[17] *http://robobees.seas.harvard.edu/*
[18] *http://www.vicon.com/*
[19] *http://www.youtube.com/watch?v=MvRTALJp8DM*
[20] *http://www.youtube.com/watch?v=-cZv5oKABPQ&feature=related*
[21] *http://www.zeitnews.org/robotics/flying-robots-build-a-6-meter-tower.html*

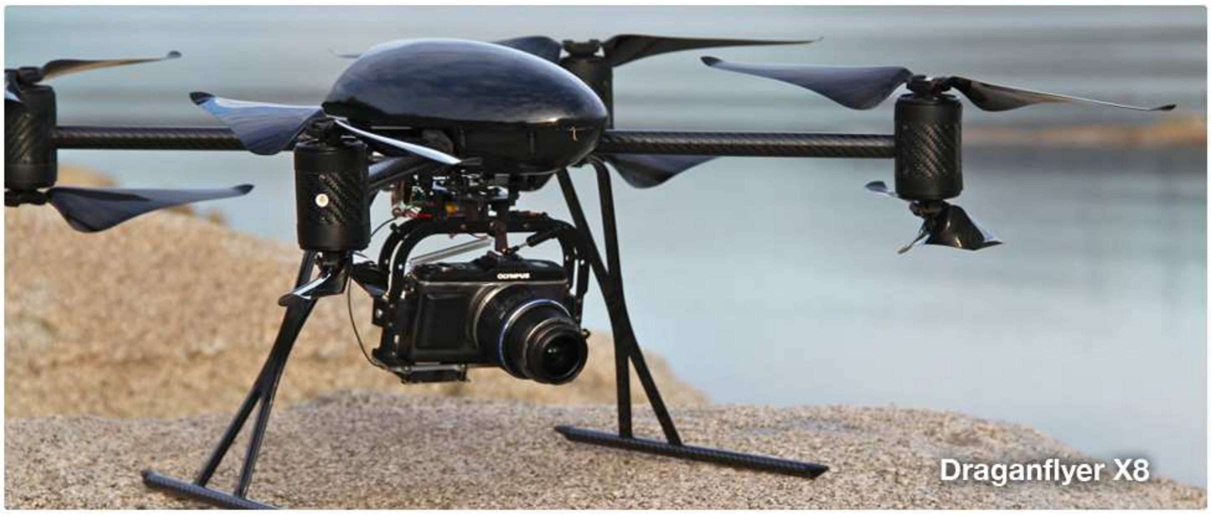


Figure 1.3: Draganflyer X8 (*http://www.draganfly.com/*)

platform is Draganflyer[22] shown in Figure 1.3 in eight rotor configuration. Maximum payload is 800g, the platform has an attitude stabilization, and can follow predefined way points.

A VTOL platform is also developed at CSOIS. Full specification of this platform is provided in Appendix A. It is worth mentioning that this platform is designed to carry a commercial RGB camera for remote sensing and is able to automatically follow way-points in a similar manner as the aforementioned Draganflyer X8.

1.3 Modelling of quadrotor dynamics

This section presents basic quadrotor dynamics, as well as control concepts. It is based on [2], where a complete derivation of the following equations can be found. Comprehensive derivation of the equations can be also found in [16]. The basic idea of quadrotor movement is shown in Figure 1.4. As can bee seen, a quadrotor is mechanically simple in comparison with a common helicopter. Movement in horizontal plane is achieved by tilting the platform whereas vertical movement is achieved by change in total thrust. However, quadrotor is still an under-actuated vehicle, which arises certain difficulties with control design.

A coordinate frame of the quadrotor is shown in Figure 1.5. The developed model is

[22]Draganfly innovation Inc. *http://www.draganfly.com/*

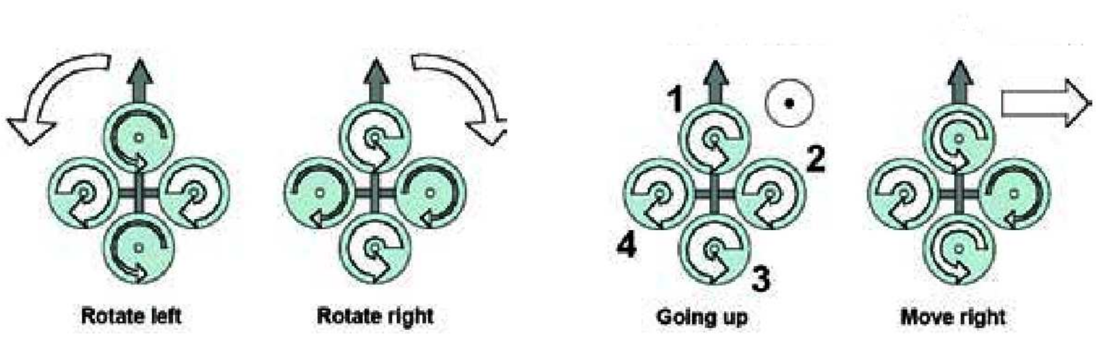


Figure 1.4: The quadrotor concept. The width of the arrows is proportional to the propellers' angular speed[2]

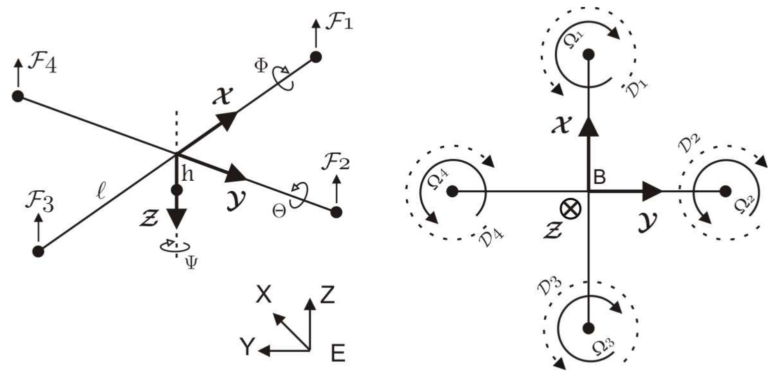


Figure 1.5: Quadrotor coordinate system[2]

based on the following assumptions[2]:

- The structure is supposed to be rigid
- The structure is supposed to be axis symmetrical.
- The centre of gravity (CoG) and the body fixed frame origin are assumed to coincide.
- The propellers are supposed to be rigid.
- Thrust and drag are proportional to the square of propeller's speed.

1.3.1 General Moments and Forces

The forces acting upon a quadrotor are provided below. J_r is a rotor inertia, T is thrust force, H is hub force (a sum of horizontal forces acting on blade elements), Q is a drag moment of a rotor (due to aerodynamic forces), R_m is a rolling moment of a rotor. Ground effect, according to [2] becomes important when $z/R \leq 1$(e.g. a ratio of vertical distance from the ground to propeller radius). For a platform described in Appendix A this happens only few centimetres above the ground (due to relatively high landing gear), thus ground effect can be neglected.

Rolling moments:

body gyro effect	$\dot{\theta}\dot{\psi}(I_{yy} - I_{zz})$
rolling moment due to forward flight	$(-1)^{i+1} \sum_{i=1}^{4} R_{mxi}$
propeller gyro effect	$J_r \dot{\theta} \Omega_r$
hub moment due to sideward flight	$h \sum_{i=1}^{4} H_{yi}$
roll actuators action	$l(-T_2 + T_4)$

Pitching moments:

body gyro effect	$\dot{\phi}\dot{\psi}(I_{zz} - I_{xx})$
hub moment due to forward flight	$h \sum_{i=1}^{4} H_{xi}$
propeller gyro effect	$J_r \dot{\phi} \Omega_r$
rolling moment due to sideward flight	$(-1)^{i+1} \sum_{i=1}^{4} R_{myi}$
pitch actuators action	$l(T_1 - T_3)$

Yawing moments:

body gyro effect	$\dot{\theta}\dot{\phi}(I_{xx} - I_{yy})$
hub force unbalance in forward flight	$l(H_{x2} - H_{x4}$
inertial counter − torque	$J_r \dot{\Omega}_r$
hub force unbalance in sideward flight	$l(-H_{y1} + H_{y3})$
counter torque unbalance	$(-1)^i \sum_{i=1}^{4} Q_i$

Where l stands for a distance between the propeller axis and CoG, h is a vertical distance from centre of propeller to CoG, Ω_r is an overall residual propeller angular speed and I is moment of inertia. Note that the DC motor dynamics is described by a first-order transfer function.

1.3.2 Equations of motion

The equations of motion are derived as follows, using moments and forces described in Section 1.3.1. Note that g is a gravitational acceleration and m represents a mass of the rigid body.

$$\begin{aligned}
I_{xx}\ddot{\phi} &= \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h\sum_{i=1}^{4} H_{yi} + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi} \\
I_{yy}\ddot{\theta} &= \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(T_1 - T_3) + h\sum_{i=1}^{4} H_{xi} + (-1)^{i+1}\sum_{i=1}^{4} R_{myi} \\
I_{zz}\ddot{\phi} &= \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + J_r\Omega_r + (-1)^{i}\sum_{i=1}^{4} Q_i + l(H_{x2} - H_{x4}) + l(-H_{y1} + H_{y3}) \\
m\ddot{z} &= mg - (\cos\psi\cos\phi)\sum_{i=1}^{4} T_i \\
m\ddot{x} &= (\sin\phi\sin\psi + c\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} \\
m\ddot{y} &= (-\cos\psi\sin\phi + s\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi}
\end{aligned} \tag{1.1}$$

1.3.3 State-space model

Using equations 1.1 we develop a non-linear state space model in form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ where $\mathbf{x}$ is a state vector and $\mathbf{u}$ is an input vector.

$$\mathbf{x} = [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad z \quad \dot{z} \quad x \quad \dot{x} \quad y \quad \dot{y}]^T \tag{1.2}$$

$$\mathbf{u} = [u_1 \quad u_2 \quad u_3 \quad u_4]^T \tag{1.3}$$

States are mapped as:

$$\begin{aligned}
x_1 &= \phi & x_2 &= \dot{x_1} = \dot{\phi} \\
x_3 &= \theta & x_4 &= \dot{x_3} = \dot{\theta} \\
x_5 &= \psi & x_6 &= \dot{x_5} = \dot{\psi} \\
x_7 &= z & x_8 &= \dot{x_7} = \dot{z} \\
x_9 &= x & x_{10} &= \dot{x_9} = \dot{x} \\
x_{11} &= y & x_{12} &= \dot{x_{11}} = \dot{y}
\end{aligned}$$

Input vector is mapped as:

$$\begin{aligned}
u_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
u_2 &= b(-\Omega_2^2 + \Omega_4^2) \\
u_3 &= b(\Omega_1^2 - \Omega_3^2) \\
u_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)
\end{aligned} \tag{1.4}$$

where b is a thrust coefficient and d a drag coefficient. Putting equations 1.1 – 1.4 together, we obtain:

$$f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1u_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2u_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}a_5 + b_3u_4 \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{1}{m}u_1 \\ \dot{x} \\ u_x\frac{1}{m}u_1 \\ \dot{y} \\ u_y\frac{1}{m}u_1 \end{pmatrix} \tag{1.5}$$

Where:

$$\begin{aligned}
a_1 &= (I_{yy} - I_{zz})/I_{xx}) & b_1 &= l/I_{xx} \\
a_2 &= J_r/I_{xx} & b_2 &= l/I_{yy} \\
a_3 &= (I_{zz} - I_{xx})/I_{yy}) & b_3 &= 1/I_{zz} \\
a_4 &= J_r/I_{yy} & & \\
a_5 &= (I_{xx} - I_{yy})/I_{zz})
\end{aligned}$$

$$\begin{aligned}
u_x &= (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\
u_y &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)
\end{aligned}$$

This model is general enough to be used for a controller design for almost all sizes of quadrotors with satisfactory results. However, it does not cover all aerodynamics effects. First possible improvement of this model is to include blade flapping dynamics as suggested in [42] and deeply analysed in [5].

Another issue is that certain parameters are changing according to the flight conditions (for example a tilted rotor in a forward flight produces a different lift than during hovering)[31]. To provide a precise model, these changes should be taken in account too.

Identification of the model is addressed in [14] and [16].

1.3.4 Octorotor modelling

Derivation of equations describing octorotor dynamics is identical to the quadrotor case as described in 1.3.1 and 1.3.2. We just have to be careful about direction of forces and momentum produced by each rotor. Octorotors have four coupled counter rotating rotors at end of each beam, thus the net torque from a couple of counter rotating rotors is zero. As a result, the movement in horizontal plane is achieved in a slightly different way than for quadrotors.

To rotate a quadrotor, two opposite rotors increase their angular velocity whereas the other two slow down as is demonstrated in Figure 1.4. It is important that the total thrust remains the same. In case of an octorotor, only one rotor in the pair increases its velocity while the other one in the pair slows down (according to the desired direction of rotation), so the total thrust is constant. Tilting is achieved in the same way as in case of a quadrotor.

Chapter 2

Visual servoing in theory

In this chapter a necessary theoretical background of visual servoing, visual odometry, and visual simultaneous localisation and mapping (V-SLAM) is provided. Finally a few words about optical flow are added. If the reader is already aware of the methods presented, it is possible to skip this chapter and move directly on visual servoing applications in Chapter 3.

Visual servoing is an important concept of a robot control based on visual information (e.g. a camera image). Other two concepts (visual odometry in Section 2.2 and visual SLAM in Section 2.3) are aimed at localisation and pose estimation of a robot in an unknown environment, based also on visual information. Although not directly related to a control, they can be thought of as a subset of Position Based Visual Servoing (Section 2.1.2). Optical flow (Section 2.4), on the other hand, is related to Image Based Visual Servoing (Section 2.1.1) as it extracts information directly from the 2D image.

2.1 Visual servoing

Visual servoing (VS) refers to the use of visual information (gathered for example by a camera) to control the motion of a robot. We assume that the camera is fixed at the robot end-effector (or more precisely rigidly mounted at the flying platform), which is commonly known as *eye-in-hand* system[7].

The aim of VS is to minimize an error $\mathbf{e}(t)$, defined as[7]:

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^* \tag{2.1}$$

where $\mathbf{m}(t)$ is a set of image measurements (e.g. image coordinates of points of interest), $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$ is a vector of k visual features, in which $\mathbf{a}$ is a set of parameters representing a potential additional knowledge about the system (e.g. camera intrinsic parameters), and vector $\mathbf{s}^*$ represents the desired values of the features (e.g. position in the image plane). There are two main approaches how to define $\mathbf{s}$ - Image Based Visual Servoing (IBVS) defines $\mathbf{s}$ to represent a set of features immediately available in the image (e.g. image plane coordinates of tracked points), whereas Position Based Visual Servoing (PBVS) defines $\mathbf{s}$ by means of a set of 3D parameters, which has to be estimated from the image.

Assume we want to design a velocity controller with a simple proportional control. Define spatial velocity of the camera as $\mathbf{v}_c = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$. Then we can derive a the relationship between $\dot{\mathbf{s}}$ and $\mathbf{v}_c$ as:

$$\dot{\mathbf{s}} = \mathbf{J}_s \mathbf{v}_c \tag{2.2}$$

where $\mathbf{J}_s \in \mathbb{R}^{k \times 6}$ is reffered as *interaction matrix*[7] or *image Jacobian*[7], [12]. From Equations 2.1 and 2.2 we obtain a relationship between the camera velocity $\mathbf{v}_c$ and time change of the error $\mathbf{e}$:

$$\dot{\mathbf{e}} = \mathbf{J}_e \mathbf{v}_c \tag{2.3}$$

where $\mathbf{J}_e = \mathbf{J}_s$. Using a proportional control (i.e. $\dot{\mathbf{e}} = -\lambda \mathbf{e}$), we obtain (using Equation 2.3):

$$\mathbf{v}_c = -\lambda \mathbf{J}_e^+ \mathbf{e} \tag{2.4}$$

Note that $\mathbf{J}_e^+ \in \mathbb{R}^{6 \times k}$ is a pseudo-inverse of $\mathbf{J}_e$ such as $\mathbf{J}_e^+ = (\mathbf{J}_e^\intercal \mathbf{J}_e)^{-1} \mathbf{J}_e^\intercal$. This is the basic idea implemented in most VS controllers[7]. In the next two sections (2.1.1 and 2.1.2) we show derivation of $\mathbf{J}_e$ for IBVS and PBVS respectively. In real systems, however the image Jacobian is never known exactly, thus only an approximation can be used.

2.1.1 Image based visual servoing

In IBVS $\mathbf{s}$ contains a set of 2D features (e.g. points, lines or circles in image plane). Image measurements $\mathbf{m}(t)$ are typically (in case of point features) the respective point coordinates in image plane, and $\mathbf{a} = (u_0, v_0, f, \alpha, \rho_u, \rho_v)$ represents camera intrinsic parameters (u_0, v_0 are pixel coordinates of the principal point, f is focal length of camera, ρ_u, ρ_v is the pixel width and height and α is pixel aspect ratio).

2.1.1.1 Perspective camera model

Assume we have a calibrated perspective camera and $\mathbf{s}$ is a single point. Relation between 3D world coordinates of that point and 2D image plane coordinates is:

$$\begin{aligned} x &= X/Z = (u - u_0)/f\alpha \\ y &= Y/Z = (v - v_0)/f \end{aligned} \tag{2.5}$$

The interaction matrix then become[7]:

$$\mathbf{J}_e = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.6}$$

Note that $\mathbf{J}$ in Equation 2.6 is generally time variant and explicitly depends on Z. It is thus necessary to estimate depth of the scene, as will be addressed later. If $\mathbf{s}$ consists of n points, we can simply stack image Jacobians for each point:

$$\mathbf{v}_c = -\lambda \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_n \end{bmatrix}^+ \mathbf{e} \tag{2.7}$$

2.1.1.2 Spherical camera model

Detailed derivation of spherical camera equations as well as transformation from fish-eye camera to a unit sphere can be found in [12]. Image Jacobian for a calibrated spherical camera and $\mathbf{s}$ containing a single point is described in terms of colatitude angle θ, azimuth angle ϕ and distance from the camera origin to the point in world coordinates R. Then we can rewrite Equation 2.2 as [12]:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \mathbf{J}_e(\theta, \phi, R)\mathbf{v}_c \tag{2.8}$$

where:

$$\mathbf{J}_e = \begin{bmatrix} -\frac{\cos\phi\cos\theta}{R} & -\frac{\sin\phi\cos\theta}{R} & \frac{\sin\theta}{R} & \sin\phi & -\cos\phi & 0 \\ \frac{\sin\phi}{R\sin\theta} & -\frac{\cos\phi}{R\sin\theta} & 0 & \frac{\cos\phi\cos\theta}{\sin\phi} & \frac{\sin\phi\cos\theta}{\sin\theta} & -1 \end{bmatrix} \tag{2.9}$$

Note that instead of Z we need to estimate R. However, similar techniques as for perspective camera can be used, as only first three columns of $\mathbf{J}_e$ depends on R. A proportional control law from Equation 2.4 would be then written as ($\ominus$ denotes modulo subtraction and returns the smallest angular distance given that $\theta \in \langle 0, \pi \rangle$ and $\phi \in \langle -\pi, \pi)$ [12]):

$$\mathbf{v}_c = -\lambda \mathbf{J}_e^+ (\mathbf{s}(\phi, \theta) \ominus \mathbf{s}^*(\phi, \theta)) \tag{2.10}$$

2.1.1.3 Depth estimation

Depth of the scene Z can be estimated in various ways. The simplest solution is to assume a constant depth, which is a valid assumption as long as the camera motion is in the plane parallel to the planar scene[12]. Another option is to use sparse stereo techniques to estimate the scene depth from consecutive images. This approach is valid as long as there is sufficient displacement between the two images. Third option is to estimate Z using measurements of robot and image motion - a *depth estimator*[12]. Rewrite Equation 2.6 as:

$$\mathbf{J}_e = \left[\begin{array}{ccc|ccc} -\frac{f}{\rho_u Z} & 0 & \frac{\bar{u}}{Z} & \frac{\rho_u \bar{u}\bar{v}}{f} & -\frac{f^2+\rho_u \bar{u}^2}{\rho_u f} & \bar{v} \\ 0 & -\frac{f}{\rho_v Z} & \frac{\bar{v}}{Z} & \frac{f^2+\rho_v \bar{v}^2}{\rho_v f} & -\frac{\rho_v \bar{u}\bar{v}}{f} & \bar{u} \end{array} \right] \tag{2.11}$$

where $\bar{u} = u - u_0$ and $\bar{v} = v - v_0$. Substitute Equation 2.11 to 2.2 and rearrange:

$$\begin{aligned} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} &= \mathbf{J}_e \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \\ \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} &= \left[\tfrac{1}{Z}\mathbf{J}_e | \mathbf{J}_\omega \right] \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \\ \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} &= \tfrac{1}{Z}\mathbf{J}_e \mathbf{v} + \mathbf{J}_\omega \omega \end{aligned}$$

$$\frac{1}{Z}\mathbf{J}_e \mathbf{v} = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} - J_\omega \omega \tag{2.12}$$

On the right hand side vector $[\dot{u} \quad \dot{v}]^\mathsf{T}$ represents the observed optical flow, from which the optical flow caused by camera rotation is subtracted. This process is called *derotating* of optical flow. The left hand side is optical flow caused by pure translation. Equation 2.12 can be written as:

$$\mathbf{A}x = \mathbf{b} \tag{2.13}$$

where $x = 1/Z$. This set of linear equations can be then solved using least-squared method. Note that depth estimation for a spherical camera would be derived in a similar manner.

As can be seen from the previous derivations, optical flow described in Section 2.4 is a natural part of IBVS, as it describes motion of the image features (in the simplest case points) between two consecutive images $(\dot{u}, \dot{v})$.

2.1.2 Position based visual servoing

In PBVS, vector $\mathbf{s}$ contains a set of 3D features, which are defined with respect to the current camera pose and world coordinate frame (e.g. world coordinates of the target). Then $\mathbf{a}$ contains camera intrinsic parameters (described in Section 2.1.1) and a 3D model of the object (3D model can degenerate to a single point)[7]. Camera pose can be estimated by Visual odometry (Section 2.2, Visual SLAM (Section 2.3) or other method[8]. In this case, image Jacobian is independent on the camera model, as we are working with 3D features (camera model is necessary for camera pose estimation), but depends on $\mathbf{s}$ and $\mathbf{s}^*$

Let's define $\mathbf{s} = (\mathbf{t}^*, \theta\mathbf{u})$, $\mathbf{s}^* = 0$, $\mathbf{e} = \mathbf{s}$, where $\mathbf{t}^* \in \mathbb{R}^{3\times1}$ gives coordinates of the origin of the object frame relative to the desired camera frame and $\theta\mathbf{u}$ gives the angle/axis parametrization for the rotation. Image Jacobian is then[7]:

$$\mathbf{J}_e = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{J}_{\theta u} \end{bmatrix} \tag{2.14}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix describing orientation of the camera relative to the desired position. $\mathbf{J}_{\theta u}$ is defined in [7]. In this case rotational and translational motion is decoupled, which simplifies the control law:

$$\begin{aligned} \mathbf{v}_c &= -\lambda\mathbf{R}^{\mathsf{T}}\mathbf{t}^* \\ \omega_c &= -\lambda\theta\mathbf{u} \end{aligned} \tag{2.15}$$

Because the control law is expressed in Cartesian coordinates, PBVS produces smooth and linear movement of the camera in world coordinate system, however as there is no direct control of the features movement in the image plane, they move in a non-linear manner and can even fall outside the visible region (as shown in Figure 2.1). On the other hand, IBVS produces linear movement of the features in the image plane, but there is no direct control of the camera motion, which can undergo undesirable paths.

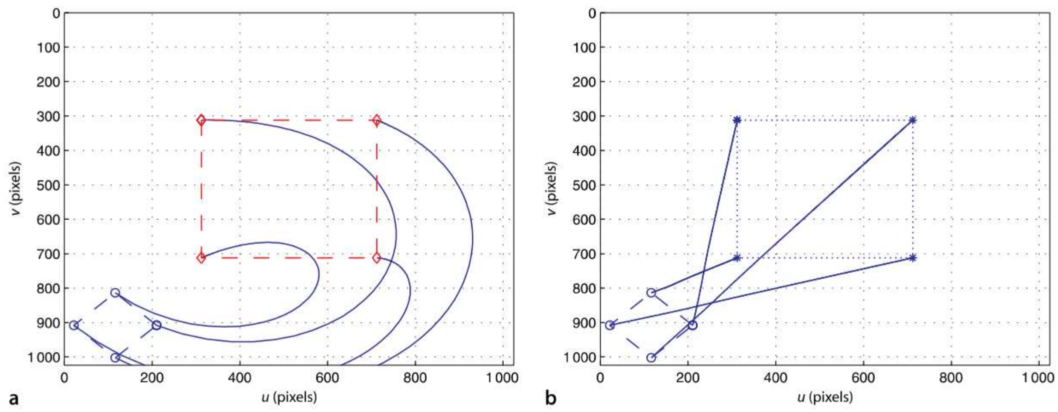


Figure 2.1: Image plane feature paths for **a** PBVS and **b** IBVS[12]

2.2 Visual odometry

Visual odometry (VO) is the process of estimating the egomotion of an agent (e.g. a flying platform) using only the input of a single or multiple cameras attached to it[44]. In a similar way as a classical odometry, VO incrementally estimates a position of the platform from images taken by an onboard camera. VO is especially useful in environments where it is not possible to use GPS for absolute positioning[51]. For our purpose we assume only monocular VO.[1]

As any other visual method, VO is usable only when certain assumptions about the scene are fulfilled. Namely a sufficient illumination of the scene is required, as well as a static scene with sufficient texture. Moreover the consecutive images have to have a sufficient overlap. Note that VO is a specific case of a recovery of a relative camera pose and 3-D structure from a set of camera pictures, which is called *structure from motion* problem[44].

VO problem is defined as follows. Two camera positions at time instances $k-1$ and k are related by the rigid body transformation:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.16}$$

where $T_{k,k-1} \in \mathbb{R}^{4\times4}$, $R_{k,k-1} \in SO(3)$ is the rotation matrix, and $t_{k,k-1} \in \mathbb{R}^{3\times1}$ is the

[1]The reason is that stereo vision degenerates into monocular case when the distance to target is much larger than the distance between the two cameras (baseline). Stereo vision thus would bring no benefit to flying platforms (just additional payload mass). Note also that stereo VO was already used in many applications, for example in Mars rovers[44]

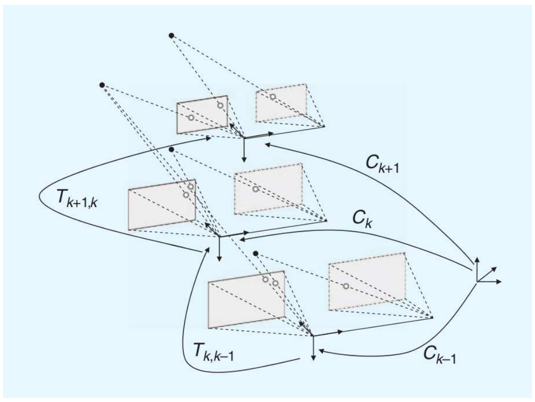


Figure 2.2: An illustration of the visual odometry problem. The relative poses $T_{k,k-1}$ of adjacent camera positions (or positions of a camera system) are computed from visual features and concatenated to get the absolute poses C_k with respect to the initial coordinate frame at $k = 0$[44].

translation vector. C_k is a camera pose at time instance k with respect to the origin, $C_k = C_{k-1}T_{k,k-1}$. The task of VO is to compute the relative transformation from images I_k and I_{k-1}, and concatenate the transformations to recover camera poses from C_0 to C_k[44]. Whole situation is shown in Figure 2.2 (note that it shows a stereo VO problem).

A pipeline of a VO algorithm is presented in Figure 2.3. First, two consecutive images are captured. Then, in both images are detected significant features using appropriate detectors (e.g. corner detectors Harris[21], Shi-Tomasi[47], or blob detectors SIFT[33], SURF[1]). Features found in image I_{k-1} are then tracked or matched with features in image I_k. Most commonly is used *random sample consensus* RANSAC[18] and its modifications. For details see [44].

Once the features are matched (tracked) between the images, motion can be estimated. There are three approaches, in short in 2D-to-2D the motion is estimated using *essential matrix* and a minimal set of features in the image plane. In 3D-to-2D approach, 3D features have to be first computed in the first image (for example by triangulation), and then projected into the second image plane (2D). In the last case (3D-to-3D), 3D features are computed in both images. In all three situations is our aim to minimize the re-projection error caused by feature misalignment. Finally, it is possible to further optimize the camera pose estimation, for example by *window bundle adjustment*[44].

In comparison with visual SLAM (VSLAM, presented in Section 2.3), VO is aimed at local consistency of the trajectory (over last n frames), whereas VSLAM is focused on

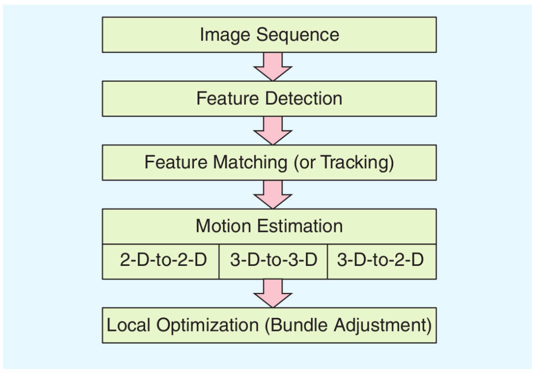


Figure 2.3: A block diagram showing the main components of a VO system[44].

global consistency of the map. VO is significantly easier to implement and is less memory and computationally demanding than VSLAM, and thus preferable in some applications. In other words, VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera[44].

A natural approach for pose estimation of a flying platform would be a fuse of visual odometry and measurements from on-board Inertial Measurement Unit (IMU). This fusion uses Kalman filtering and both theoretical analysis ([49], [35], [39] and especially [27] for UAVs) and practical implementations ([26], [28], [43]) were done. UAV attitude estimation using IMU and visual data is addressed in Section 3.4.

2.3 Visual SLAM

Simultaneous localisation and mapping problem is aimed not only at providing a pose estimation of the robot (as Visual Odometry does), but also at building a consistent map of an a-priori unknown environment surrounding the robot. Thus we know not only the position of the robot, but also its location within the on-line built map. A solution to the SLAM problem has been seen as a ”holy grail” for the mobile robotics community as it would provide the means to make a robot truly autonomous[15]. Visual simultaneous localization and mapping (V-SLAM) emphasise that information about the surrounding environment is gathered through a camera. State of the art is implementation of V-SLAM algorithm for autonomous navigation, take-off and landing of a quadrotor[51].

SLAM is a process by which a mobile robot can build a map of an environment and

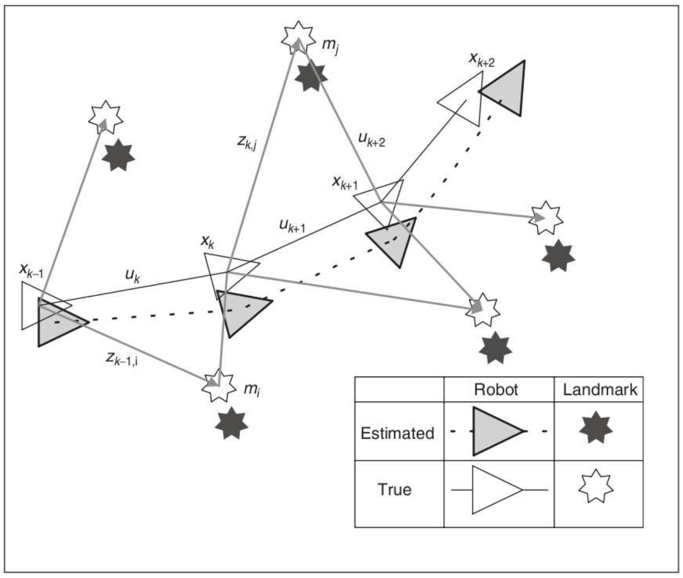


Figure 2.4: The essential SLAM problem. A simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations[15].

at the same time use this map to deduce its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location[15]. A visualisation of a SLAM problem is shown in Figure 2.4. It can be viewed as an extension of a visual odometry problem, where we keep history of the estimated poses and tracked features (landmarks) and refine our guess as we are getting more observations.

To achieve this goal, extended Kalman filter or Rao-Blackwellized particle filter is used[15]. This implies that we have to know a model of a vehicle kinematics and a probabilistic estimation of the measurement noise. Another important concept in (V)SLAM is a *loop closure* - i.e. once the robot returns to its initial position, it is possible to globally optimize its previous path.

In practical implementations of V-SLAM, a 3D map of the environment is built based on *keyframes*, which are images rich on features selected from the image sequence. An example of such a map is shown in Figure 2.5. The need to store the map and the observed features practically limits its use for indoor UAV only, as the problem reaches computational capacity of current hardware.

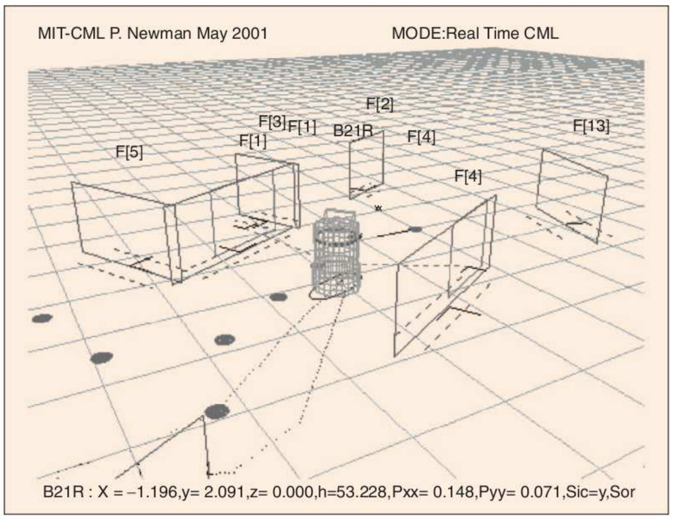

MIT-CML P. Newman May 2001
MODE:Real Time CML
F[2]
F[3]F[1]
B21R
F[4]
F[13]
F[5]
F[1]
F[4]
B21R : X = –1.196,y= 2.091,z= 0.000,h=53.228,Pxx= 0.148,Pyy= 0.071,Sic=y,Sor

Figure 2.5: A real-time SLAM visualisation[40].

2.4 Optical Flow

Optical flow (OF) is a vector field mapping one image to another in a consecutive image sequence. Optical flow represents movement in the image, and can be caused either by moving objects in the scene, or by camera motion. Furthermore, optical flow can be also caused by a moving source of light, producing a virtual motion[25]. Generally it is impossible to determine whether the movement was caused by a moving object or by a moving camera without any a-priori knowledge (assuming a static light source). However, placing certain constraints on the scene (e.g. a planar scene without moving objects, or a stationary camera) allows us to use optical flow in a variety of applications, such as target tracking[10], point of contact estimation[11], VTOL terrain following[22] or VTOL hovering and landing[23].

OF is calculated by tracking movement of image pixels between two images. If all pixels in the image are tracked a ”dense” OF is produced. If only a subset of pixels (features) in the image is tracked, the produced OF is ”sparse”. All OF methods assume uniform illumination of the scene, brightness constancy between the images, temporal consistency on ”small” movements (e.g. the tracked feature is constant during a small motion) and spatially smooth image motion.

The basic method of calculation OF is Horn-Schunck method[25]. It is a ”dense” method, as in its native implementation it calculates optical flow for *each* pixel in the image. Another popular method is Lukas-Kanade Tracker[34], which tracks only a-priori

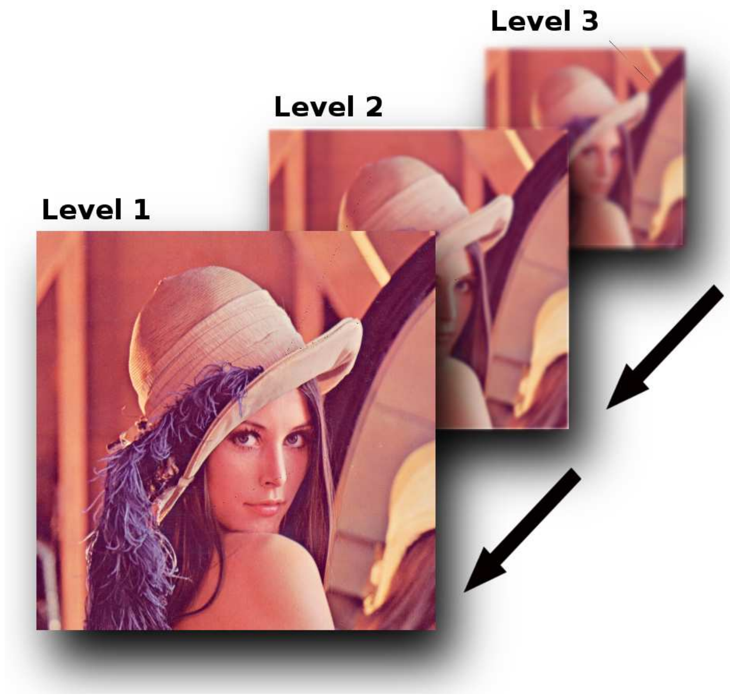


Figure 2.6: Image pyramid of level 3 using the famous Lenna Sjööblom picture (*http://www.lenna.org*)

selected features (thus a ”sparse” method). In practice these features are either selected by an appropriate feature detector (e.g Harris[21]), placed on a mesh grid to evenly cover the image or a combination of both methods is used. Note that in a limit case when number of features is close to number of pixels, we might obtain a dense OF.

Third method is Fractional Order Optical Flow[9], which is a dense method and utilises fractional order calculus. In short, as both Horn-Schunck and Lukas-Kanade methods work with first derivation of pixel intensity, with fractional order calculus it is possible to use r-th derivation($\frac{d^r u}{dx^r}, r \in \mathbb{R}$), where r serves as an additional parameter.

If a large motion in the scene is expected, it is beneficial to extend the OF algorithm using *image pyramids*. It means that the image is down-sampled according to the pyramid height, and OF is then calculated from the smallest to the largest image (”coarse to fine strategy”, see Figure 2.6), iteratively improving OF estimation. Details about pyramidal implementation of LK Tracker can be found in [3].

Chapter 3

Visual servoing in applications

In this chapter the concepts described in Chapter 2 are applied on real flight tasks, specifically:

Take-Off and Landing: theoretical analysis, prototype development and evaluation, real-time implementation

Ground Speed Estimation: theoretical analysis, prototype development

Obstacle Avoidance: theoretical analysis

Attitude Estimation: theoretical analysis

Detailed description of each task as well as possible visual servoing concepts are described in the rest of this chapter. Discussion about the flight tasks and description of further steps is provided in Section 3.5.

3.1 Vertical take-off and landing

The task of the biggest concern is autonomous vertical take-off and (especially) landing. Landing is probably the most difficult part of the flight, namely while wind gusts are present. Having a safe and robust landing procedure is desired as it increases the overall reliability of UAV. Even though take-off seems to be simple in comparison with the landing, it is still a non-trivial procedure while side winds are present.

Visual servoing for landing have been already successfully applied to fixed wings UAV ([45], [48]). Similar attempts were also made for high-end outdoor VTOL platforms ([46]

and [50]), and more recently for small indoor quadrotors too ([29] and [51]). Vision-based landing is theoretically well described, however a reliable implementation on a medium sized outdoor platform (such as an quadrotor described in Appendix A) is still challenging.

VTOL platform landing phases consist of selection (or finding) a landing spot, stabilization of the platform, descend and touchdown, and are described in the following sections.

3.1.1 Landing spot selection

There is a number of ways of selecting a suitable landing spot. In indoor applications, the spot is marked by a distinctive marker (a target) and this target is tracked during whole landing procedure (as in [29]).

In outdoor applications, the landing pad can be highlighted as in the indoor case, however it requires the UAV to look for the target during whole flight. A common approach is to select GPS coordinates of the landing pad. This is sufficient in general case, but due to precision of most commercially available GPS receivers (± 2 meters) it is not sufficient for a precise landing. Another possibility is to let the UAV to select a safe landing spot automatically, based on an unstructured terrain analysis procedure[6]. This method is favourable, as it would ensure safe landing in unknown environment (e.g. in case of emergency landing). Moreover it can be used in indoor applications too. Unfortunately autonomous safe landing spot identification is not yet reliable enough and further research in this area is needed.

A combination of previous methods is indeed possible. Imagine given GPS coordinates (waypoint) of the marked landing spot. Once UAV gets to the coordinates, it starts looking for the target and once it finds it, it tracks it. This approach saves computational power as target tracking is performed only during the landing phase. There is a number of search patterns (as shown in Figure 3.1) in which the UAV can move from the initial position. Once the target is found, UAV moves to the descent phase (Section 3.1.2).

3.1.2 Stabilization and descend

Once the landing spot is found (using any method mentioned in Section 3.1.1), it is desirable to stabilize UAV over the landing spot (i.e. $\phi, \theta = 0$ and $\dot{\phi}, \dot{\theta}, \dot{\psi} = 0$). The

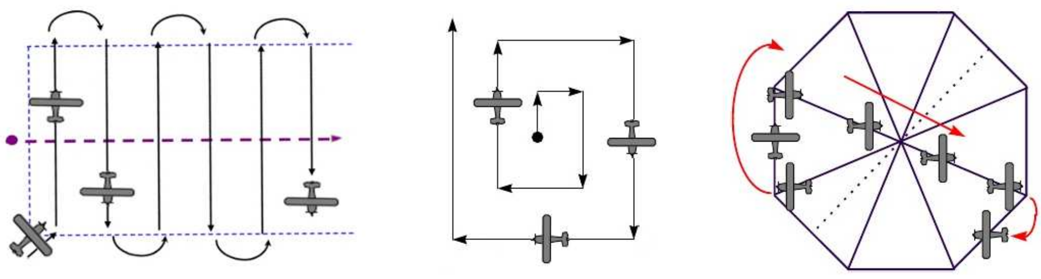


Figure 3.1: Left: Creeping line search, Middle: Box search, Right: Sector search (*http://www.scribd.com/doc/15048146/91/CREEPING-LINE-SEARCH*)

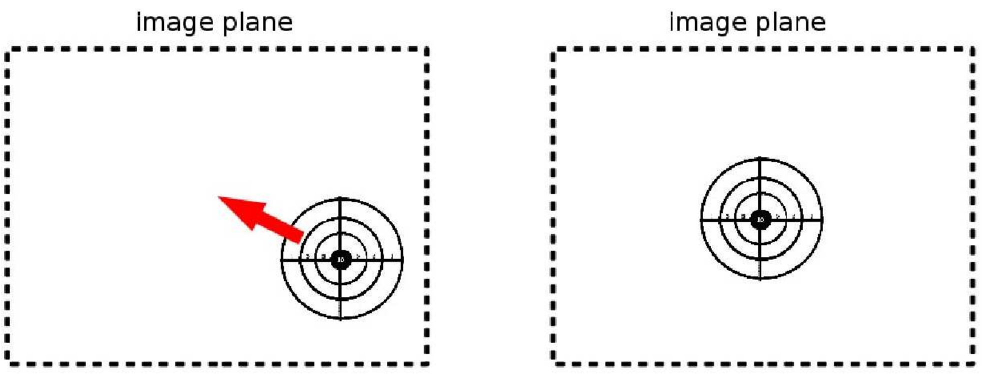


Figure 3.2: Centring the target in the image plane (target taken from *http://air.guns.20megsfree.com*

target should be in the middle of the image plane as is shown in Figure 3.2, so losing the target during descend is less likely. The descend and touchdown is identical to a helicopter landing.

3.1.3 Vision-based autopilot

As shown in Section 2.1, there are two basic concepts of Visual Servoing. Image-based VS, where we are concerned about the motion of tracked features within the image plane (i.e. optical flow), and Position-based VS, where we estimate camera position relative to the target (using e.g. VO or V-SLAM). Both concepts have been already applied (IBVS: [29], PBVS: [51]) and the choice depends on the platform and intended mission.

For indoor platforms is generally more suitable PBVS scheme (with VO or V-SLAM),

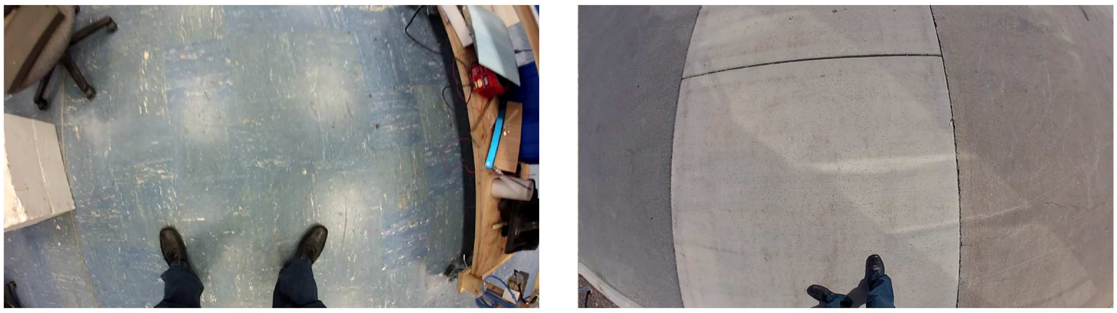


Figure 3.3: Indoor (left) and outdoor (right) ground image at altitude of 2 meters. (*Courtesy of CSOIS*)

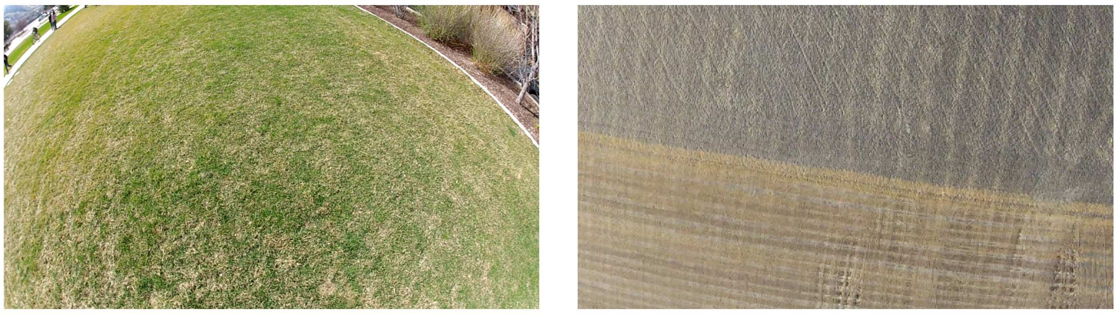


Figure 3.4: Ground image in HD resolution at altitudes of 2 meters (left) and 100 meters (right). (*Courtesy of CSOIS*)

because indoor environment typically offers better features to track and stable illumination, in comparison with outdoor environment (see Figure 3.3). Using V-SLAM gives us simultaneously a map of the surrounding environment, which is beneficial for other tasks (e.g. obstacle avoidance and ground speed estimation). Note that PBVS is more computationally demanding (especially when using V-SLAM) than IBVS.

Outdoor environments is more challenging in terms of disturbances (wind gusts, varying illumination and cloud cover), as well as in wider range of operating altitudes (see Figure 3.4). As shown in [51] the achieved worst-case bandwidth of the pose estimation system was 5Hz (using PBVS with V-SLAM). To mitigate aforementioned disturbances at least doubled bandwidth is desirable. The speed constrain then favours IBVS and Optical flow. Note that fusion of information from camera and IMU might be necessary to further improve autopilot's precision (see Section 3.4 for details). Implementation of a VTOL autopilot as well as experimental results can be found in Chapter4.

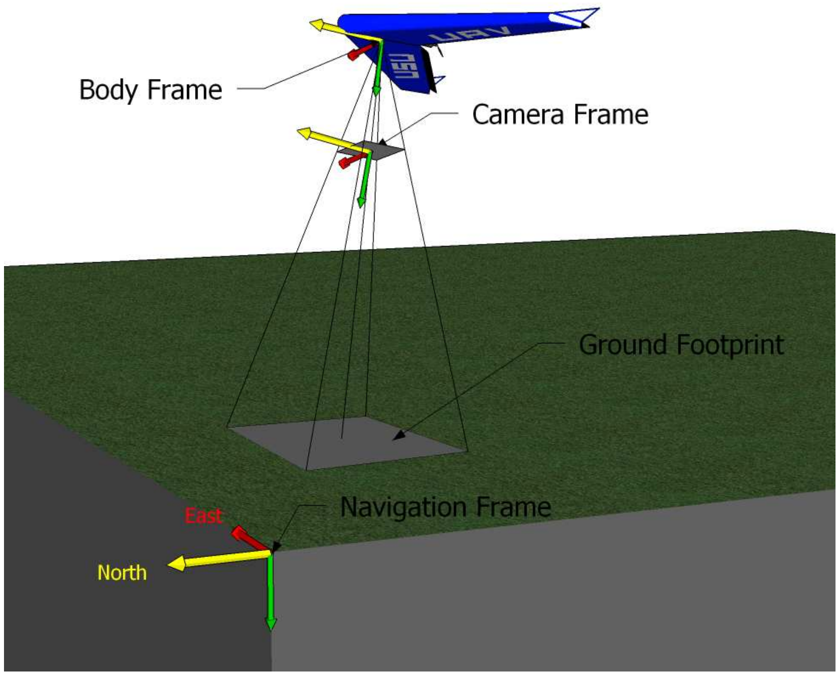


Figure 3.5: UAV and its footprint on the Earth.
(*Courtesy of Austin Jensen*)

3.2 Ground speed estimation

Ground speed estimation (GSE) becomes important during a forward flight (a constant speed is desired), as well as during hovering (the desired speed is zero). Constant ground speed is important for remote imaging purposes.

For a fixed wing platform a ground speed can be estimated from GPS and accelerometer data, using Kalman filtering. However, dynamics of a VTOL platform is less predictable and thus estimation based only on GPS and accelerometers is not reliable.

To correctly estimate the ground speed, we need to know a camera pose (or UAV attitude) and altitude. In case we estimate camera pose using VO or V-SLAM, ground speed is the horizontal translation between frames. In case we use IBVS and OF, attitude is taken from IMU, altitude from the altimeter and ground speed is calculated using simple geometric relations. Averaging magnitude and direction of OF gives a good approximation of the ground speed at the image principal point. Figure 3.5 shows relation between body frame, camera frame and world coordinate frame. Note that once we know the camera pose or calculate OF, GSE is not computationally expensive and thus can run simultaneously on a top of another visual servoing system.

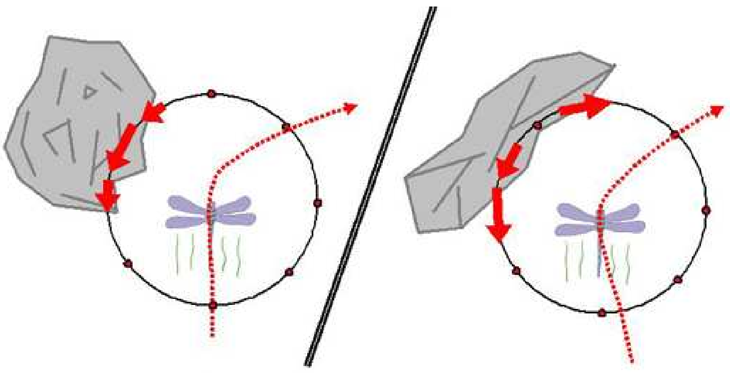


Figure 3.6: Saccade away from obstacle
(*http://centeye.com/technology/optical-flow/*)

3.3 Obstacle avoidance

Obstacle avoidance (OA) is of a growing importance for two main reasons. First, having a reliable OA method, it is possible to autonomously operate UAV at low altitudes in completely unknown environment, and in narrow areas (such as urban canyons). Second, as number of UAVs in air will most likely increase in near future, so will the possibility of a collision. A reliable OA method can safely recover UAV from a collision course.

First possibility is to use V-SLAM (Section 2.3) and a map of the environment to avoid stationary obstacles. However, first problem arises when we assume non-stationary obstacles (such as other UAVs) which are not tracked in the map. Second problem is speed. As mentioned in Section 3.1.3, worst-case bandwidth of the state-of-the-art V-SLAM systems is around 5 Hz. In case of an aggressive manoeuvres or a fast flight close to obstacles, higher bandwidth is desirable.

An inspiring solution is provided by nature itself. Pigeons use optical flow to precise velocity control during landing[30]. Insects (such as dragonflies) use optical flow to avoid obstacles during flight[20]. A rapid motion of an insect away from the obstacle is called *saccade* and is shown in Figure 3.6. Objects close to a camera (or an eye) naturally induce larger optical flow than objects in distance. Using this simple principle, insects *turn away* from areas with large optical flow. This approach was already implemented both on insect-like[1] and fixed wing[2] UAVs, using a small and fast optical flow sensor.[3].

[1] *http://robobees.seas.harvard.edu/*
[2] *http://www.youtube.com/watch?feature=player_embedded&v=qxrM8KQlv-0*
[3] *http://centeye.com/*

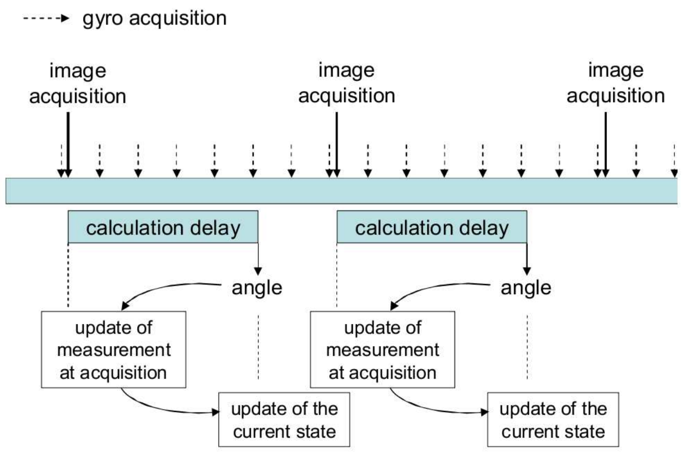

gyro acquisition
image acquisition
image acquisition
image acquisition
calculation delay
calculation delay
angle
angle
update of measurement at acquisition
update of measurement at acquisition
update of the current state
update of the current state

Figure 3.7: Time delay in the sensor fusion[24]

3.4 Attitude estimation

Precise and robust attitude estimation is of an utmost importance for low-cost UAV, which are typically equipped with low-end sensors. Correctly estimated attitude is extremely important for performance of all other tasks mentioned before.

Although a complementary filter is widely used for precise attitude estimation([17]), we are interested in fusion of visual and inertial data. One solution is V-SLAM (Section 2.3). Main benefit of V-SLAM is that it estimates not only the camera pose, but also a map of the surrounding environment. However, as mentioned in Section 2.3, its main disadvantage is computational complexity and thus is effectively limited to indoor use only. Second approach is to use VO (Section 2.2) combined with state estimation. This approach is further described.

Optimal state estimator (Kalman filter) for UAV and theoretical analysis of the method is already available ([39] and more recent [35]). In [43] an overview of published work on IMU and vision data fusion is provided. Obviously, a speed of camera is limiting the bandwidth of the system, as gyroscopes and accelerometers operate on significantly higher frequencies. Using a high-frame-rate camera as in [24] an outstanding precision can be obtained, however such a camera is out of question for the real UAV use. With standard cameras (up to 30 fps), time delay between the measurements have to be taken in account while designing the estimator (as shown in Figure 3.7).

In previously mentioned work only simple features such as points or corners were used. A significant improvement in precision of outdoor flights can be made if we track the horizon line (as proposed in [19]), or line segments in urban areas[26].

3.5 Summary

In this chapter the possible visual servoing applications were presented. Taking in account the available platform (in Appendix A) and the given time-frame, VTOL autopilot is the priority task to be implemented. With VTOL autopilot, the aforementioned platform is capable of fully autonomous missions, which is a desirable goal. Implementation of the Vision-based VTOL autopilot is described in Chapter 4. Obstacle avoidance is a task of high interest, however it requires a forward facing camera and thus changes in the airframe design. Attitude estimation on the current airframe is assumed to be precise enough, thus no specific effort is put into this task.

In summary, a detailed description of the VTOL autopilot is provided in Chapter 4, whereas experimental results and flight tests are described in Chapter 5.

Chapter 4

VTOL autopilot

In this chapter a complete design of a VTOL autopilot is presented. The chapter is divided as follows. In Section 4.1 the autopilot structure is described, as well as specifications and mission requirements. Section 4.2 is dedicated to the optical flow estimation during the landing manoeuvre. Selection of an optimal field of view of the camera is addressed in Section 4.2.1.

In Section 4.2.2 an algorithm for finding focus of expansion is proposed and evaluated on sample videos. Recommendations for real time FOE estimation are summarized in Section 4.2.3. Section 4.3 is dedicated to a landing pad detection. Design of the landing pad is addressed, as well as recommendations for real time implementation. Finally in Section 4.4 a discussion over the selected flight hardware is provided.

4.1 Autopilot design

When designing the VTOL autopilot, we first have to deal with the given requirements. The requirements are following:

- The airframe arrives at given GPS coordinates, then it has to autonomously find the landing spot and perform a safe descend.
- The initial altitude prior to the landing manoeuvre is 5 meters.
- The landing autopilot has to be able to land safely in presence of wind gusts up to 3 m/s.

- The take-off autopilot has to be able to perform a safe ascend from zero to 5 m altitude in presence of wind gusts up to 3 m/s.
- Maximal allowed pitch and roll angle is $10°$
- The output of the VTOL autopilot is the desired position of the airframe. It is then fed into the position controller.
- The desired VTOL autopilot bandwidth is at least 10 Hz

Prior to the design process, certain assumptions about the viewed scene have to be made to allow us to use image processing algorithms. The assumptions are following:

- Assume a planar landing surface without moving object in the scene.
- Assume an uniform illumination within the viewed scene.
- Assume brightness constancy, i.e. an image pixel does not change its brightness between two consecutive frames[4].
- Assume "small movement," i.e. the image motion of a surface patch changes slowly in time[4].
- Assume spatial coherence, i.e. neighbouring points in the scene belong to the same surface and have a similar motion[4].

Block diagram of the VTOL controller structure is shown in Figure A.7. Velocity and attitude control loops, as well as a position controller are already implemented. The airframe can fly in two autonomous modes - *Auto1* when attitude stabilization is active and the airframe follows commands from RC safety pilot (e.g. thrust increase, turn left, right). *Auto2* is fully autonomous mode when the airframe can follow predefined way-points and stabilize itself. Details about the airframe can be found in Appendix A.

4.1.1 Landing autopilot

Taking in account requirements from the preceding section, we can aim at landing autopilot design. A flowchart of the landing procedure is shown in Figure 4.1. Assume we are starting at given landing coordinates (x_1, y_1, h_1). Given h_1 is 5 m, expected h_2 is 2 m (a this distance from the ground the ultrasonic sensors should be accurate enough).

1. Start searching for the landing spot (the target). The target is described in Section 4.3. Use box search (Figure 3.1), as it is easiest to perform.
2. Once the target is found, stabilize over the target (i.e. hover in such a way that the target is in the middle of the image plane, as in Figure 3.2)
3. Start descending from height h_1 to height h_2 using optical flow and target tracking to keep the target right under the airframe.
4. Once is h_2 reached (ground is close enough), start fine touchdown control mainly driven by ultrasonic sensors.
5. After touchdown, switch-off the motors.

Because the position of the target is controlled, the autopilot is based on IBVS. However, we need to know the current attitude and position to be able to determine desired position for the position controller. A simple proportional controller should be sufficient for this task.

4.1.2 Take-off autopilot

Take-off autopilot is relatively simple in comparison with its landing counterpart. The identical requirements apply, thus output of the autopilot is also desired position. The biggest problem during take-off is a side wind, which can carry the airframe far away from its initial position. The autopilot has to be able to reject this disturbance. The take-off will be performed from the landing pad, which can provide additional position reference. The desired take-off altitude is 5 m (h_1). The take-off procedure is following:

1. Start motors
2. Start ascending. Use optical flow and target tracking to keep the airframe right over the target. Use ultrasonic sensors for precise altitude reference.
3. When the desired altitude is reached, stabilize over the target.
4. Hand over the control to flight autopilot (*Auto1* or *Auto2*)

Position of the target in the image plane is controlled, so the take-off autopilot is based IBVS. During the initial part of the take-off when the airframe is close to the ground,

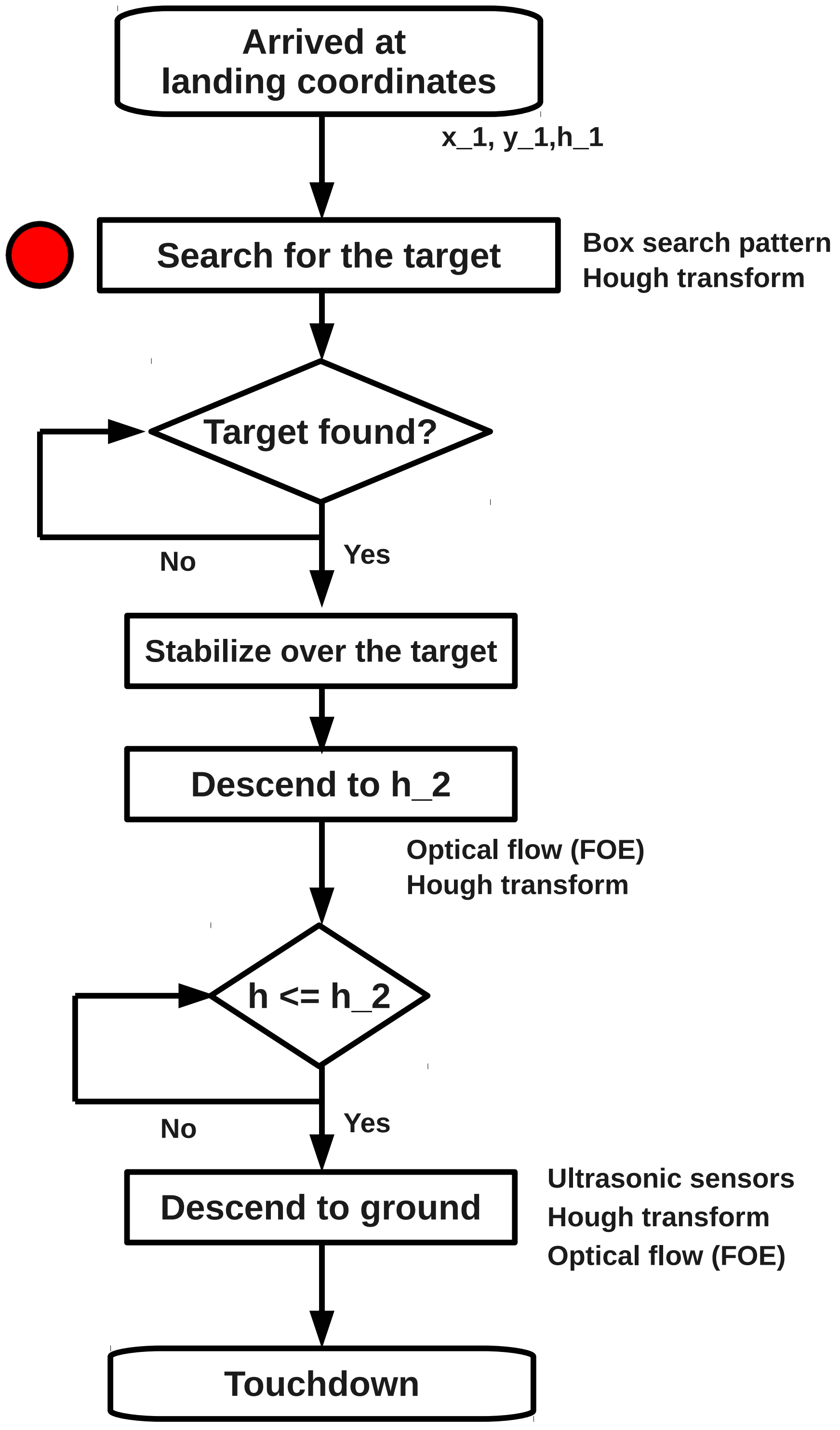


Figure 4.1: Landing autopilot flowchart

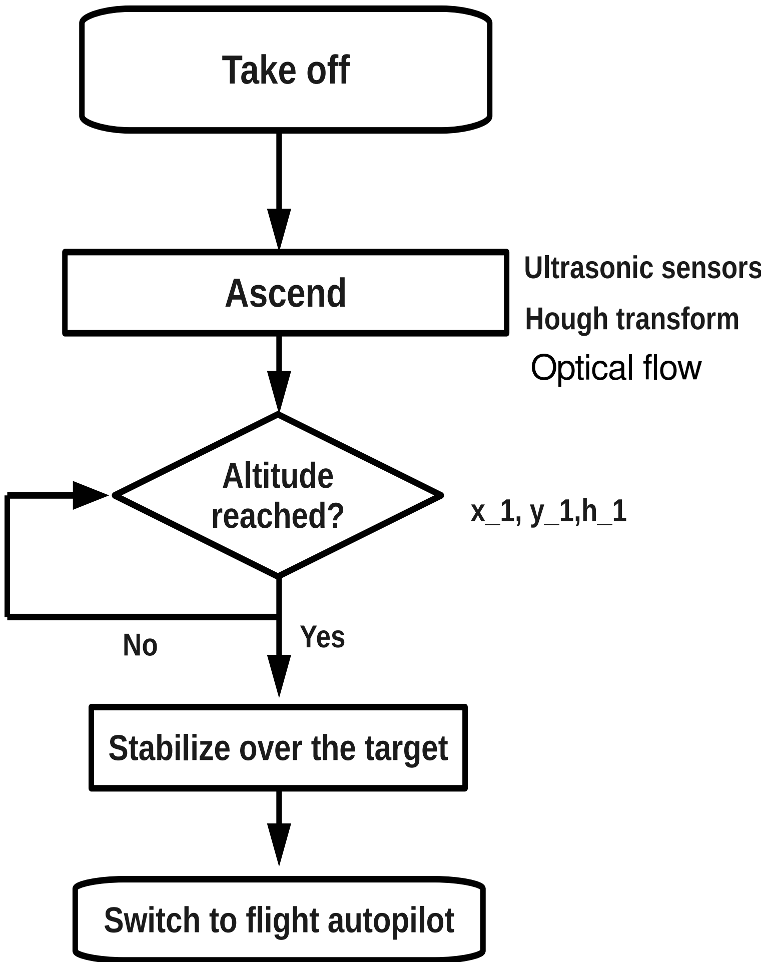


Figure 4.2: Take-off autopilot flowchart

proximity sensors have the highest importance for control. Note that unless altitude of at least 1 m is reached, the target occupies almost whole image plane and thus is difficult to track it reliably. However, at higher altitudes (close to 5 m), target tracking will be the most important method.

4.1.3 Safety considerations

Because we are dealing with the control of a flying vehicle, it is important to examine safety issues related with VTOL control. Mechanical and power failures are mitigated by careful pre-flight inspection. Failures in the lower control loops are also assumed to be mitigated. In case of the VTOL autopilot, we are concerned about two possible issues:

1. Initialization failure

2. Target tracking failure

To mitigate possible initialization failure, it has to be possible to visually check if the autopilot is ready before the flight (e.g. a flashing light). If the autopilot for some reason does not initialize prior to the landing sequence, the position controller has to regain the control authority and warn the safety pilot while keeping the airframe steady. In that case manual landing has to be performed.

Target tracking failure incorporates both *incorrect target detection* and *target loss*, Possibility of incorrect target detection can be reduced by placing the target far away from objects with similar appearance as the target, and from uneven terrain and obstacles. If the landing is performed within a view of the safety pilot, a visual control is possible.

If the target is lost during the landing/take-off manoeuvre, the airframe should stabilize itself at its current position and wait if the target is to be found again. If not, the landing procedure should be restarted - i.e. the airframe will go back to its initial position and start the box-search for the target.

Another possibility is that the desired position will be calculated incorrectly due to noisy data (e.g. incorrect target detection, FOE estimation or altitude estimation). This would usually happen for only a few frames, then the estimation should be corrected. To mitigate this problem, only a maximal allowed change in the desired position should be allowed, excluding too aggressive manoeuvres.

4.2 Optical Flow Estimation

As mentioned in Section 2.4 there are three main methods of calculating optical flow. Horn-Schunck method[25], Lukas-Kanade Tracker[34] and Fractional Order Optical Flow[9], which extends HS method using fractional order calculus. It was desirable to use a method which produces the most precise optical flow. For a comparison of the available methods a Matlab prototype was developed.

Matlab implementation of each method was downloaded from *Matlab Central*[1]. Optical flow was calculated from sample videos and image sequences taken either by hand or from [37]). Resulting OF from each method is shown in Figure 4.3. Note that although the images are in colours, optical flow is calculated from grayscale (intensity) images.

[1] *http://www.mathworks.com/matlabcentral/fileexchange/*

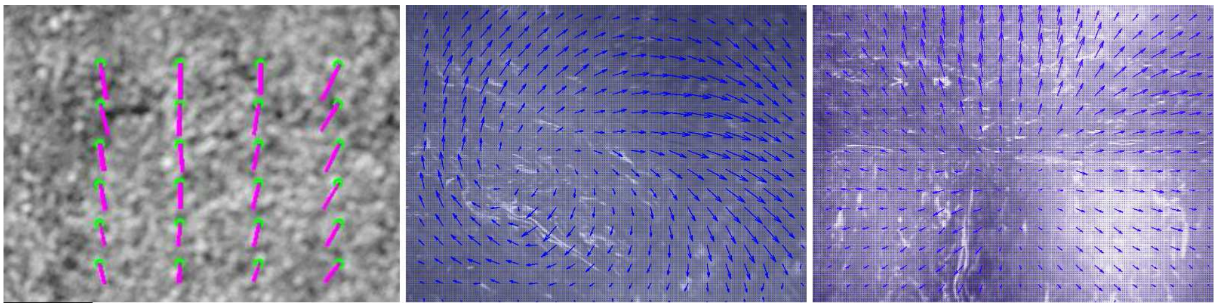


Figure 4.3: Left: Lukas-Kanade Tracker with a mesh of tracked points,
Middle: Horn-Schunck method,
Right: Fractional Order Optical Flow

The sample videos were not annotated,[2] and thus it was impossible to provide a qualitative analysis of the correctness of the OF. However, a comparison of all three methods is provided in [9], where on the annotated datasets FOOF method outperforms the other two. Nonetheless, none of the method was superior on the given video data. As a result, the final choice of the method depends on the performance of their C++ implementation, however FOOF is preferred. Regarding the frame rate, it was experimentally concluded that if the frame rate drops below 10 fps, a movement between two consecutive images is too large and optical flow is not estimated correctly.

4.2.1 Camera lenses

Both narrow field-of-view and fisheye camera lenses were used for a comparison:

- **Canon PowerShot A495:**[3] FOV $\sim 45°$, 320×240 px
- **GoPro Hero2:**[4] FOV 170°, HD resolution
- **Omni-tech Unibrain Fire-i BCL:**[5] a lens on a standard CCD chip, FOV 190°, 256×256 px, (datasets taken from [37])

A sample image from three different cameras is shown in Figure 4.4 (note the lens distortion). Advantages and disadvantages of both types of lenses are summarized below.

[2] i.e. no ground truth was known
[3] *http://www.canon-europe.com/For_Home/Product_Finder/Cameras/Digital_Camera/PowerShot/PowerShot_A49*
[4] *gopro.com*
[5] *http://www.omnitech.com/fisheye.htm*

Fisheye lens FOV $\geq 100^\circ$

- + Target is less likely to leave the image plane
- − Complicated geometry (image projected on a unit sphere)
- − Moving objects can appear on the edge of the image (as FOV is wider)
- − Sun or other strong light sources can appear in the image plane and deteriorate camera performance

Narrow FOV lens

- + Simple geometry
- − Target more likely to leave the image plane as FOV is narrower

Obviously fisheye lens has certain issues that has to be overcome (such as more complicated image projection). On the other hand, wider FOV ensures that the target (i.e. landing spot) stays in the image plane and can be tracked. A rule of thumb in robotics says that wider FOV of the vision sensor allows the robot to observe larger area with smaller motion. This is beneficial when UAV search for the target. As a result, a fisheye lens is preferred.

It is important to briefly address camera autofocus, because a vast majority of USB webcams nowadays has this ability. Assume a lens with a fixed focal length (*prime lens*). When focusing at a specific distance, the focal length inevitably slightly changes. This effect is called *focus breathing* and is proportionally dependent on focal length. Webcams usually have focal length of a few millimetres[19],[6] but the focus breathing is still visible. Fisheye lenses have smaller focal length,[7] as it is inversely proportional to FOV. Thus the wider FOV, the smaller focus breathing.

A preferred camera would have a fixed focus, because autofocus adds unnecessary complexity to the system. A fisheye lens with a small focal length has its focus range usually from few centimetres to infinity, which is sufficient for our system.

[6]For example Logitech HD Pro Webcam C910 f=4.3mm, Microsoft LifeCam Cinema f=4.5mm

[7]For example ORIFL190-3 fisheye lens from Omnitech has only f=1.24mm

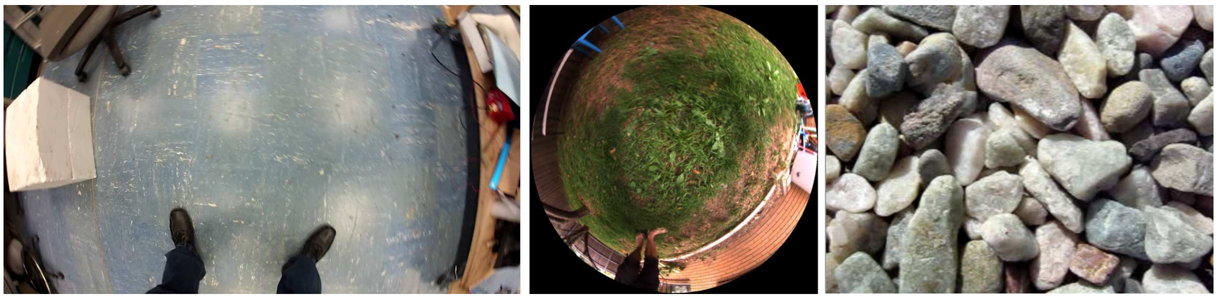


Figure 4.4: Left: GoPro (FOV 170°)
Middle: Unibrain fisheye lens (FOV 190°)
Right Canon PowerShot A495 (FOV $\sim$ 45°)

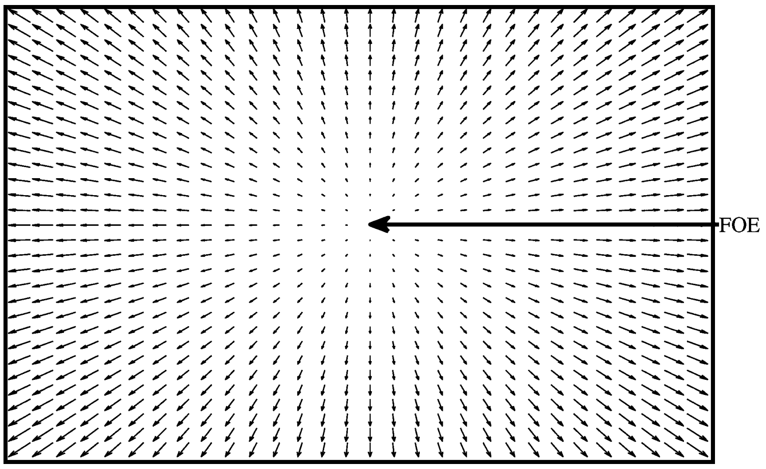


Figure 4.5: Diverging optical flow and the FOE[36]

4.2.2 Focus of Expansion

When approaching a planar surface, the time of contact τ is inversely proportional to the divergence of the optical field([36] and [37]). Divergence is defined as:

$$D(x,y) = \frac{\partial u(x,y)}{\partial x} + \frac{\partial v(x,y)}{\partial y} \tag{4.1}$$

where $u(x,y)$ is the horizontal component of OF, and $v(x,y)$ is the vertical component of OF. In an ideal situation OF is diverging from a single point in the image plane, as shown in Figure 4.5. This point is called Focus of Expansion (FOE). During landing on a planar surface, FOE represents the point of contact with the ground. When we know position of FOE in the image plane, we know at which point the UAV will touch the ground. It is indeed desirable to align FOE and the landing spot.

Several methods of FOE estimation exist ([36], [37] and [32]). After several experiments, a novel approach was suggested, based solely on the magnitude of the optical flow. Assume FOE lies within the image, and that the surface is perfectly planar. FOE is then the point with the smallest magnitude ($M(u,v) = \sqrt{u^2 + v^2}$) of optical flow, as is clearly visible in Figure 4.5.

The basic algorithm pipeline is following (assume that OF was already calculated):

```
1. Grab a new image and corresponding OF
2. Down-sample OF (use every n-th pixel only)
3. Calculate magnitude of the OF
4. Convert magnitude into binary image with a selected threshold
5. Invert (low values of magnitude are now 1 instead of 0)
6. Label separate components
7. Discard unreliable components
    • Optional: Apply morphological operations (e.g. opening/closing)
    • Discard components that are not solid enough
    • Discard components that are not circular enough
    • Select a component with the largest area
8. FOE is the centroid of the final candidate
```

This algorithm should be further improved to be robust enough, although for the sample sequences it provided good results. A Matlab prototype was developed to test the performance over various datasets. Note that using fixed thresholds required tuning for specific video sequences, thus it might be necessary to use adaptive thresholding. Another possible improvement is to include IMU measurements, so the algorithm will search for FOE only when the platform undergo horizontal acceleration (i.e. ascend or descent).

In Figures 4.6 and 4.7 is shown outdoor landing sequence taken with the Omni-tech fisheye lens. We can see a correct FOE estimation, especially during the last part of the landing (which is critical as UAV is close to the ground). In Figure 4.8 is shown an indoor landing sequence. The landing was under approximately 20° and FOE was

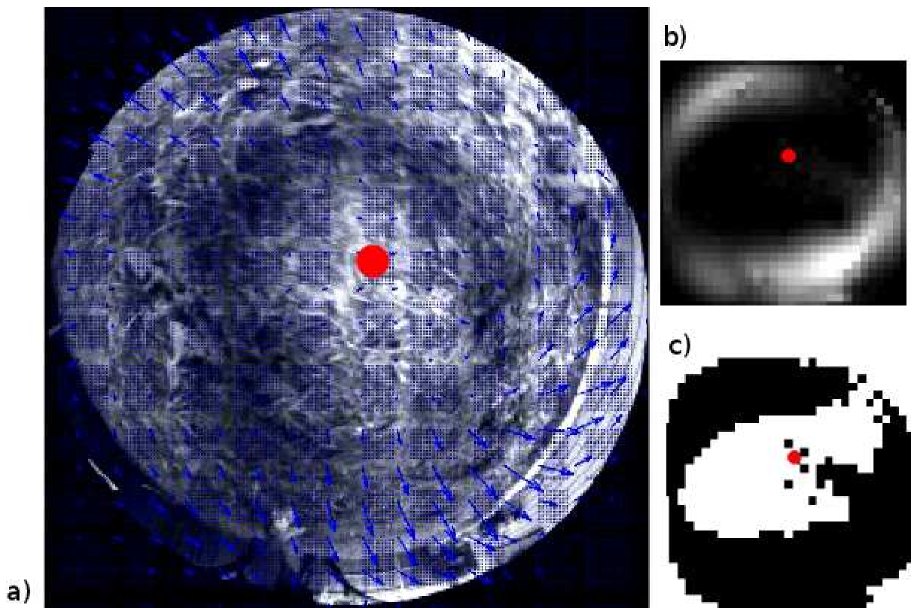


Figure 4.6: Outdoor grass landing, dataset from [37], FOE marked red. a) Original image with plotted OF, b) magnitude, c) labelled components

estimated correctly. Note that for a fisheye lens we have to take in account only the spherical part of the image plane and discard the bounding box.

Figure 4.9 shows outdoor landing sequence taken by hand with GoPro camera. Note the texture-poor environment. FOE estimation is biased because of moving objects in the scene (e.g. shadows). This has to be taken in account for the real-time implementation.

In Figures 4.10, 4.11 and 4.12 is shown a narrow FOV footage. Note correct FOE estimation even in presence of rotational movement (Figure 4.11). Figure 4.12 shows a take-off sequence. Note that FOE estimation as proposed in Algorithm 8 works also for take-off sequence, when the distance to the ground is increasing. This is an important fact for take-off autopilot implementation. Obviously when using Algorithm 8, better performance is achieved with perspective camera images (i.e. narrow FOV). For a fisheye camera, optical flow should be either mapped to local tangent planes of the image sphere (as proposed in [37]), or the algorithm for FOE estimation should be fed with angular rates and position angles from IMU.

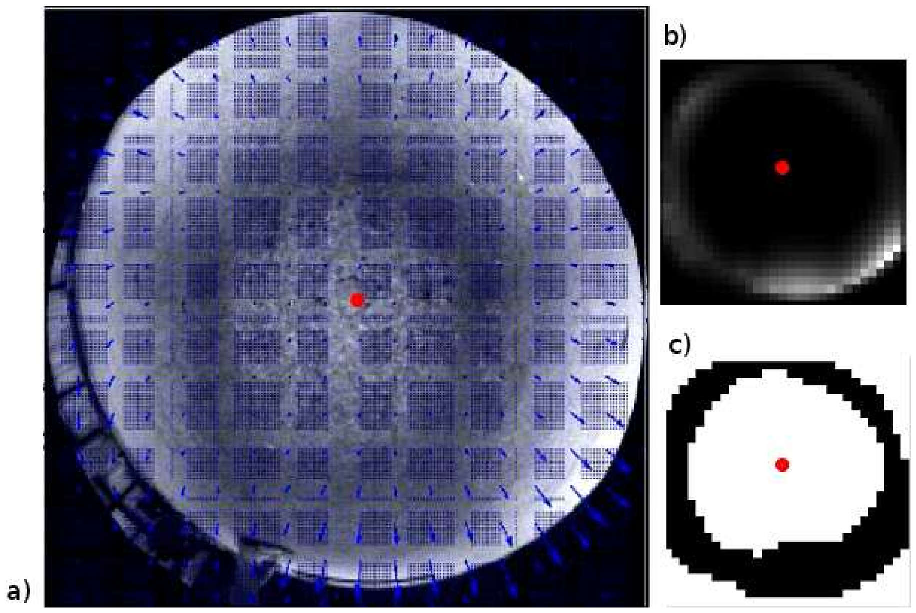


Figure 4.7: Outdoor cement landing, dataset from [37], FOE marked red.
a) Original image with plotted OF, b) magnitude, c) labelled components.
Video available at *http://www.youtube.com/watch?v=C7U4V0lqyVo*

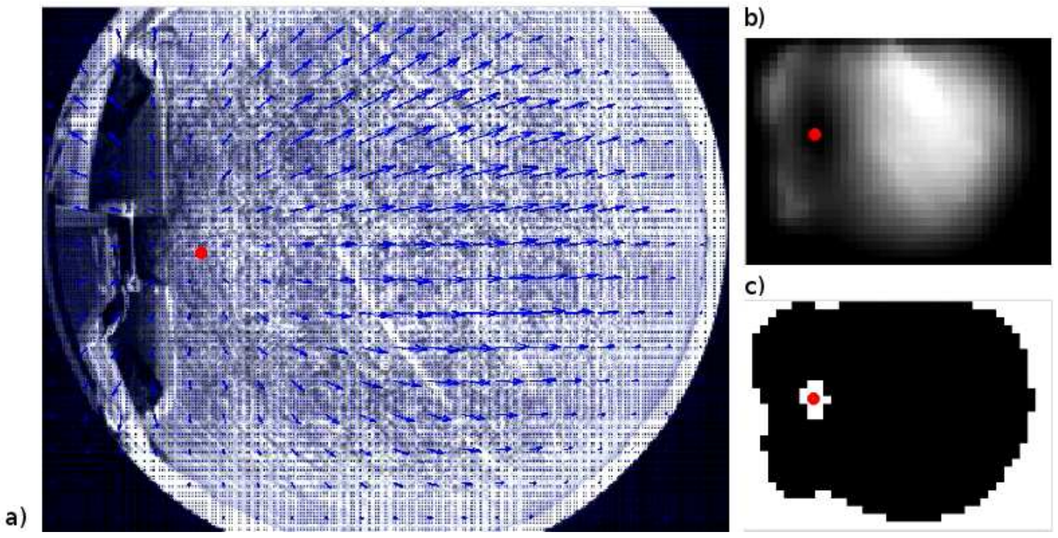


Figure 4.8: Indoor carpet landing, dataset from [37], FOE marked red.
a) Original image with plotted OF, b) magnitude, c) labelled components.
Video available at *http://www.youtube.com/watch?v=ZrpEa1oZLCY*

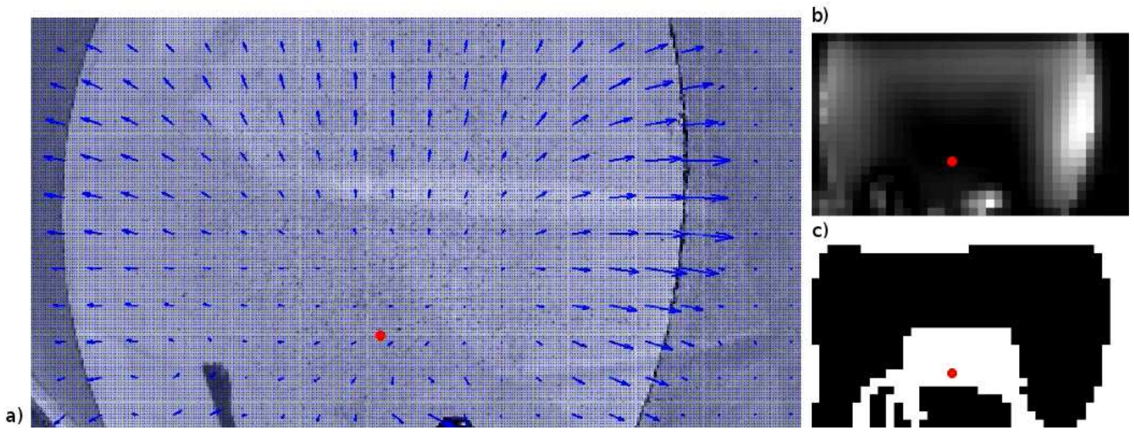


Figure 4.9: Outdoor cement landing, GoPro camera, FOE marked red.
a) Original image with plotted OF, b) magnitude, c) labelled components.
Video available at *http://www.youtube.com/watch?v=cK8D8-HkEGs*

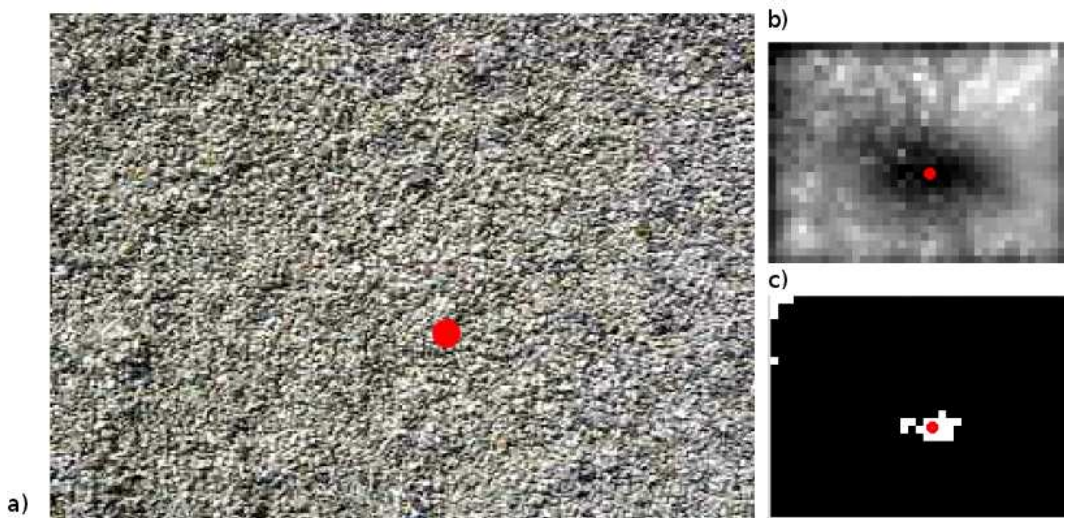


Figure 4.10: Outdoor gravel landing, Canon PowerShot, FOE marked red. a) Original image, b) magnitude, c) labelled components. Video available at *http://www.youtube.com/watch?v=1A-brG4Wq1Y*

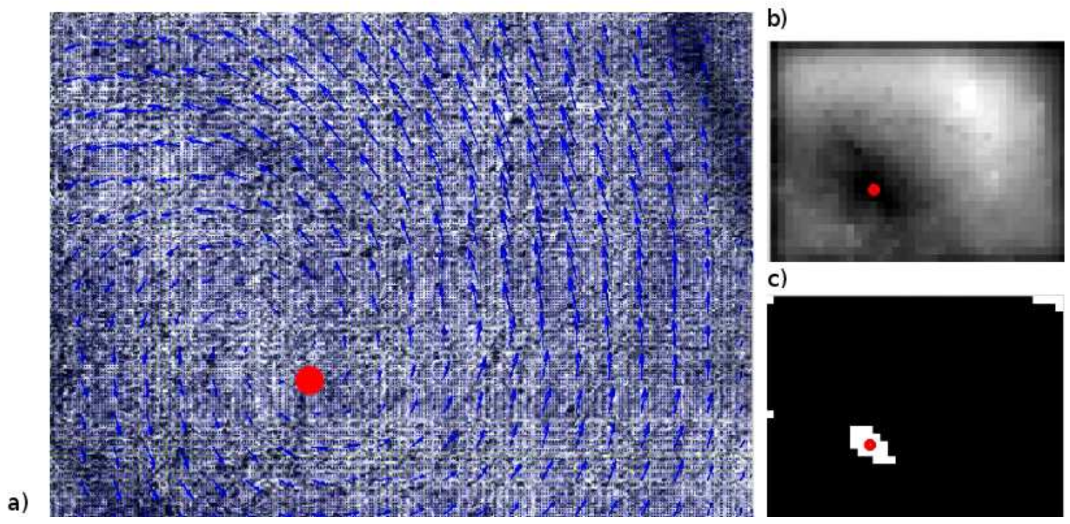


Figure 4.11: Outdoor grass landing, Canon PowerShot, FOE marked red. a) Original image with plotted OF, b) magnitude, c) labelled components.
Video available at *http://www.youtube.com/watch?v=9Yb9g9N8nR4*
Note the rotational motion.

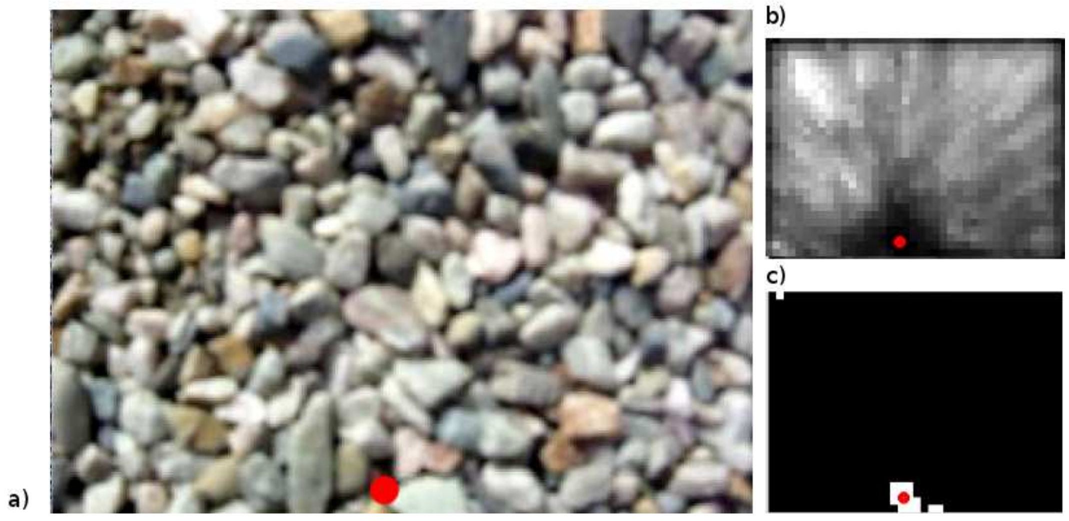


Figure 4.12: Outdoor gravel take-off, Canon PowerShot, FOE marked red.
a) Original image, b) magnitude, c) labelled components.
Video available at *http://www.youtube.com/watch?v=QfiwmshgLVQ*
Note that the performance is consistent despite the blurred image.

4.2.3 Recommendation

Summarizing the previous experiments, we can give this recommendation for the real time implementation of the autopilot:

Down-sample the original image - to improve performance.

Calculate OF selectively - chose only a mesh of points/features for which will be OF calculated to further improve performance.

Use pyramid implementation - of the OF method to be able to track even large motion.

Use Fisheye lens with fixed focus - with wide FOV it is easier to track the target, fixed focus decreases complexity of the system.

For the FOE estimation, the following suggestions should be taken in account:

Tune component selection - implement adaptive thresholding and possibly apply morphological operations to labelled components.

Fuse with IMU data - to provide a robust FOE estimator.

Computational cost of FOE estimation is negligible in comparison with OF calculation. The real bottleneck is the OF calculation and thus should be made as fast as possible. On a desktop computer with quad core 64-bit AMD processor and 8 GB RAM we were able to achieve frame rate roughly 10 fps. However, main delay was caused by highly unoptimized code and Matlab visualisation issues, so it is reasonable to expect a similar worst-case frame rate for a real-time implementation.

Note that FOE cannot be found when there is no movement in vertical direction. Also when the vertical velocity is small relatively to the altitude, FOE cannot be found either. These two constraints put a limit for real time application, and taking them in account, we can assume that optical flow and FOE estimation will be most reliable in a proximity of the ground (i.e. final part of the landing and initial part of the take-off manoeuvres).

4.3 Target detection

A reliable method for landing spot selection is a combination of GPS coordinates and a specially designed landing pad, as described in Section 3.1.1. Once reaching the given coordinates, UAV will start searching for a landing pad. When the landing pad (target) is found, it will be tracked until a successful touchdown. Similar procedure can be used during take-off, when the target provides a position reference.

The originally proposed target is shown in Figure 4.13. Assuming that perfectly circular objects are scarce in outdoor environment, such a target can be easily found using Hough circular transform. Knowing the target dimensions and a height from which UAV starts its search, we can specify minimal and maximal radius of the searched circle. Reliability of the detection can be increased if the target is painted in a distinctive color. We can then incorporate color filtering before Hough transformation.

Indoor sample images were processed with Matlab implementation of Hough Transform[8] to prove the feasibility of this method. Results are shown in Figure 4.14 for the black and white target. As a rule of thumb we can say that no reliable detection of the target is possible if it covers less than 1/10 of the image plane. To further improve the robustness of the tracking, a new bright red target was designed (as shown in Figure 4.13),

[8]Matlab Central: *http://www.mathworks.com/matlabcentral/fileexchange/*

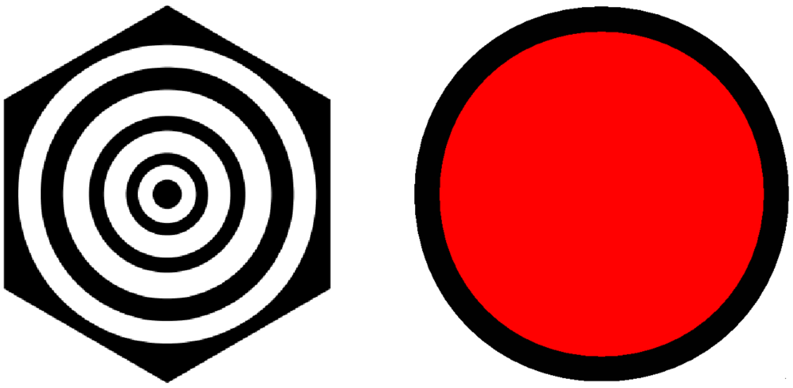


Figure 4.13: Left: The landing pad as proposed in [29]
Right: The final landing pad.

which allows color filtering. Red color was chosen because it is distinctive and relatively rare in natural scenes.

Using the same desktop computer as for OF estimation and images in HD resolution, a similar worst-case frame rate was achieved (at least 10 fps). Hough transform is known to be computationally intensive, but it is a simple and reliable method. There are a few options how to reduce the computational load, namely:

Down-sample images - using the same resolution as for OF calculation should be sufficient

Use the color mask - to reduce computation time and increase robustness

Note that in case of a fisheye lens, objects close to the edge of the image are distorted. A modification (such as elliptical Hough transform) of the detection method might be necessary. A combination of circular Hough transform in vicinity of the principal point and elliptical Hough transform near the edges of the image is an option.

Taking in account the previous recommendations, an optimal radius of the target was determine to be 0.75 m. Details about this calculation are provided in Appendix D.

4.4 Hardware

In this section separate hardware components of the autopilot are described.

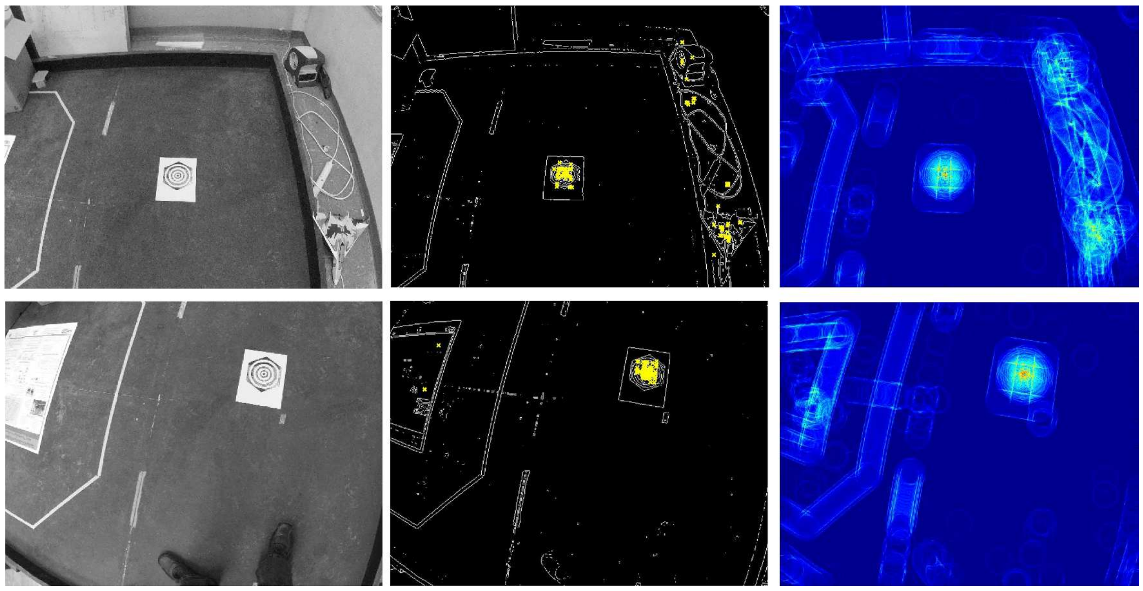


Figure 4.14: Hough transformation of sample images (GoPro camera)
Left: Grayscale images, Middle: Edges and detected circles,
Right: Hough accumulator space.

4.4.1 Camera and the lens

The main requirements for the camera were USB connection, frame rate at least 10fps with a resolution of 640 × 480 pixels, fixed focus and easy access to the lens and CCD sensor. A natural solution is a webcam, as it has an USB connection and all electronics packed on a small board. Unfortunately a majority of new webcams comes with autofocus lenses, and those are more difficult to remove.

The selected camera is a *Logitech Webcam C600*[9]. It has USB 2.0, 2 Megapixel 1/3.2" CCD sensor, frame rate up to 30 fps, video resolution up to 1600 × 1200 pixels at 10 fps, and fixed focus M10 × 0.5mm lens. Logitech don't ship this type any more, but it can be ordered either new or used from Amazon[10] or other resellers. A used camera can be bought for 20$. Another option for a similar price is a *PS Eye Camera*[11].

The selected lens is an M12 × 0.5mm fisheye lens *DSL125*[12] with horizontal FOV of 175° (at 1/3.2" sensor). *DSL125* can be ordered for 99$. A step-by-step procedure of customizing the webcam is provided in Appendix B, the camera is shown in Figure 4.15.

[9] *http://www.gizmag.com/logitech-c600-webcam-vid-software-review/12500/*
[10] *http://www.amazon.com/Logitech-Webcam-C600-Built-Microphone/dp/B002GP7ZT6*
[11] *http://www.amazon.com/PlayStation-Eye-3/dp/B000VTQ3LU*
[12] *http://www.sunex.com*

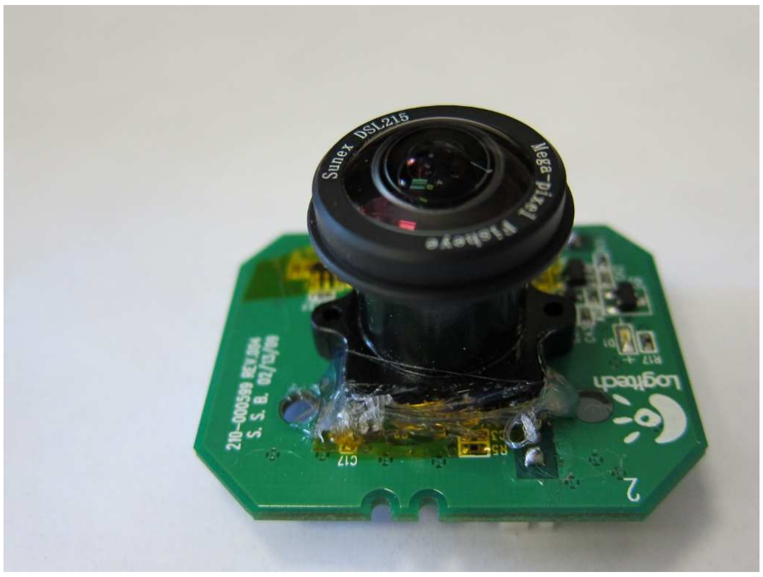


Figure 4.15: Logitech Webcam C600 with DSL125 lens without a cover box

4.4.2 On-board computer

As an on-board computer to perform all necessary computations is used *Pandaboard*[13]. It is a platform based on Texas Instruments *OMAP4* processor (dual-core ARM9, 1Ghz), with 1GB DDR2 RAM and SD/MMC card as a memory storage. Full specification can be found at the web page. For our purposes it is important that *Pandaboard* is powerful (yet affordable) enough to perform image processing on-board. To operate the board an operating system is required.

The camera is plugged into an USB port, the position controller is connected via RS-232 serial port. The detailed set-up of the *Pandaboard* is shown in Appendix C and the description of the VTOL airframe is provided in Appendix A.

4.4.3 Ultrasonic proximity sensor

For range measurements an ultrasonic sensor XL-MaxSonar MB1210[14] is used. It has the following features: ±1 cm resolution, measurement ranges from 20 to 700 cm, reading rate 10 Hz and either analog or serial (digital) output. One sensor can provide reasonable measurements of altitude, however having four of these sensors (one on each leg of the

[13] *http://www.pandaboard.org*
[14] *http://www.maxbotix.com*

airframe) gives enough information for precise position estimation (taking in account difference in measured height).

Pandaboard has only two serial ports available and one is needed for communication with position controller. For that reason a custom made board was used, which is equipped with a number of A/D converters and I2C interface. The details of the board and ultrasonic sensor set-up can be found in Appendix C.

4.5 Software

In this section the software structure of the VTOL autopilot is described. Autopilot is written in *C++* which is a natural choice for on-board devices programming, where performance is an issue. For image processing part is used *OpenCV* library[4] which is open source and provide a vast amount of ready-to-use functions. Additional libraries for *RS-232* and *I2C* communication are used, as well as *TinyXML*[15] for reading XML document.

An operating system for the on-board computer is required, thus *Ubuntu 12.04 Desktop edition* is used. Real time operation systems were also considered (and Linux kernels with pre-emptive planning are available), but as the bandwidth of the autopilot is only 10 Hz, using real-time OS is not worth the added complexity. The autopilot pipeline is shown in Figure 4.16, and it should be described more thoroughly.

- Upon initialization, settings are read from an XML file and various checks are performed to ensure that all components are working correctly.
- After initialization and a correct set-up, the main loop is entered.

  1. A frame is queried from the camera and transformed to an appropriate color-space (RGB)
  2. Target recognition algorithm finds the areas with desired color (e.g. red), on this color-based mask Hough circular transformation is performed.
  3. Optical flow between the current and the previous frame is calculated, and FOE is estimated.

[15] *http://www.grinninglizard.com/tinyxmldocs/index.html*

4. The controller compares the position of the target, the location of FOE and received range measurements with the airframe position and calculates the new desired position.
5. The controller also checks if the target was lost or incorrectly tracked (see the Safety considerations in Section 4.1.3).
6. The new desired position is send to the position controller.
7. Check if the desired altitude is reached. If no, go back to the beginning of the loop.

When the desired altitude is reached (which means either a successful take-off or touch-down), the autopilot switched to standby mode and hands over the control of the airframe. The code is included on the CD with all necessary libraries.

4.6 Summary

In this chapter a complete design of the VTOL autopilot was presented. Autopilot structure was described in Section 4.1, optical flow and FOE estimation was tested on sample videos and the results are presented in Section 4.2. Target detection algorithm was tested on sample images (see Section 4.3), and the results were used for designing a new target as shown in Appendix D. An overview of used hardware and software was provided in Sections 4.4 and 4.5.

Based on information from this chapter, an autopilot can be finally implemented and tested in real-time on the airframe. Results of the flight tests, autopilot calibration and performance benchmarking are to be found in the next chapter.

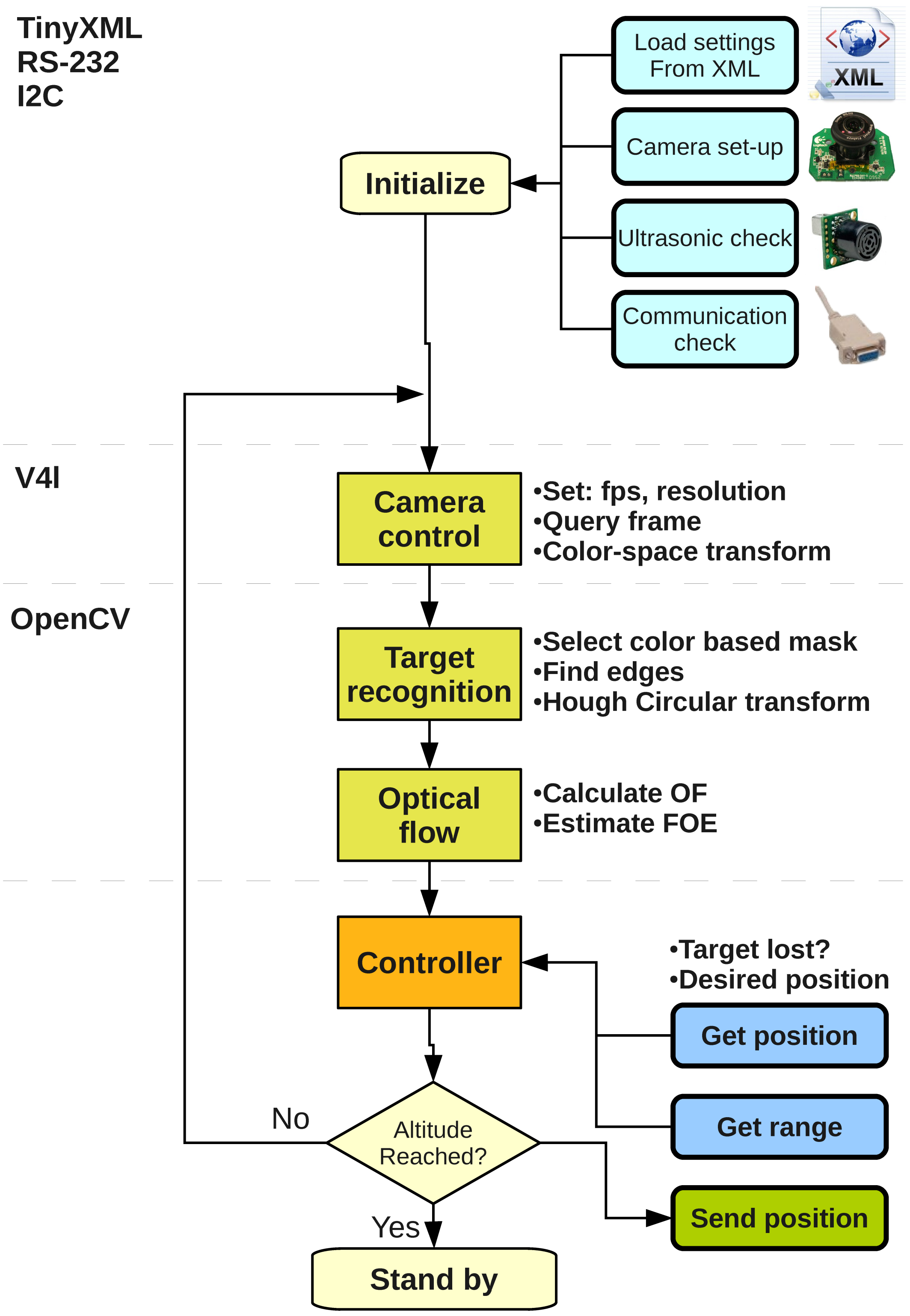


Figure 4.16: VTOL autopilot software pipeline

Chapter 5

Flight tests and Conclusion

In this chapter the real-time implementation of the VTOL autopilot from Chapter 4 is presented. It is important to mention that upon the thesis submission deadline, no flying platform was available to carry out the tests. The VTOL platform described in Appendix A was still in the phase of velocity controllers tuning. However, the work is carried on and the final results are to be presented during the thesis defence (on June 18 2012).

At this point, preliminary results are presented in the following section. Note that both the source code and the sample videos are available on the included CD. Sample videos were taken both outdoors and indoors, using the camera described in Appendix B.

5.1 Preliminary results

5.1.1 Calibration procedure

As proposed in Section 4.3, the target detection method was improved using color filtering. The commonly used RGB colorspace is not suitable for computer vision applications (mainly because it is difficult to express color "similarity" in the RGB cube), thus Hue-Saturation-Value[1] colorspace is used instead. Its main advantage is that we can express color similarity in Hue channel (e.g. only red tones) and then specify saturation and intensity of the color in the remaining two channels. As a result, HSV based color filtering is more robust against illumination changes.

[1] *http://en.wikipedia.org/wiki/HSL_and_HSV*

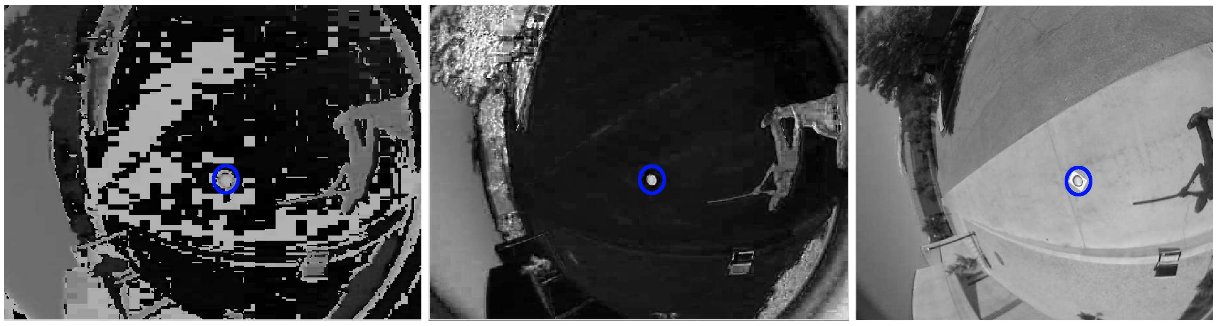


Figure 5.1: *Left:* Saturation, *Middle:* Hue, *Right:* Value
Video taken during a bright sunny day. The target is artificially marked blue.

The low-grade webcam used in this project perceives the same color differently under different light conditions (e.g. indoor vs. outdoor, or bright sun vs. cloudy day). To mitigate this issue and further improve the robustness of the target detection, a calibration procedure was implemented. This procedure allows the user to either calibrate the algorithm before the flight, or to define a set of algorithm settings for various conditions (e.g. different thresholds will be used indoors and outdoors). We can see the three HSV channels during calibration procedure in Figure 5.1. Note the clear difference between the target and the background in Hue channel (middle), whereas the scene illumination (Value channel) is almost uniform.

5.1.2 Improved target detection

The target detection was improved in two ways. First, the color based mask was used (as mentioned in Section 5.1.1). Second, a circular mask covering the Region of Interest (ROI) around the centre of the image plane was used to mitigate possible false detection on the edge of the field of view. Target detection algorithm in practice is shown in Figure 5.2. Note the color based mask precisely bounding the target.

5.1.3 Optical flow calculation

On the contrary, optical flow calculation and consequent focus of expansion estimation does not seem to be as precise as expected. In Figure 5.3 is shown a typical example. In close vicinity of the ground (i.e. altitude ≤ 1 m) the optical flow is calculated correctly (as

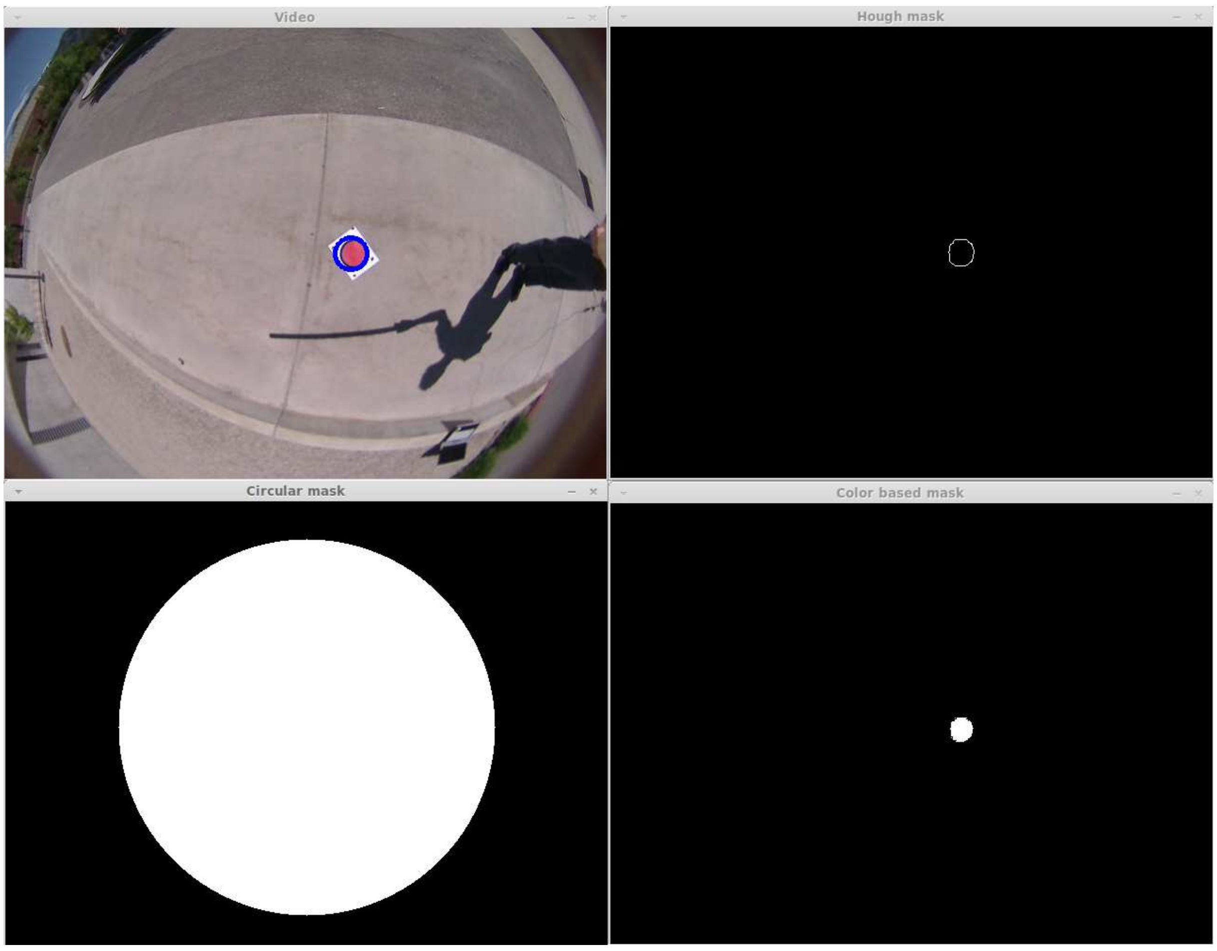


Figure 5.2: Target detection on an asphalt road. *Top-left:* The original image with the detected target, *Bottom-left:* Circular mask (ROI is white), *Top-right:* Edges passed to Hough circular transform, *Bottom-right:* Color based mask of the original image

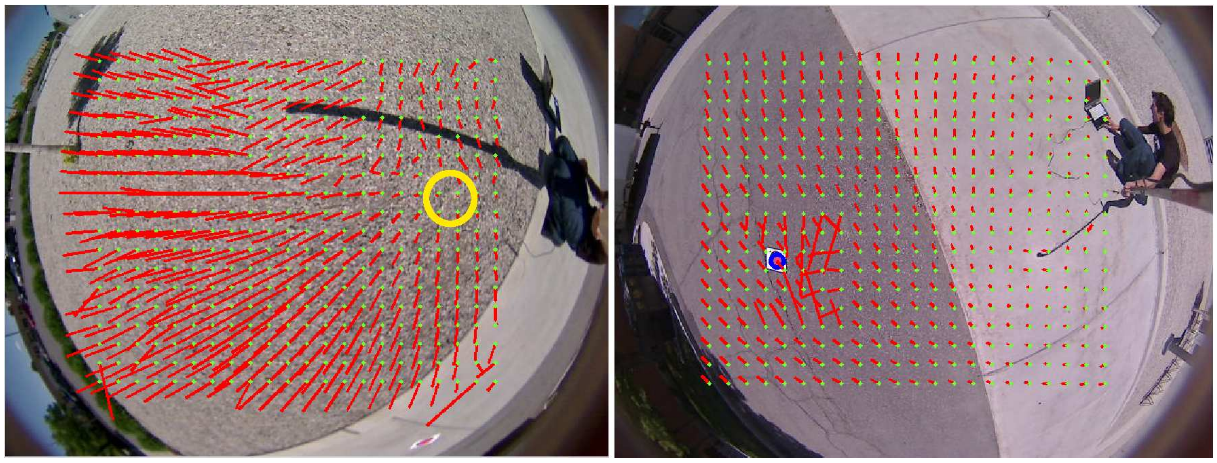


Figure 5.3: Optical flow estimation. *Left:* Optical flow in close proximity (below 1 m) to the ground (FOE marked yellow), *Right:* Optical flow at approximately 2 m altitude. Note the large error in vicinity of the detected target (blue circle).

there is a feature rich background and significant vertical movement) and we can localize the focus of expansion (assuming non-zero velocity in vertical direction). However, at altitudes ≥ 1 m the calculated optical flow is ambiguous and suffer from large errors. This is caused firstly by featureless background (such as an asphalt road), where it is difficult to track the motion of pixels. Second, if the vertical velocity is small in comparison to altitude, there is simply not enough movement between the consecutive images (as already noted in Section 4.2.3. Thus FOE estimation is not possible. Note that FOE is found easily for rotational movement, as long as the axis of rotation lies within the image plane.

Because of the aforementioned reasons, OF calculation and FOE estimation is not reliable enough to be used as an input for the VTOL autopilot during the whole landing/take-off manoeuvre. However, it provides sufficient precision in close vicinity of the ground (i.e. below 1 m) and shall be used in combination with ultrasonic range measurements.

5.2 Conclusion

In the presented work, the potential of visual servoing techniques for VTOL platforms was explored. First, an overview of UAV applications and state-of-the-art quadrotors were introduced in Chapter 1. A necessary theoretical background of quadrotor modelling and visual servoing was provided in Sections 1.3 and 2. The solutions based on visual servoing for the most important flight tasks (such as vertical take-off and landing, obstacle avoidance and attitude estimation) were described in Chapter 3. Finally, a VTOL autopilot for a medium-sized outdoor quadrotor was developed and described in detail in Chapter 4.

Note that it is intended to use the developed VTOL autopilot for future versions of the AggieVTOL platform (the quadrotor developed at Utah State University, see Appendix A). For that reason all the necessary software and hardware settings were described in the Appendix, so this thesis can serve as a reference for future use.

As mentioned in the beginning of this chapter, the reason why only the preliminary results are included is, that no flying platform was available upon the submission deadline of this thesis. The AggieVTOL platform equipped with the VTOL autopilot is being prepared for AUVSI Student competition of unmanned aerial vehicles[2], which takes place on June 15 2012 in Maryland. The flight tests and results of the competition are to be presented during the defence of this thesis.

[2] *http://www.auvsi-seafarer.org/*

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Samir Bouabdallah. *Design and control of quadrotors with application to autonomous flying.* PhD thesis, EPFL Lausanne, 2007.

[3] Jean-Yves Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm.* Intel Corporation Microprocessor Research Labs, 2000.

[4] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly, 2008.

[5] Pierre-Jean Bristeau, Philippe Martin, Erwan Salaün, and Nicolas Petit. The role of propeller aerodynamics in the model of a quadrotor UAV, August 2009.

[6] A. Cesetti, E. Frontoni, A. Mancini, and P. Zingaretti. Autonomous safe landing of a vision guided helicopter. In *2010 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA)*, pages 125–130. IEEE, July 2010.

[7] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics & Automation Magazine, IEEE*, 13(4):82–90, December 2006.

[8] F. Chaumette and S. Hutchinson. Visual servo control. II. advanced approaches [Tutorial]. *Robotics & Automation Magazine, IEEE*, 14(1):109–118, 2007.

[9] DaLi Chen, YangQuan Chen, and Hu Sheng. Fractional variational optical flow model for motion estimation. In *Proceedings of FDA'10*, Badajoz, Spain, 2010.

[10] Jay Hyuk Choi, Dongjin Lee, and Hyochoong Bang. Tracking an unknown moving target from UAV: extracting and localizing an moving target with vision sensor based

on optical flow. In *2011 5th International Conference on Automation, Robotics and Applications (ICARA)*, pages 384–389. IEEE, December 2011.

[11] Roberto Cipolla and Andrew Blake. Image divergence and deformation from closed curves, 1997.

[12] P.I. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 1st edition, 2011.

[13] Aaron Dennis, Christopher Hoffin, Jacob Marble, and Brandon Stark. Rotary-Wing open source autonomous miniature (ROSAM) unmanned aerial system (UAS) design and implementation for the 2011 AUVSI SUAS competition, May 2011.

[14] L. Derafa, T. Madani, and A. Benallegue. Dynamic modelling and experimental identification of four rotors helicopter parameters. In *IEEE International Conference on Industrial Technology, 2006. ICIT 2006*, pages 1834–1839. IEEE, December 2006.

[15] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006.

[16] Jaromír Dvořák. *Micro Quadrotor: Design, Modelling, Identification and Control*. PhD thesis, Czech Technical University in Prague, Prague, 2011.

[17] M. Euston, P. Coote, R. Mahony, Jonghyuk Kim, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing UAV. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 340 –345, September 2008.

[18] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[19] Tobias Fromm. *Visual Flight Support for Low-Cost Personal Remote Sensing Systems*. Master thesis in computer science, Hochschule Ravensburg-Weingarten, 2011.

[20] W.E. Green, P.Y. Oh, and G. Barrows. Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2347 – 2352 Vol.3, May 2004.

[21] C. Harris and J. Pike. 3d positional integration from image sequences. 1988.

[22] B. Herisse, S. Oustrieres, T. Hamel, R. Mahony, and F. -X Russotto. A general optical flow based terrain-following strategy for a VTOL UAV using multiple views. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3341–3348. IEEE, May 2010.

[23] B. Herisse, F. -X Russotto, T. Hamel, and R. Mahony. Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 801–806. IEEE, September 2008.

[24] Deokhwa Hong, Hyunki Lee, Hyungsuck Cho, Youngjin Park, and Jong Hyung Kim. Visual gyroscope: Integration of visual information with gyroscope for attitude measurement of mobile platform. In *International Conference on Control, Automation and Systems, 2008. ICCAS 2008*, pages 503–507. IEEE, October 2008.

[25] Berthold Horn and Brian Schunck. Determining optical flow. In *Artificial intelligence 17*, pages 185–205. 1981.

[26] Myung Hwangbo and T. Kanade. Visual-inertial UAV attitude estimation using urban scene regularities. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2451–2458. IEEE, May 2011.

[27] Jonathan Kelly and Gaurav S. Sukhatme. *An Experimental Study of Aerial Stereo Visual Odometry.* 2007.

[28] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large scale visual odometry for rough terrain. In *In Proc. International Symposium on Robotics Research*, 2007.

[29] S. Lange, N. Sunderhauf, and P. Protzel. 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, 2009. ICAR 2009*, pages 1–6. IEEE, June 2009.

[30] D. N Lee, M. N. O Davies, P. R Green, and F. R (Ruud). Van Der Weel. Visual control of velocity of approach by pigeons when landing. *Journal of Experimental Biology*, 180(1):85–104, July 1993.

[31] J. Gordon Leishman. *Principles of Helicopter Aerodynamics.* Cambridge University Press, December 2002.

[32] H. Li. Global interpretation of optical flow field: a least-squares approach. In *, 11th IAPR International Conference on Pattern Recognition, 1992. Vol.I. Conference A: Computer Vision and Applications, Proceedings*, pages 668–671. IEEE, September 1992.

[33] David G. Lowe. Distinctive image features from Scale-Invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[34] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision, 1981.

[35] A. Martinelli. Vision and IMU data fusion: Closed-Form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44–60, February 2012.

[36] C. McCarthy, N. Barnes, and R. Mahony. A robust docking strategy for a mobile robot using flow field divergence. *IEEE Transactions on Robotics*, 24(4):832–842, August 2008.

[37] Chris McCarthy and Nick Barnes. A unified strategy for landing and docking using spherical flow divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):1024–1031, May 2012.

[38] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, January 2012.

[39] A.I. Mourikis and S.I. Roumeliotis. A Multi-State constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565 –3572, April 2007.

[40] P. Newman, J. Leonard, J.D. Tardos, and J. Neira. Explore and return: experimental validation of real-time concurrent mapping and localization. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1802 – 1809 vol.2, 2002.

[41] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, 2010.

[42] Paul Pounds, Robert Mahony, and Joel Gresham. Towards dynamically favourable Quad-Rotor aerial robots, 2004.

[43] D. I.B Randeniya, S. Sarkar, and M. Gunaratne. Vision–IMU integration using a Slow-Frame-Rate monocular vision system in an actual roadway setting. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):256–266, June 2010.

[44] D. Scaramuzza and F. Fraundorfer. Visual odometry [Tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80 –92, December 2011.

[45] P. Serra, F. Le Bras, T. Hamel, C. Silvestre, and R. Cunha. Nonlinear IBVS controller for the flare maneuver of fixed-wing aircraft using optical flow. pages 1656–1661, 2010.

[46] C.S. Sharp, O. Shakernia, and S.S. Sastry. A vision system for landing an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1720 – 1727 vol.2, 2001.

[47] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, June 1994.

[48] Daquan Tang, Fei Li, Ning Shen, and Shaojun Guo. UAV attitude and position estimation for vision-based landing. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 9, pages 4446 –4450, August 2011.

[49] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz. A new approach to vision-aided inertial navigation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4161 –4168, October 2010.

[50] T. Templeton, D.H. Shim, C. Geyer, and S.S. Sastry. Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1349 – 1356, April 2007.

[51] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874, November 2011.

Appendix A

AggieVTOL quadrotor

In this section a detailed description of the VTOL platform from Utah State University CSOIS lab is provided. This platform is developed for scientific and eventually commercial use, specifically for personal remote sensing. Unlike the flying wing platforms, VTOL's main strength is a precise imaging of a specific place, rather than covering a large area. VTOL and fixed wing UAVs can cooperate in such a way, that the area is firstly scanned by the fixed wing platform, and consequentially a high-resolution images of the points of interest are taken by VTOL.

Note that initially an octorotor platform with increased payload capability was developed, however due to vibration issues the effort was aimed at a quadrotor platform instead. AggieVTOL is shown in Figures A.1, A.2 and A.3.

Mechanical design

Dimensions

- Length (without propellers): 81 cm
- Height (without legs and payload): 10 cm
- Total take-off mass (without payload): 2500 g
- Maximal allowed take-off mass: 3100 g

Materials

Whole airframe is made from carbon fibre, the arms are made from commercially available carbon fibre tubes (diameter 15 mm), the payload and battery box is custom made. The

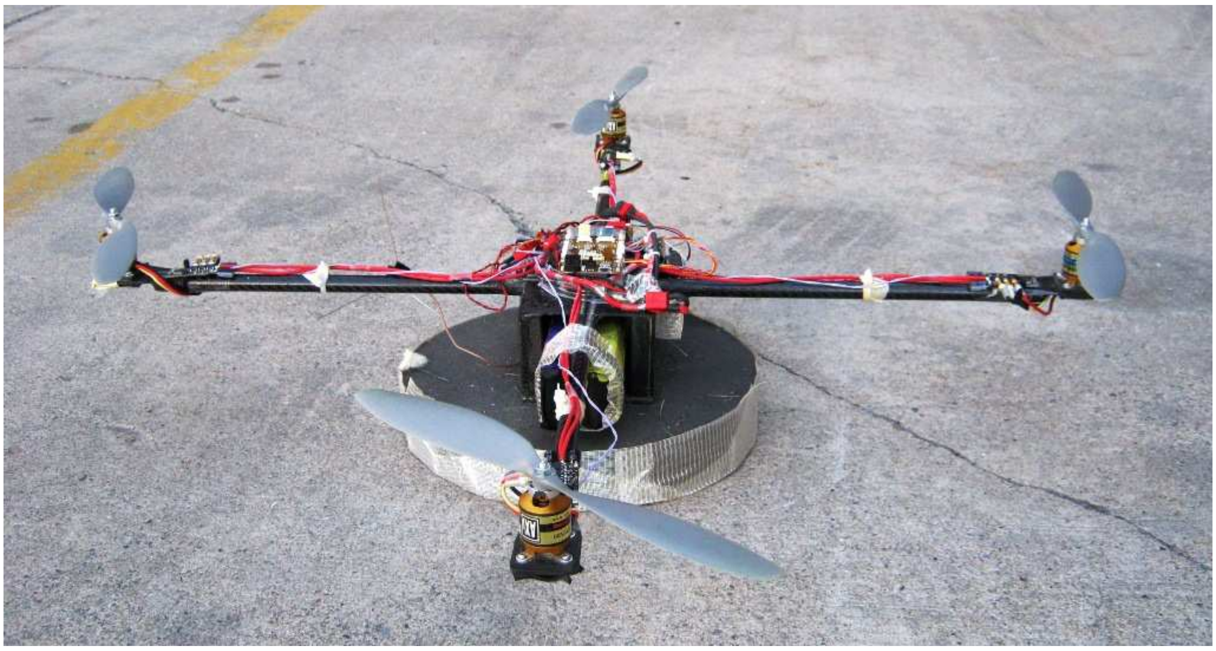


Figure A.1: AggieVTOL without landing gear (Styrofoam landing pad is used instead during attitude controllers tuning)

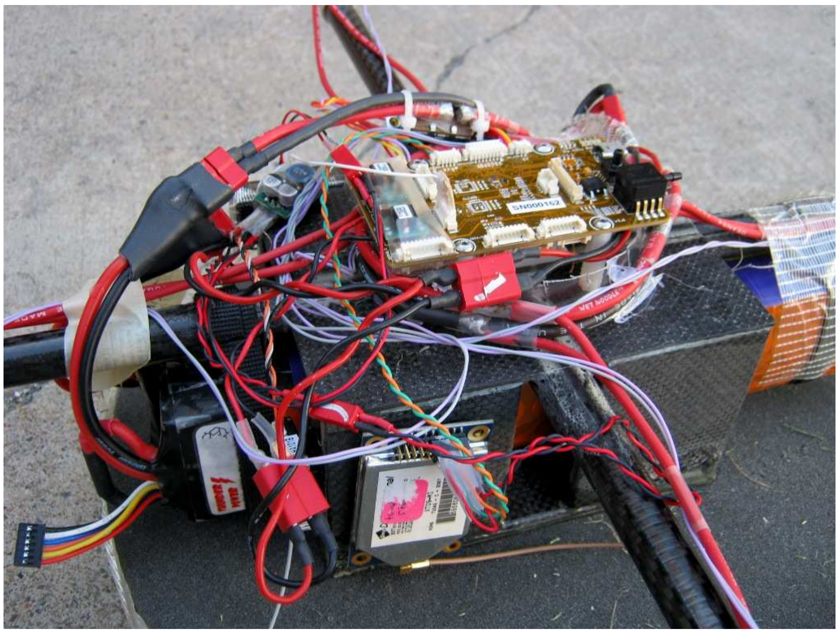


Figure A.2: AggieVTOL electronics - LISA board on the top, GMX IMU hidden below, radio control transmitter on the side of the battery box

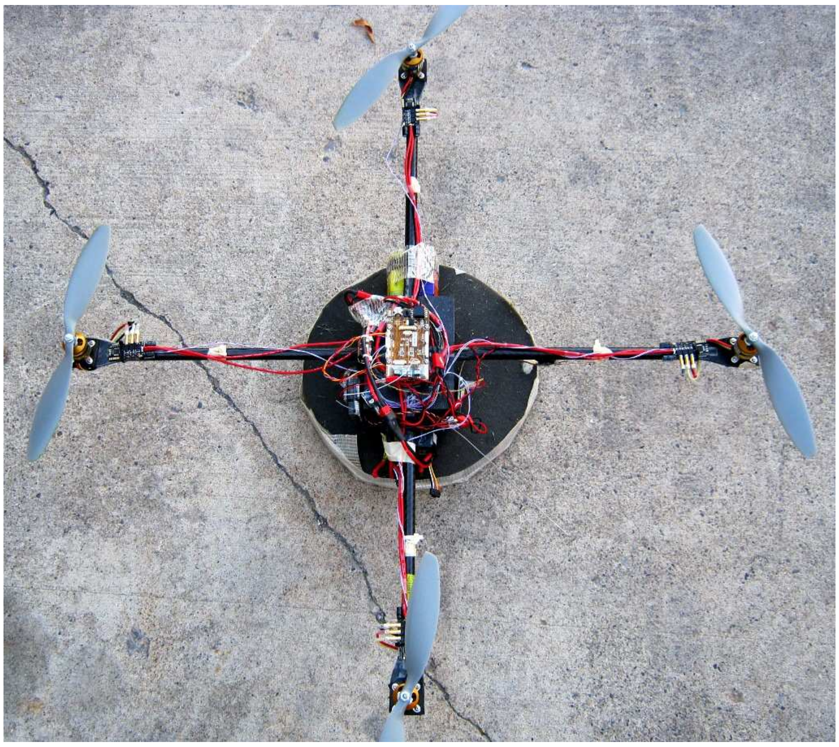


Figure A.3: Top view of AggieVTOL

only metal part is the middle cross which hold together the four arms.

Motors

Motors from Czech company *AXI* are used, specifically *AXI Gold 2217/20 Outrunner Motor*[1]. Motor *Avroto 2814 Short Shaft*[2] has a similar performance and can be considered as a backup solution. 12×3.5 SFP *APC* propellers are used, model *LP12038SFP*[3].

Estimated current taken at nominal thrust (i.e. when the airframe hovers, around 50% of the thrust) is around 4.5 A per motor. The propeller is a standard off-the-shelf propeller for RC motors. Estimated thrust of each motor is around 1.3 kg.

Battery

Two *Thunder Power*[4] Li-Polymer batteries are used, each with the capacity 8 Ah and voltage 14.8 V. Weight of a single battery is 630 g, thus batteries make the majority of

[1] *http://www.modelmotors.cz/*
[2] *http://montorc.com/M2814Shaft.aspx*
[3] *http://www.apcprop.com/ProductDetails.asp?ProductCode=LP12038SF*
[4] *http://www.thunderpowerrc.com/*

the airframe weight. With these batteries and maximal payload is the expected mission duration around 20 minutes.

Payload

The payload is *Canon PowerShot S95*[5] camera (weight around 170 g). alternatively an infra-red camera of similar weight can be used.

Hardware

AggieVTOL is based on *Papparazi*[6] open-source project which comes up with both hardware and software solutions for UAVs. *LISA* main board is used for the autopilot. *LISA* board is driven by a STM32 processor, a 32-bit ARM Cortex MCU that operates at 72MHz. While a separate IMU is necessary, several interfaces are available including SPI, UART and I2C for communications. On-board features include power management, two barometric sensors and a connector for a separate GUMSTIX5 Overo single board computer which features a 600 MHz OMAP3530 processor and USB 2.0 connectivity[13]. As IMU servers *3DM-GX3*[7] high-end navigation unit.

System architecture

An overview of the on-board system architecture can be seen in Figure A.4. In this system, four sensor sources are used to calculate the attitude and the position. The 6 DoF IMU and the magnetometer are fused together by the Attitude and Heading Reference System (AHRS) module. The AHRS provides this data to the Inertial Navigation System (INS) along with positioning data from a GPS module and a barometer. These data are then processed to form an estimation of the vehicle's true position and altitude. The attitude and position information is used by the controller, along with any control information from either an RC transmitter at 2.4GHz for manual control or a Ground Control Station (GCS) via the 900Hz modem. The actuators for the UAV are controlled by ESCs. A payload can be controlled autonomously by the controller or manually through the GCS. The GCS provides the operator with real-time status information as well as the ability to form navigational paths for the UAV to follow[13].

[5] *http://usa.canon.com/cusa/consumer/products/cameras/digital_cameras/powershot_s95*
[6] *http://paparazzi.enac.fr/*
[7] *http://www.microstrain.com/inertial/3DM-GX3-25*

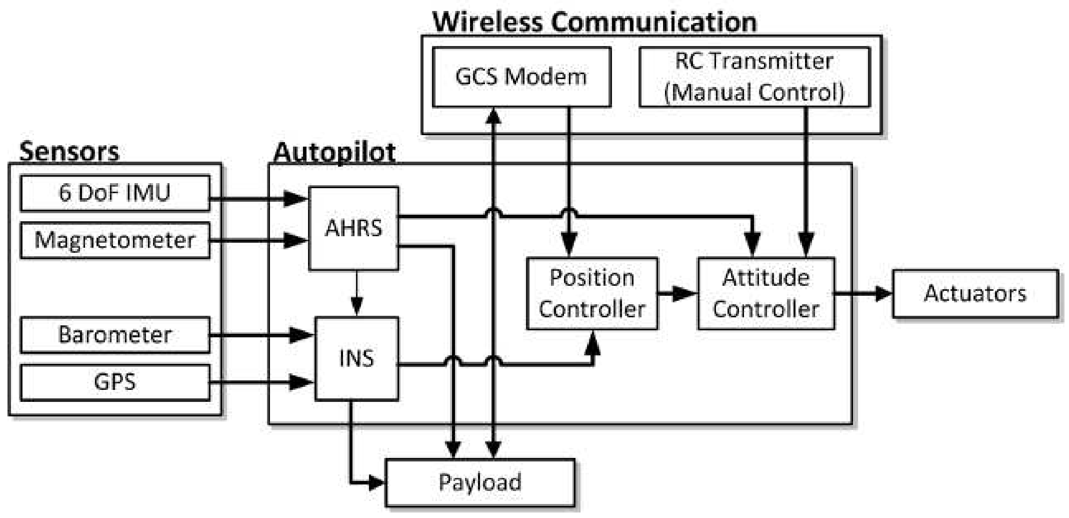


Figure A.4: Quadrotor software architecture

The velocity control is provided by Electronic Speed Controller (ESC) from *Mikrokopter*[8]. It has the maximum current capability of 35A and uses I2C to communicate with the autopilot[13].

The attitude and position control systems within the LISA autopilot are simple PID controllers with feed forward. The control of Qaudrotors is certainly a well explored topic, however its commonly accepted a simple PID controller is adequate for attitude control. Position control as well can be accomplished with a simple PID controller. The number of control systems that have tried for attitude and position control are vast and ever expanding. However due to their simplicity and reliability simple PID controllers were better for for AggieVTOL[13].

The VTOL platform can operate in three modes:

Manual - full manual control over RC transmitter

Auto1 - manual control with attitude stabilization

Auto2 - fully automatic control including way-point following

Communication channels

AggieVTOL uses the following communication channels:

[8] *http://www.mikrokopter.de/*

Telemetry - Digi Xtend 900 MHz serial modem[9]

Radio Control - Spectrum DX7s 2.4 GHz RC Transmitter[10]

Payload - Ubiquiti Networks Bullet 5 GHz transmitter[11]

For safety reasons, a *killswitch* which immediately shuts down all power is implemented on the RC transmitter.

Ground control station

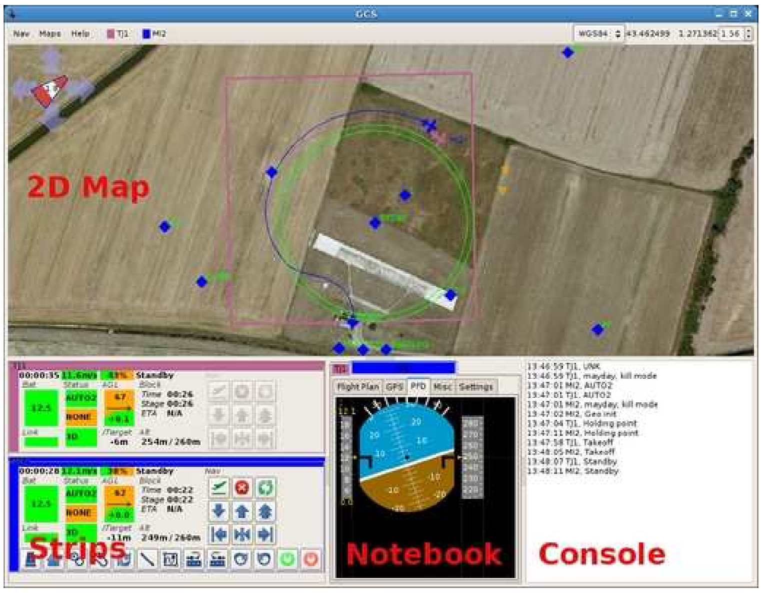
The Paparazzi Center allows for all configuration settings relevant to flight to be defined. This includes the flightpath set, and various other settings such as the airframe configuration file. Within the airframe configuration file parameters such as the type of actuator, Com link, IMU, AHRS and controller tuning are defined. Once all the configurations have been set, the code is compiled and can then be uploaded to the LISA board via USB. Once uploaded the center can execute the program, this will bring up the GCS with the uploaded flightpath set, as well as other settings[13].

The Paparazzi GCS (ground control station) consists of several tools including the flight plan editor, real-time displays and reconfigurable switches. The Paparazzi GCS also provides the GCS operator with the ability to re-task the UAV in the event of emergent targets. Paparazzi GCS is shown in Figures A.5 and A.6. Further description can be found on the Paparazzi web page.

[9] *http://store.digi.com/index.cfm?fuseaction=product.display&Product_ID=490*

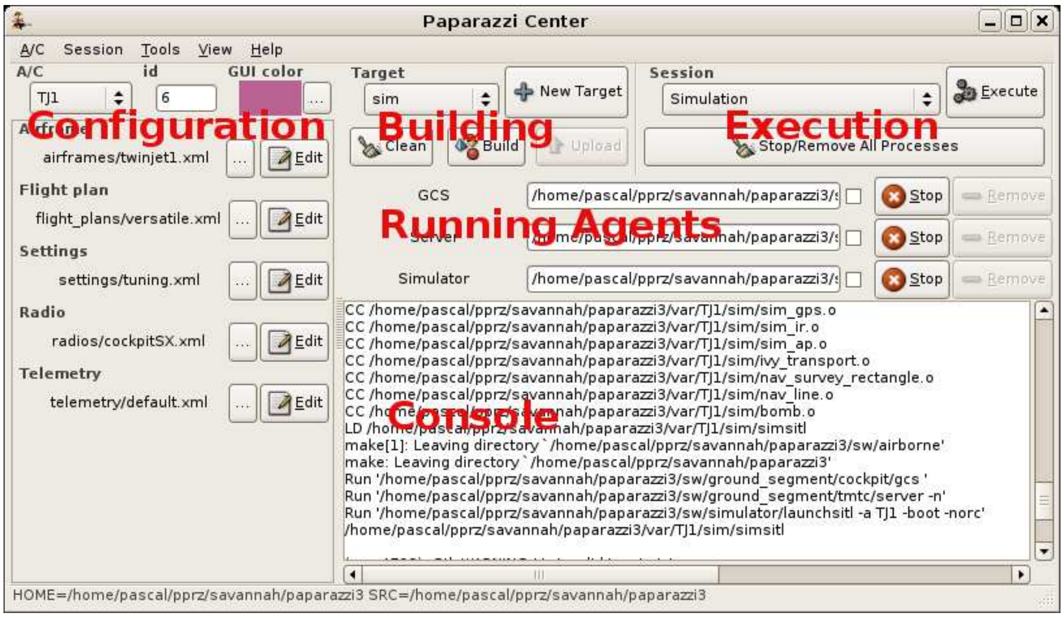
[10] *http://www.spektrumrc.com/Products/Default.aspx?ProdId=SPM7800*

[11] *http://www.ubnt.com/bullet*



```
13:46:59 TJ1, UNK
13:46:59 TJ1, mayday, kill mode
13:47:01 MI2, AUTO2
13:47:01 TJ1, AUTO2
13:47:01 MI2, mayday, kill mode
13:47:02 MI2, Geo init
13:47:04 TJ1, Holding point
13:47:11 MI2, Holding point
13:47:58 TJ1, Takeoff
13:48:05 MI2, Takeoff
13:48:07 TJ1, Standby
13:48:11 MI2, Standby
```

Figure A.5: Paparazzi GCS (*http://paparazzi.enac.fr/wiki/GCS*)



A/C: TJ1 id: 6 GUI color

Airframe: airframes/twinjet1.xml

Flight plan: flight_plans/versatile.xml

Settings: settings/tuning.xml

Radio: radios/cockpitSX.xml

Telemetry: telemetry/default.xml

Target: sim

Session: Simulation

GCS /home/pascal/pprz/savannah/paparazzi3/s

Server /home/pascal/pprz/savannah/paparazzi3/s

Simulator /home/pascal/pprz/savannah/paparazzi3/s

```
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/sim_gps.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/sim_ir.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/sim_ap.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/ivy_transport.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/nav_survey_rectangle.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/nav_line.o
CC /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/bomb.o
LD /home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/simsitl
make[1]: Leaving directory `/home/pascal/pprz/savannah/paparazzi3/sw/airborne'
make: Leaving directory `/home/pascal/pprz/savannah/paparazzi3'
Run '/home/pascal/pprz/savannah/paparazzi3/sw/ground_segment/cockpit/gcs '
Run '/home/pascal/pprz/savannah/paparazzi3/sw/ground_segment/tmtc/server -n'
Run '/home/pascal/pprz/savannah/paparazzi3/sw/simulator/launchsitl -a TJ1 -boot -norc'
/home/pascal/pprz/savannah/paparazzi3/var/TJ1/sim/simsitl
```

HOME=/home/pascal/pprz/savannah/paparazzi3 SRC=/home/pascal/pprz/savannah/paparazzi3

Figure A.6: Paparazzi Control Center (*http://paparazzi.enac.fr/wiki/*)

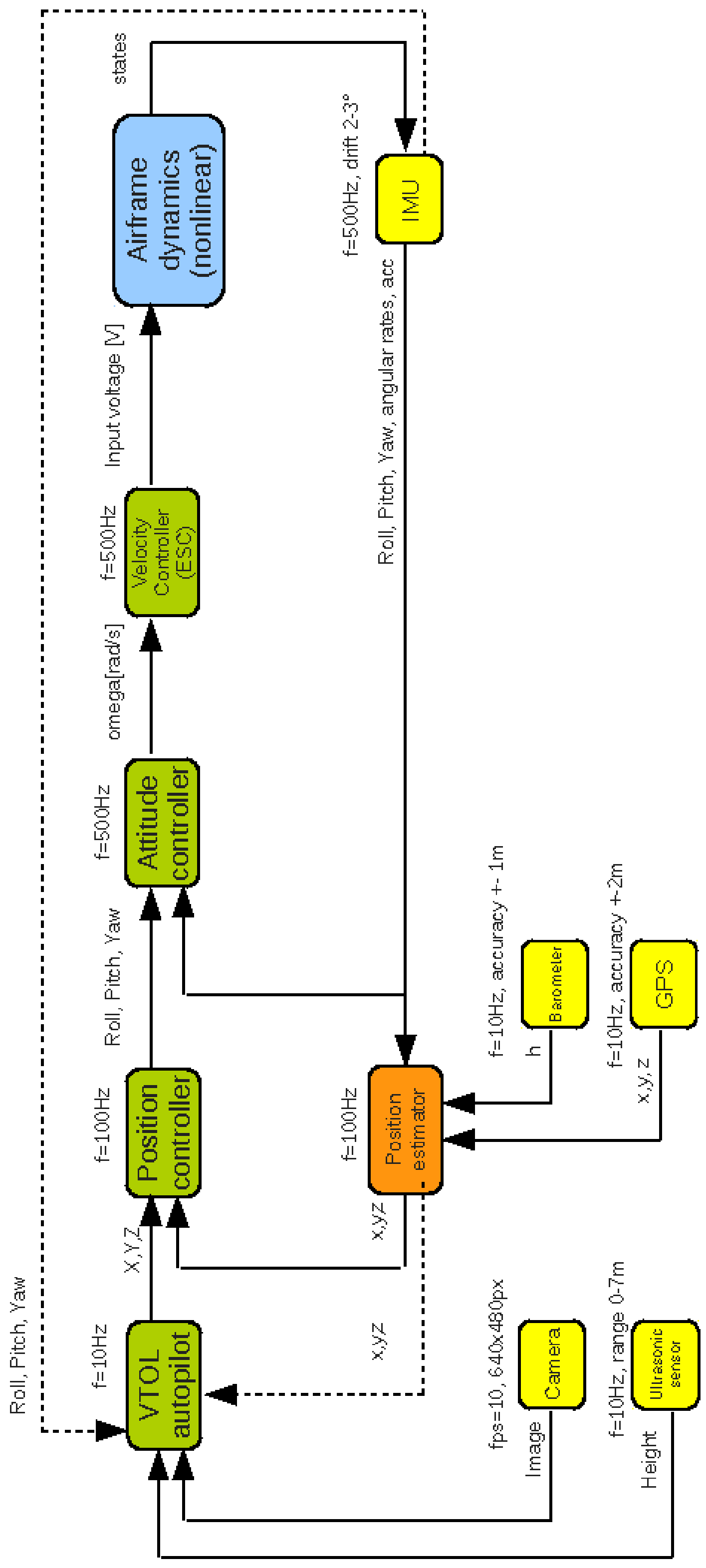


Figure A.7: Quadrotor controllers block diagram

Appendix B

Camera setup

The following components are needed:

- Logitech Webcam C600
- Sunex DSL125 lens
- Sunex CMT107 lens mount

A step-by-step tear-down of the Logitech camera is provided at *http://www.ifixit.com/Teardown/Logitech-2-MP-Webcam-C600-Teardown/1504/1*. Follow the instructions there.

The CMT107 mount thread is 16 mm high, however DSL125 has only 12.5 mm long thread. Thus the mount has to be shortened by at least 2.5 mm. The CMT107 is also wider than the original Logitech lens mount, and interferes with on-board resistors and capacitors. It has to be sanded off on the edges to fit on the board. Finally hot glue was used to seal the mount and to protect the CCD sensor from dirt.

The board with the camera is screwed on a frame from balsa wood. This frame is then glued on the airframe. A carbon fibre box (it is light and durable) was made to protect both the camera and lens. The procedure as well as the final setting is illustrated in Figures B.1– B.5.

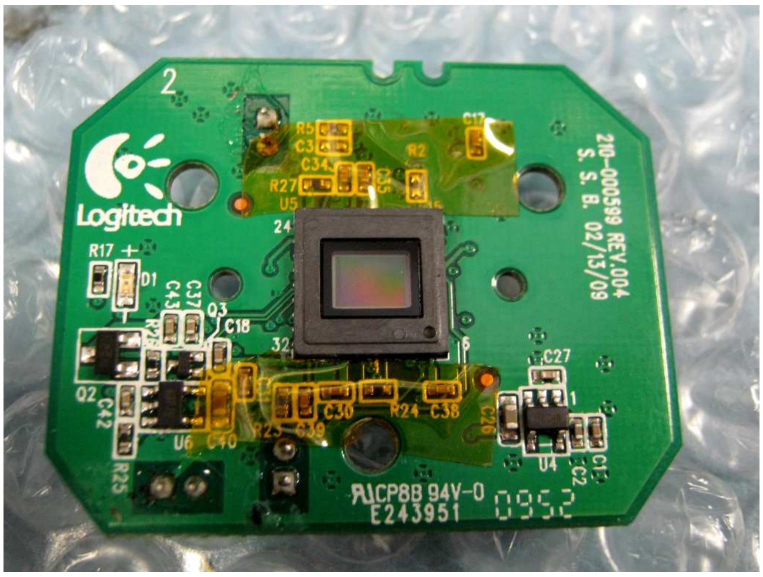


Figure B.1: Logitech Webcam C600 with removed lens and a bare 1/3.2” CCD sensor

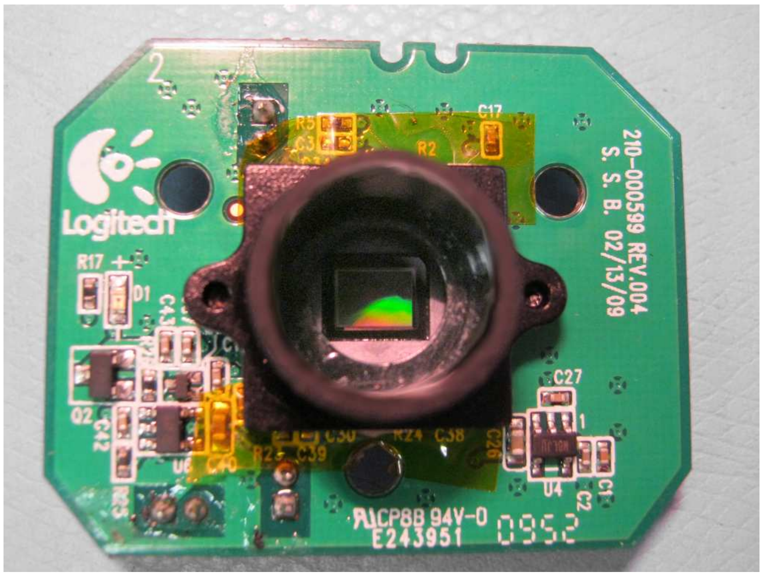


Figure B.2: Logitech Webcam C600 with CMT107 M12 tits lens mount. Note the cover tape in vicinity of the mount.

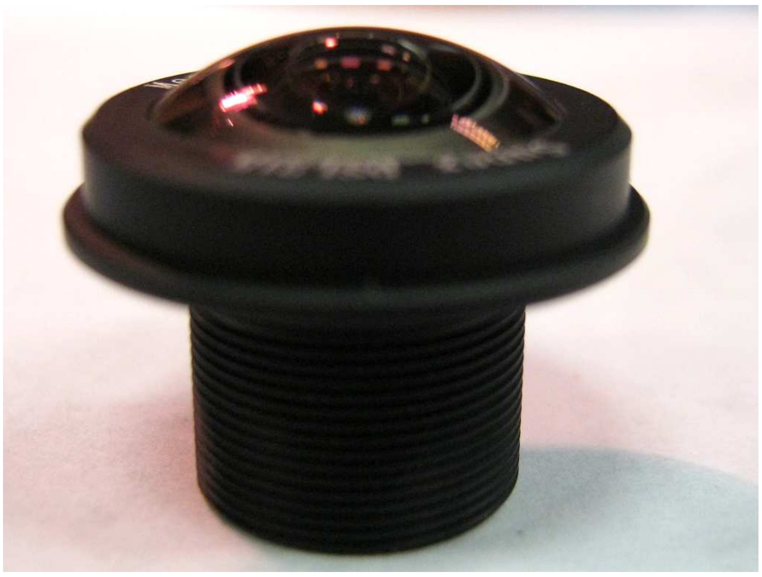


Figure B.3: Fisheye DSL125 lens

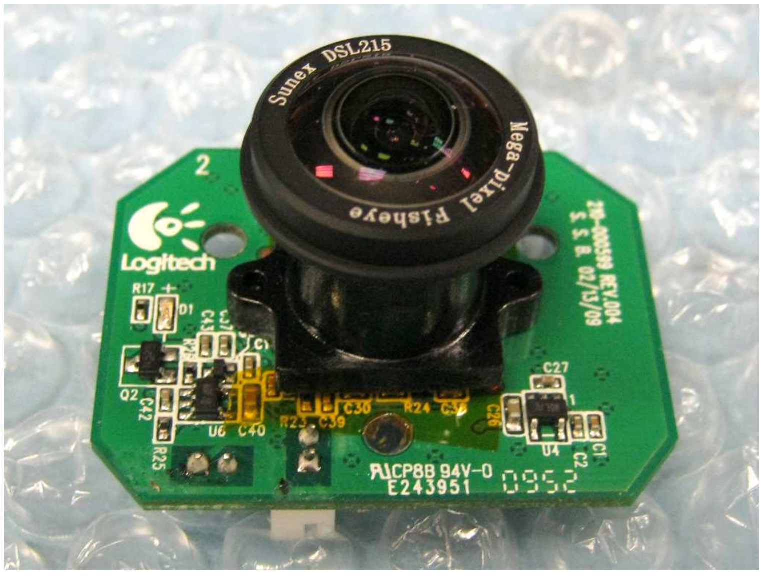


Figure B.4: Logitech Webcam C600 with mounted and sealed DSL125

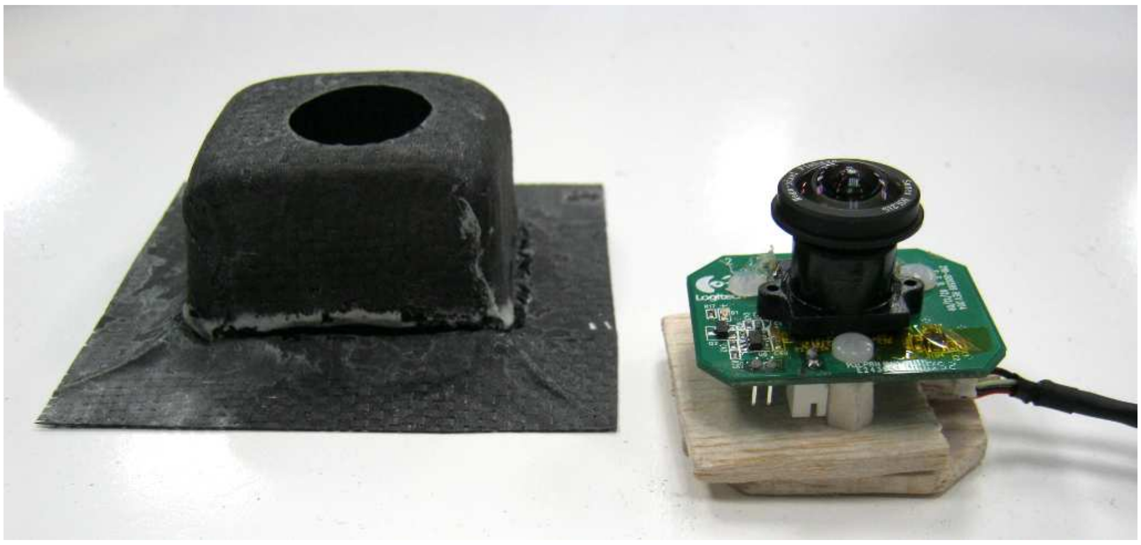


Figure B.5: Camera on a balsa wood frame and the carbon fibre box.

Appendix C

Panda board set-up

The following components are needed:

- *Pandaboard*
- XL-MaxSonar MB1210
- Custom made A/D converter board

A detailed description and system reference manual for *Pandaboard* can be found at *http://pandaboard.org/*. Installation of operating system (*Ubuntu 12.04 Desktop*), *OpenCV* and additional libraries is described at *https://sites.google.com/site/andrewssobral/pandaboard*. When this set-up is done, we have to add connectors for I2C communication. The following pins on J3 expansion connector are used:

- Pin 1: 5V DC power
- Pin 23: I2C Serial Data
- Pin 24: I2C Clock
- Pin 25, 26: Digital ground

A layout and circuit diagram of the custom made A/D converter board with Digi-Key AD7997BRUZ-1-ND chip[1] is shown in Figures C.1 and C.2. The ultrasonic sensor are connected on Pin 3, which provides analog output. The other input is grounded.

[1] *http://search.digikey.com/us/en/products/AD7997BRUZ-1/AD7997BRUZ-1-ND/998053*

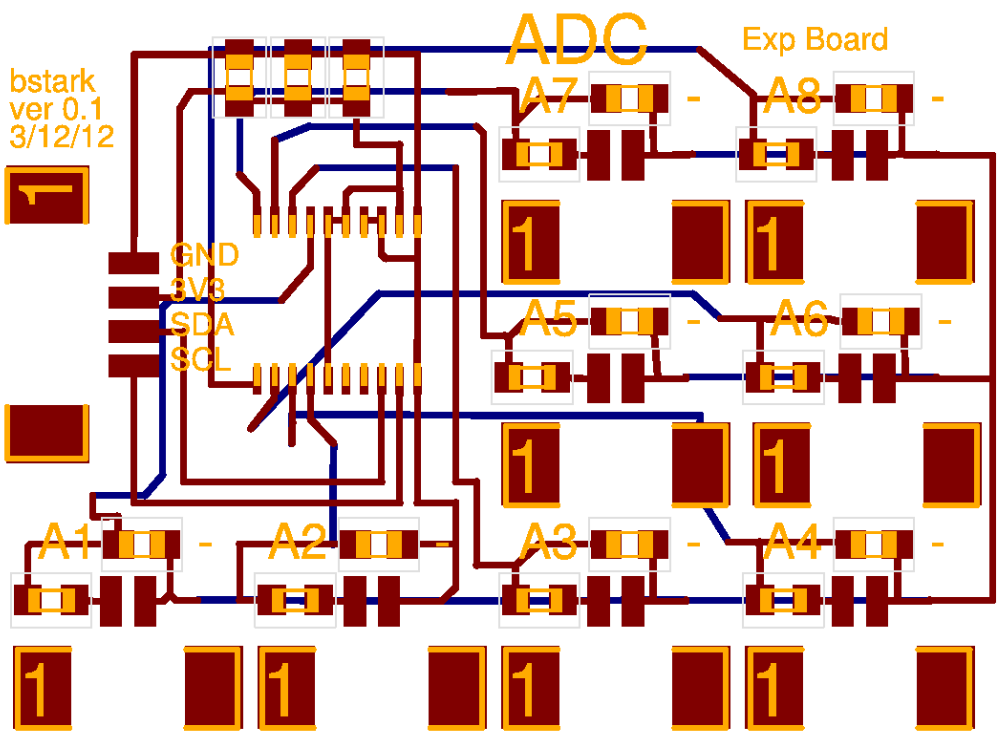


Figure C.1: Layout of the A/D converter board
(*Courtesy of Brandon Stark*)

The board does not provide power supply, so a separate 5 V power channel has to be established between the *Pandaboard* and the sensor.

For reading the analog output from the sensors, an analog low-pass (RC) filter had to be used. Update rate of the sensor is 10 Hz, bandwidth of the filter is 6 Hz (using $C = 2.2\mu F$, $R = 12k\Omega$). Capacitors at the power supply line has values $C_1 = 0.1\mu F$, $C_2 = 100\mu F$, $C_3 = 0.1\mu F$.

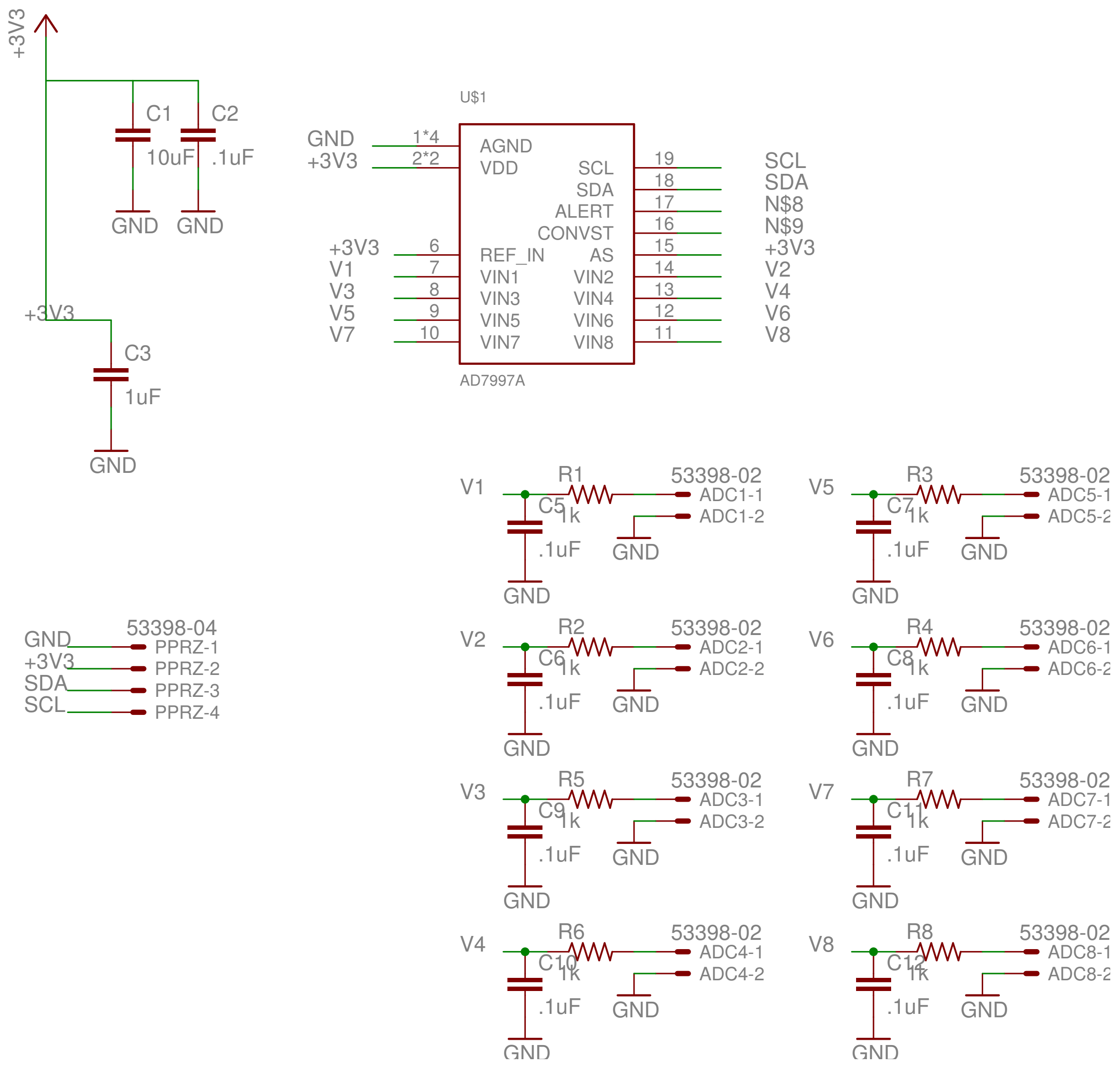


Figure C.2: Circuit diagram of the A/D converter board
(*Courtesy of Brandon Stark*)

Appendix D

Landing pad design

In this section a method of determining the optimal size of the landing pad (the target) is described. The requirements are that it has to be possible to track the target reliably from the initial height of 5 m, and a whole target has to be visible during the final part of the descend (e.g. at height of 2 m), as described in Section 4. Below this altitude ultrasonic sensors are the main source of position information and visibility of whole target is not required.

Fisheye imaging model

A fisheye lens DSL215 with horizontal FOV 175° is used. Assume that the principal point is in the middle of the image plane. Image formation for a fisheye lens is described by the following equations[12]:

$$R = \sqrt{X^2 + Y^2 + Z^2} \tag{D.1}$$

$$\theta = \arccos(\frac{R}{Z}) \tag{D.2}$$

$$\phi = \arctan(\frac{Y}{X}) \tag{D.3}$$

where X, Y, Z are the point coordinates in world reference frame. The point is projected to the image plane in polar coordinates (r, ϕ) with respect to the principal point. $r = r(\theta)$ and the mapping function depends on the type of the lens. In case od DSL125 an equiangular projection[1] is used, so we can approximate the equation as $r = k\theta$. Having a 640×480 px image, $k \doteq 203$ (assuming $\theta \geq 0$). The conversion from polar coordinates to pixel coordinates (u, v) is:

$$u = r\cos\phi \tag{D.4}$$

$$u = r\sin\phi \tag{D.5}$$

Putting equations D.1 and D.2 together, we obtain:

$$r = k \arccos\left(\frac{\sqrt{X^2 + Y^2 + Z^2}}{Z}\right) \tag{D.6}$$

Experimental results

To verify the presented calculations, a calibration scene was prepared. Three signs marking a center of a circle and 1 m radius were placed in the camera optical axis (i.e. $X = 0$ m), and then to 5 m and 10 m distance from the optical axis. A re-projection of these points was calculated using aforementioned equations and compared with the image.

In Figure D.1 we can see the result. At the centre of the image plane the re-projection concurs with the signs, however as we are getting further from the centre, the re-projection error increases. This is caused by imperfections of the lens (there is a certain angular distortion at the edges of the image which is not taken in account in the equiangular equation), and also by the offset of the optical axis (we can see that it is not exactly in the centre of the image, but slightly on the left). This offset is caused by the camera mount.

Despite these imperfections, the used projection model provides a sufficient accuracy and can be used for calculating the re-projection of the target. Two sized of the landing pad were considered. One meter radius and 0.75 m radius. Assuming average accuracy of GPS being 2± m, we have to consider three cases. Target size at the distance of 5 m (the desired starting altitude), of 7 m (worst case starting altitude) and 2 m (altitude at which ultrasonic sensor are used for fine landing control). Figures D.2, D.3 and D.4

[1] *http://www.optics-online.com/Detail.asp?PN=DSL215B%2D690%2DF2%2E0*

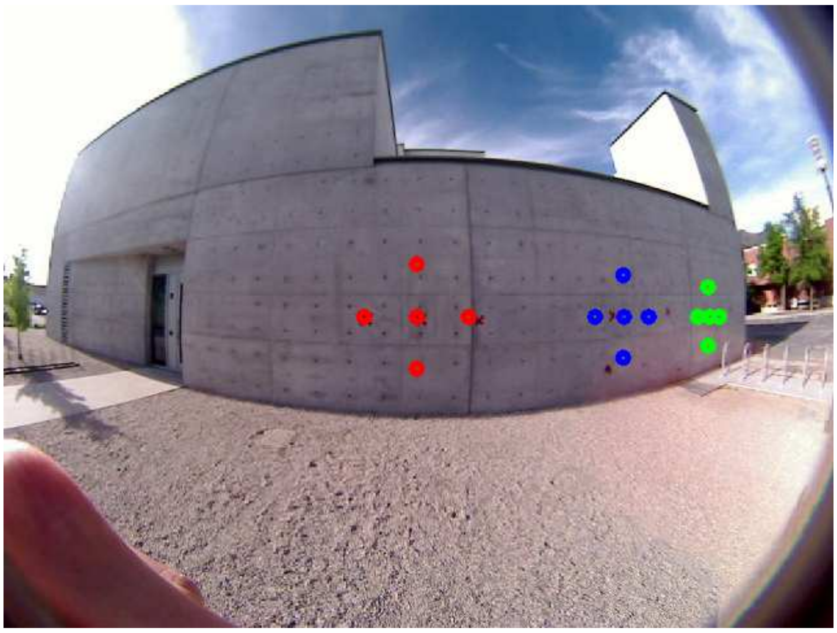


Figure D.1: The calibration scene, h=5 m, red crosses are the calibrating signs. Red, blue and green dots are representing the reprojection of a circle with 1 m diameter at 0, 5 and 10 meters right from the camera optical axis.

shows the target size at various heights (red, blue and green dots). Calibration marks are also shown (red crosses).

As can be seen, the difference in the landing pad size is most visible at higher altitude. However, the smaller size ($r = 0.75$ m) is considered as sufficient for a reliable target tracking. Thus the final size of the landing pad is a circle with radius of 0.75 m.

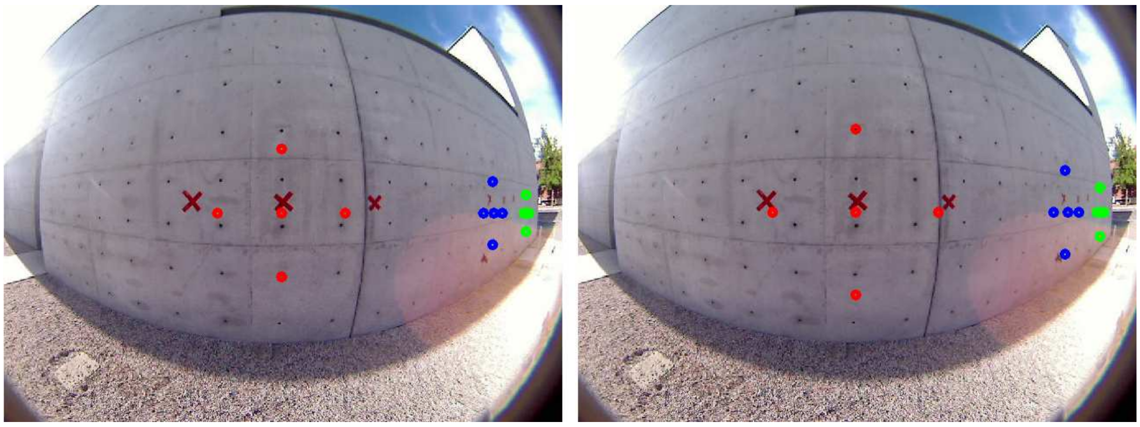


Figure D.2: Left: Height 2 m, target radius 0.75 m, re-projected at 0, 5, 10 m
Right: Height 2 m, target radius 1 m, re-projected at 0, 5, 10 m

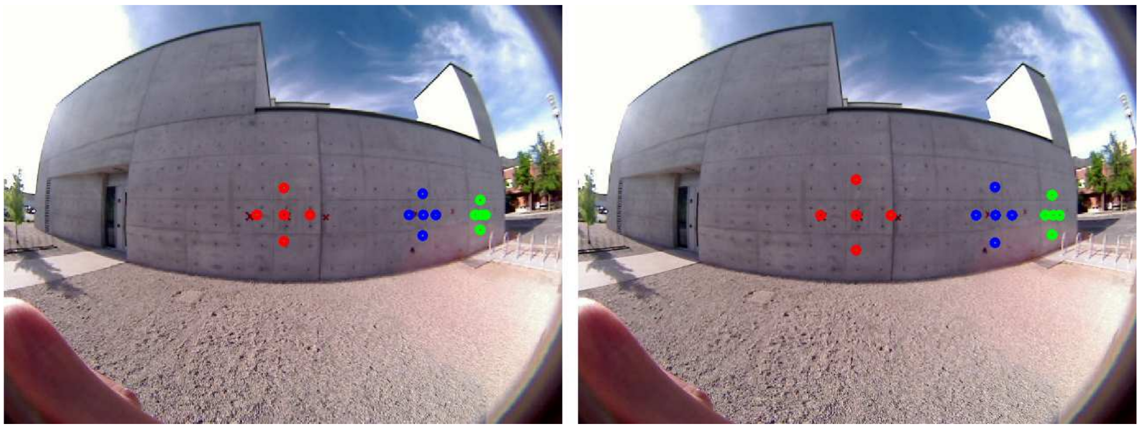


Figure D.3: Left: Height 5 m, target radius 0.75 m, re-projected at 0, 5, 10 m
Right: Height 5 m, target radius 1 m, re-projected at 0, 5, 10 m

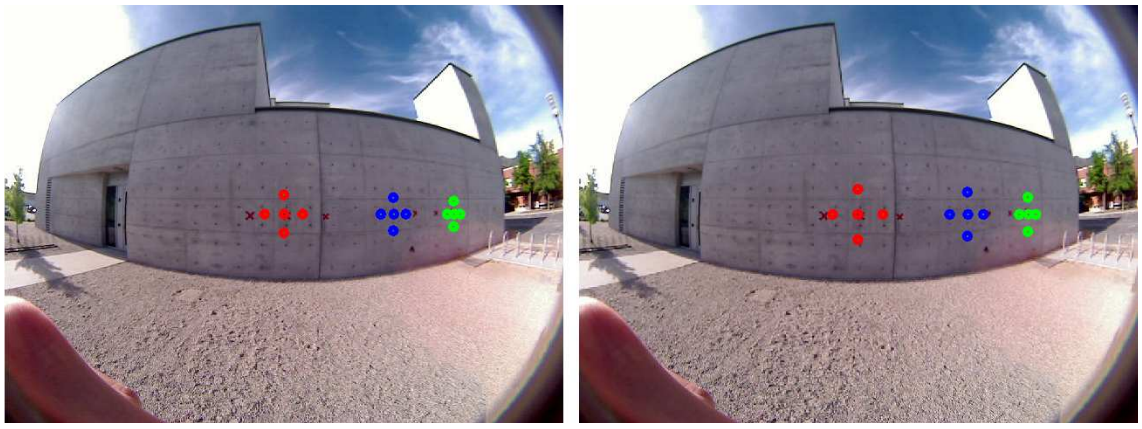


Figure D.4: Left: Height 7 m, target radius 0.75 m,
re-projected at 0, 5, 10 m
Right: Height 7 m, target radius 1 m,
re-projected at 0 m, 5 m, 10 m

Appendix E

Contents of the CD

The content on the CD is organized as follows:

- **AggieVTOL Nonlinear Model** - Simulink model of AggieVTOL platform described in Appendix A
- **Autopilot** - C++ code of the designed autopilot described in Chapter 4
- **Datasheets** - datasheets for hardware (*Pandaboard*, ultrasonic sensor, A/D converter), see Section 4.4
- **MATLAB** - related Matlab code
  - **code**
    - **general** - general purpose functions
    - **gui** - Graphical user interface for FOE estimation, Ground speed estimation and Optical flow calculation (Sections 4.2 and 4.3)
    - *cameraCalculations.m* - Script used for calculating the optimal size of the landing pad (including sample images)
  - **Flow** - implementation of Bronx and Sanders optical flow method (modification of Horn-Schunck method)
  - **FOHS MODEL** - implementation of Fractional Order Horn Schunck method
  - **houghcircle** - implementation of Hough circular transform
  - **HS** - implementation of the original Horn Schunck method
  - **lk** - implementation of Lukas Kanate Pyramidal Optical flow algorithm

  - **OpticalFlow** - implementation of Lukas Kanate Pyramidal Optical flow algorithm for fast computation using a graphical card
  - **Srinivasan** - fast Fourier transform based method for estimating field of expansion

- **Sample sets** - sample video and image sets presented in the thesis (an alternative for *youtube* viewing)
  - **Camera_calibration** - images used for camera calibration and landing pad design (see Appendix D)
  - **Flight_tests** - videos from Chapter 5
  - **FOE_estimation** - result of Matlab based FOE estimation, presented in Section 4.2
  - **Target_detection** - images shown in Section 4.3 for Matlab based Hough circular transform and landing spot detection
- **Status updates** - a collection of presentations held during the work on the thesis
- **Zotero** - references in Zotero format[1]
- *Podhradsky_Michal_thesis_2012_PRINT.pdf* - Printable version of the thesis
- *Podhradsky_Michal_thesis_2012_ONLINE.pdf* - Electronic version of the thesis for on-screen viewing

[1] *http://www.zotero.org/*