Linköping Studies in Science and Technology

Dissertation No. 1260

Vision-Based Localization and Guidance for Unmanned Aerial Vehicles

by

Gianpaolo Conte

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköping universitet
SE-581 83 Linköping, Sweden

Linköping 2009

ISBN 978-91-7393-603-3
ISSN 0345-7524
Printed by LiU-Tryck, Linköping, Sweden

To Antonio.

Preface

The thesis has been developed as part of the requirements for a PhD degree at the Artificial Intelligence and Integrated Computer System division (AIICS) in the Department of Computer and Information Sciences at Linköping University.

The work focuses on issues related to Unmanned Aerial Vehicle (UAV) navigation, in particular in the areas of guidance and vision-based autonomous flight in situations of short and long term GPS outage.

The thesis is divided into two parts. *Part I* (*Simulation and Guidance*) describes a helicopter simulator and a path following control mode developed and implemented on an experimental helicopter platform. *Part II* (*Vision-based Navigation*) presents an approach to the problem of vision-based state estimation for autonomous aerial platforms which makes use of geo-referenced images for localization purposes. The problem of vision-based landing is also addressed with emphasis on fusion between inertial sensors and video camera using an artificial landing pad as reference pattern. In the last chapter, a solution to a vision-based ground object geo-location problem using a fixed-wing micro aerial vehicle platform is proposed.

The helicopter guidance and vision-based navigation methods developed in the thesis have been implemented and tested in real flight-tests using a Yamaha Rmax helicopter. Extensive experimental flight-test results are presented.

Linköping, May 2009

Gianpaolo Conte

The work in this thesis is supported in part by grants from the Wallenberg Foundation, the SSF MOVIII strategic center, the NFFP04-031 "Autonomous flight control and decision making capabilities for Mini-UAVs" project grants and LinkLab, a Saab-LiU Center for Future Aviation Systems.

Related publications

The content of this thesis is based on the following publications:

Paper I *G. Conte, P. Doherty.* Vision-based Unmanned Aerial Vehicle Navigation Using Geo-referenced Information. *Accepted for publication in the EURASIP Journal of Advances in Signal Processing, 2009.*

Paper II *G. Conte, P. Doherty.* An Integrated UAV Navigation System Based on Aerial Image Matching. *In Proceedings of the IEEE Aerospace Conference. Big Sky, Montana, 2008.*

Paper III *G. Conte, M. Hempel, P. Rudol, D. Lundström, S. Duranti, M. Wzorek, and P. Doherty.* High Accuracy Ground Target Geolocation Using Autonomous Micro Aerial Vehicle Platforms. *In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit. Honolulu, Hawaii, 2008.*

Paper IV *S. Duranti, G. Conte.* In-flight Identification of the Augmented Flight Dynamics of the Rmax Unmanned Helicopter. *In Proceedings of the 17th IFAC Symp. on Automatic Control in Aerospace. Toulouse, 2007.*

Paper V *M. Wzorek, G. Conte, P. Rudol, T. Merz, S. Duranti, P. Doherty.* From Motion Planning to Control - A Navigation Framework for an Autonomous Unmanned Aerial Vehicle. *21th Bristol UAV Systems Conference, 2006.*

Paper VI *G. Conte, S. Duranti, T. Merz.* Dynamic 3D Path Following for an Autonomous Helicopter. *In Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles. Lisbon, 2004.*

Paper VII *T. Merz, S. Duranti, and G. Conte.* Autonomous Landing of an Unmanned Aerial Helicopter Based on Vision and Inertial Sensing. *In Proceedings of the 9th International Symposium on Experimental Robotics (ISER). Singapore, 2004.*

Acknowledgements

When I had the opportunity to come to Sweden to work with unmanned helicopters I realized that, what I had been doing as hobby for the past twenty years was going to become my work. I did not have to think too long about packing my clothes and moving to Sweden to begin this experience. I have to thank my family which has supported me in taking such an important decision and I have to say that their role has been essential throughout the several peaks and valleys which one inevitably encounters during one's PhD studies. A special thanks goes to Roberta, the sunshine of my life, who can make the sun appear even in the middle of the night.

I also would like to express my gratitude to:

my supervisor Patrick Doherty for believing in my potential and giving me the opportunity to work in such a stimulating workplace;

Simone and Torsten for sharing their experience and talent with me and for the never ending, challenging discussions;

Mariusz and Piotr for always being helpful and sharing their skills in addition to the daily fun of working with UAVs;

Björn, David, Fredrik (thanks for checking over errors in the thesis), Jonas, Karol, Lukasz, Maria, Martin, Per-Magnus, Per Nyblom, Piero, Robert and Tommy for making the workplace always friendly and enjoyable;

finally, my flatmates throughout the years and all my friends!

Contents

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is an aerial vehicle without a human pilot on board. It can be autonomous, semi-autonomous or radio-controlled. In the past, the use of UAVs have been mostly related to military applications in order to perform the so-called Dirty, Dull and Dangerous (D3) missions such as reconnaissance, surveillance and location acquisition of enemy targets. Recently, interest for UAV systems has begun to grow also in the direction of civil applications.

The different aspects of aircraft navigation involve capabilities in decision making, obstacle perception, aircraft state estimation (estimation of position, velocity and attitude) and aircraft control. In the earlier days of aeronautic history, the on-board pilot had to solve these tasks by using his own skills. Nowadays the situation is quite different since a high level of automation is present in modern military and civil aircrafts. However, the replacement of the pilot skills with a fully automated system is an extremely hard task. This is the reasons why the introduction of UAVs in non-segregated airspace represents a challenge.

The work presented in this thesis was initiated as part of the WITAS UAV Project [18, 17], where the main goal was to develop technologies and functionalities necessary for the successful deployment of a fully autonomous Vertical Take Off and Landing (VTOL) UAV. The typical operational environment for this research has been urban areas where an

autonomous helicopter can be deployed for missions such as traffic monitoring, photogrammetry, surveillance, etc.

To accomplish complex autonomous missions, *high-level* functionalities such as mission planning and real world scene understanding have to be integrated with *low-level* functionalities such as motion control, sensing and control mode coordination. Details and discussions relative to the WITAS UAV software architecture can be found in [19, 42].

This thesis addresses two problems related to the navigation task, therefore it is divided into two parts. Part I addresses the problem of *Simulation and Guidance* where a simulation tool and a spline following control mode are described. Part II addresses the problem of *Vision-based Navigation* without GPS. Moreover, a vision-based method for ground target geo-location from a Micro Aerial Vehicle platforms (MAV) is also presented.

Part I describes the helicopter dynamic model used in the simulator and the development and flight-testing of a Path Following Control Mode (PFCM) which enables a UAV helicopter to follow 3D geometric paths. A basic functionality required for a UAV is the ability to fly from a starting location to a goal location. In order to achieve such a task safely the helicopter must perceive and than avoid eventual obstacles on its way. Additionally, it must stay within the allowed flight envelope. The UAV developed during the WITAS UAV Project has the capability of using a priori knowledge of the environment in order to avoid static obstacles. An on-board *path planner* uses this knowledge to generate collision-free paths. Details of the path planning methods used can be found in [48, 47, 64]. The PFCM developed in this thesis executes 3D path segments generated by the path planner. The velocity set-points are generated in the PFCM taking into account the helicopter's kinematic constraints. The PFCM has been implemented on-board an autonomous helicopter and flight-test results are presented in this thesis.

In Part II the challenging problem of coping with long-term GPS outages using vision-based techniques is addressed. Such a problem is of extreme importance when deploying a UAV in real world scenarios. A UAV navigation system in fact depends on the GPS system as the primary position source. It is well known that the GPS system can be vulnerable in certain situations and can be jammed using commercial GPS jammers. Such a critical issue must be addressed and a back-up solution must be

found for a safe deployment of any UAV system. Part II of this thesis deals with this issue and proposes a vision-based navigation architecture which can cope with GPS outages using geo-referenced imagery as absolute position source. The proposed vision-based solution is implemented on-board a UAV helicopter and experimental flight-test results will be presented. In addition the thesis presents experimental results of a vision-based autonomous landing approach using an artificial pattern as landing pad.

Geo-referenced imagery is also used for high accuracy ground target localization from a small autonomous fixed-wing MAV platform. The accuracy of such a method is evaluated through field trials. The approach was also used during the 3rd US-European Micro Aerial Vehicle competition (MAV07, Toulouse, 2007) achieving the best accuracy among the competitors resulting in 1st place in the competition.

The UAV platform used for most of the field experiments is a Rmax unmanned helicopter manufactured by Yamaha Motor Company. The hardware and software necessary for autonomous flight have been developed during the WITAS UAV Project using *off-the-shelf* hardware components.

Flight missions are performed in a training area of approximately one square kilometer in the south of Sweden, called Revinge (Figure 1.1). The area is used for emergency service training and includes a number of building structures and road networks. An accurate 3D model of the area is available on board the helicopter in a GIS (Geographic Information System), which is used for mission planning purposes.

The image in Figure 1.1 is also used for helicopter localization purposes as a geo-reference image in chapter 5.

Figure 1.1: Orthophoto of the flight-test area in Revinge (south of Sweden).

Chapter 2

Platform

This chapter presents an overview of the UAV hardware/software architecture developed during the WITAS UAV Project. A description of the UAV helicopter platform used for the experimental tests will also be presented.

2.1 Hardware/software architecture

The software architecture developed is a complex distributed architecture for high level autonomous missions. The architecture enables the UAV to perform so-called *push button missions*. This is intended to cover missions where a UAV is capable of planning and executing the mission from take-off to landing with limited, or no human intervention. An example of such a mission demonstrated in an actual flight experiment is a building inspection mission. In this scenario a UAV helicopter is given the task of taking pictures of each of the facades of a selected set of buildings input by a ground operator using an on-screen display. Once the ground operator has selected the buildings of interest and given the start mission command, a flight-plan is generated on-board the helicopter from take-off to landing including the computation of the most advantageous helicopter position relative to the building facades to inspect [63].

To accomplish high-level autonomous missions, an architecture which

integrates many different functionalities is required. Figure 2.1 provides an overview of the software architecture that has been developed during the WITAS Project. The three main blocks of Figure 2.1 represent:

- the deliberative/reactive system (DRC) which implements a number of high level planning and monitoring functionalities such as path planner, task planner, execution monitoring, GIS, etc.
- the image processing system (IPC) which implements the image processing functions, part of the sensor fusion architecture described in chapter 5 and handles everything which is related to the video camera (frame grabbing, camera pan/tilt control, etc.)
- the primary flight control system (PFC) which implements the control modes (hovering, path following, take-off, landing, etc.), the Kalman filters (INS/GPS, INS/camera) and handles communication with the helicopter platform and with the other sensors (GPS, pressure sensor, etc.)

each system is implemented on a separate computer.

The PFC executes predominantly hard real-time tasks such as flight control modes or the sensor fusion algorithms. This part of the system uses a Real-Time Application Interface (RTAI) [39] which provides industrial-grade real-time operating system functionality. RTAI is a hard real-time extension to a standard Linux kernel (Debian) and has been developed at the Department of Aerospace Engineering of Politecnico di Milano.

The DRC instead, has reduced timing requirements. This part of the system uses the Common Object Request Broker Architecture (CORBA) as its distribution backbone. Currently an open source implementation of CORBA 2.6 called TAO/ACE [31] is in use. More details about the complete software architecture can be found in [19, 63, 42].

The on-board hardware schematic is displayed in Figure 2.2. The avionic system is based on three PC104 embedded computers. The PFC system is implemented on a 700Mhz Pentium III and includes a wireless Ethernet bridge, a GPS receiver, a barometric altitude sensor and a compass. The PFC communicates with the Yamaha Rmax helicopter through a serial line RS232C. It sends control inputs to the servos and reads the

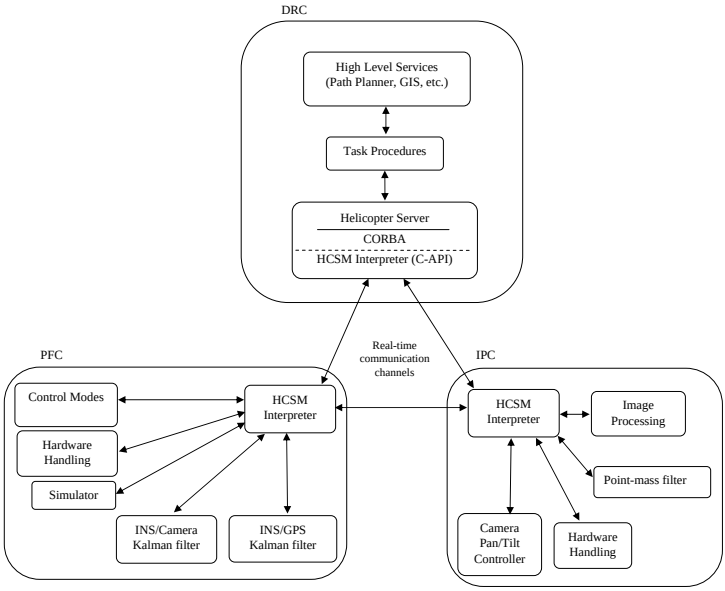


Figure 2.1: UAV software architecture schematic.

inertial sensor data from an inertial measurement unit integrated in the helicopter. The IPC runs on a second PC104 embedded computer (PIII 700MHz), it includes a color CCD camera mounted on a pan/tilt unit, a video transmitter and a video recorder (miniDV). The DRC system runs on the third PC104 embedded computer (Pentium-M 1.4GHz) and executes planning and monitoring functionalities. Network communication between computers is physically realized with serial line RS232C and Ethernet. Ethernet is mainly used for CORBA applications, remote login and file transfer, while serial lines are used for hard real-time networking.

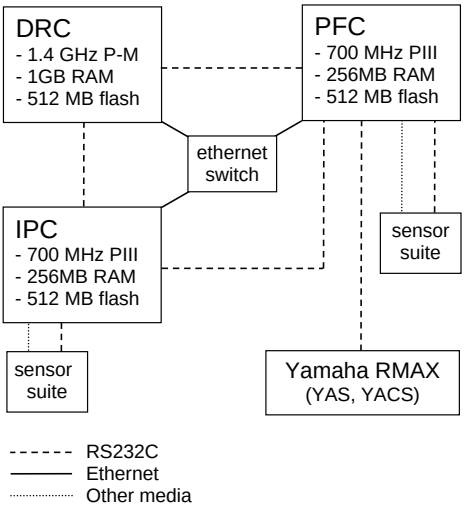


Figure 2.2: On-board hardware schematic.

2.2 Helicopter platform

The Yamaha Rmax helicopter used for the experimentation (Figure 2.3) has a total length of 3.6 m (including main rotor). It is powered by a 21 hp two-stroke engine and has a maximum takeoff weight of 95 kg.

The Rmax rotor head is equipped with a Bell-Hiller stabilizer bar (see Figure 2.4). The effect of the Bell-Hiller mechanism is to reduce the response of the helicopter to wind gusts. Moreover, the stabilizer bar is used to generate a control augmentation to the main rotor cyclic input. The Bell-Hiller mechanism is very common in small-scale helicopters but quite uncommon in full-scale helicopters. The reason for this is that a small-scale helicopter experiences less rotor-induced damping compared to full-scale helicopters. Consequently, it is more difficult to control for a human pilot. It should be pointed out though that an electronic control system can stabilize a small-scale helicopter without a stabilizer bar quite

Figure 2.3: The Rmax helicopter.

efficiently. For this reason the trend for small-scale autonomous helicopters is to remove the stabilizer bar and let the digital control system stabilize the helicopter. The advantage in this case is a reduced mechanical complexity in the system.

Figure 2.4: The Rmax rotor head with Bell-Hiller mechanism.

The Rmax helicopter has a built-in attitude sensor called YAS (Yamaha Attitude Sensor) composed of three accelerometers and three rate gyros. The output of the YAS are acceleration and angular rate on the three helicopter body axes (see section 3.4 for a definition of body axis). The YAS also computes the helicopter attitude angles. Acceleration and angular rate from the YAS will be used as inertial measurement data for the sensor fusion algorithms described in this thesis.

The Rmax also has a built-in digital attitude control system called YACS (Yamaha Attitude Control System). The YACS stabilizes the helicopter attitude dynamics and the vertical channel dynamics. The YACS is used in all the helicopter control modes implemented in the control architecture as an attitude stabilization system.

Part I

Simulation and Guidance

Chapter 3

Simulation

3.1 Introduction

This chapter describes the simulation tool which has been used to develop and test the guidance algorithms implemented on the Rmax helicopter. It is used to test the helicopter missions both in the lab and on the field. The Rmax helicopter simulator is implemented in the C language and allows testing of all the control modes developed. Many flight-test hours have been avoided due to the possibility of running, in simulation, the exact code of the control system which is executed on the on-board computer during the actual flight-test.

In order to develop and test the helicopter control system a mathematical model which represents the dynamic behavior of the helicopter is required. The simulator described in this section is specialized for the Yamaha Rmax helicopter in the sense that the model includes the dynamics of the bare platform in addition to the Yamaha Attitude Control System (YACS). The YACS system stabilizes the pitch and roll angles of the helicopter, the yaw rate and the vertical velocity.

The dynamics model of the Rmax helicopter has been derived in two separate steps. First, the helicopter attitude dynamics with the YACS in the loop, has been identified using system identification methods. Sub-

sequently, the derived attitude dynamics has been used as input to the longitudinal equations of motion representing the helicopter velocity dynamics.

The reason why the YACS has been used was to speed up the control development process and to shift the focus toward development of functionalities such as the PFCM presented in chapter 4. All of the currently developed flight modes use the YACS as an inner control loop which stabilizes the high frequency helicopter dynamics. Experimental tests have shown that the YACS decouples the helicopter dynamics so that the pitch, roll, yaw and vertical channels can be treated separately in the control system design. In other words the channels do not influence each other.

3.2 Related work

The derivation of the helicopter model described here is quite unusual since it includes the attitude control system in the loop. The main interest here was to investigate the suitability of using the YACS as the inner control loop of a complete helicopter control system.

Typical work on scale helicopter modeling found in the literature address the problem of modeling the helicopter dynamics without a control system in the loop. Extensive insight on helicopter dynamics can be found in [50]. In [44], a model identification technique has been used to identify the dynamics of the bare unmanned helicopter R50 also manufactured by the Yamaha Motor Company. In that work, a mathematical structure of the model of the helicopter dynamics was assumed beforehand, while the parameters of the model were identified through a parameter identification technique.

3.3 Hardware-in-the-loop simulation

Two versions of the simulator have been developed. A non real-time version, used to develop the control modes, where the purpose of the simulation in this case is the tuning and testing of the internal logic of the control modes. A real-time version is instead used to test the complete UAV archi-

tecture, where simulations are performed with *hardware-in-the-loop*. The latter is used to test a complete helicopter mission and can be used on the flight-test field as a last minute verification for the correct functioning of hardware and software. Both simulators use the same dynamic model which will be presented in this chapter.

The helicopter dynamics function is located in the PFC system and it is called every 20 ms when the system is in simulation mode. Figure 3.1 shows the hardware/software components involved in a real flight-test and in the simulation test. The components and connections represented with a dashed line are not active during the respective test modalities.

The diagram shows which components can be tested in simulation. Observe that in simulation the helicopter servos are connected and can move, but there is no feedback from the servo position to the simulator. The fact that their movement can be visually checked is used as a final check that the system is operating appropriately.

The YACS control system is also part of the loop, but since it is built into the helicopter, it cannot be fed with simulated sensor outputs, so it still takes the input from the YAS sensor which, obviously, does not deliver any measurement from the helicopter. This is not problematic because the simulator does not have the purpose of testing the correct functioning of the YACS.

Currently, the video camera is not used in the simulation loop but the system can still control the camera pan/tilt. A virtual environment (Figure 3.2), reproducing the flight-test area described in the introduction (Figure 1.1), is used for visualization purposes. Theoretically the image processing functions could be fed with synthetic images in order to feedback from the virtual environment, this is a topic for future work.

3.4 Reference systems

This section provides definitions of the different reference systems used in this part of the thesis. The reference systems and transformations relative to the camera system will be given in Part II of the thesis.

The *Earth system* (Figure 3.3) has its origin at the center of mass of the Earth and axes which are fixed with respect to the Earth. Its X^e axis

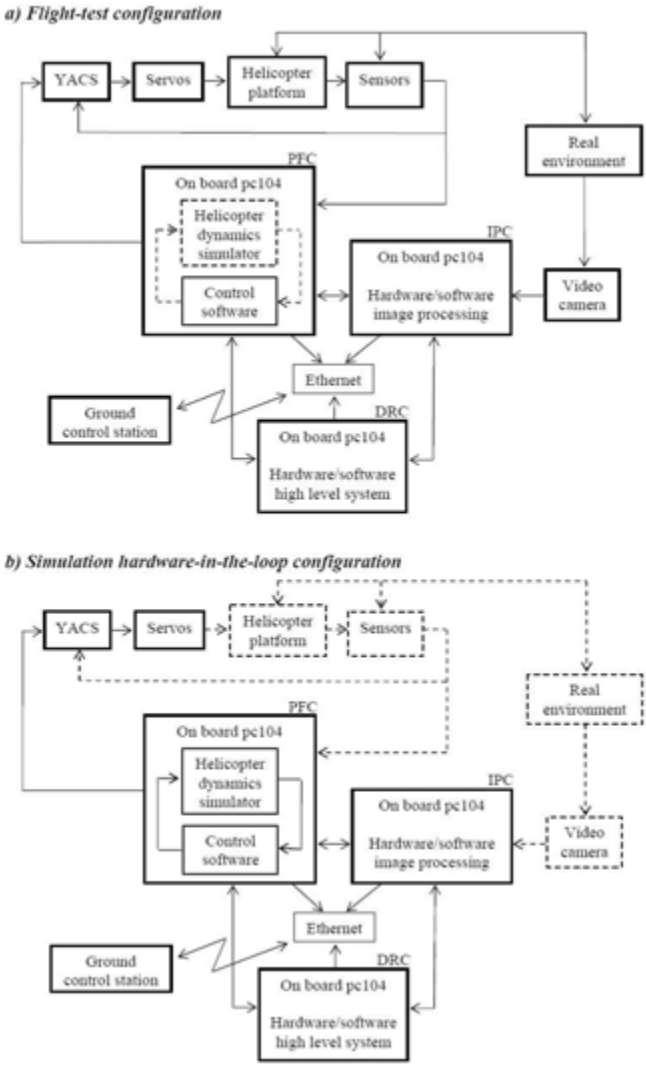


Figure 3.1: Figure a) depicts the hardware/software architecture in flight-test configuration. Figure b) depicts the architecture during the hardware-in-the-loop simulation.

Figure 3.2: Virtual environment used for flight mission simulation.

points toward the mean meridian of Greenwich, the Z^e axis is parallel to the mean spin axis of the Earth, and the Y^e axis completes a right-handed orthogonal frame.

The *navigation system* (Figure 3.3) is a local geodetic frame which has its origin coinciding with that of the sensor frame and axes with the X^n axis pointing toward the geodetic north, the Z^n axis orthogonal to the reference ellipsoid pointing down, and the Y^n axis completing a right-handed orthogonal frame.

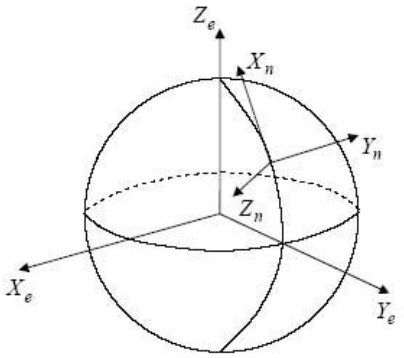


Figure 3.3: Earth and navigation reference systems.

The *body system* (Figure 3.4) is an orthogonal axis set which has its origin coinciding with the center of gravity of the helicopter, the X^b axis pointing forward to the helicopter's nose, the Y^b axis orthogonal to the X^b axis and pointing to the right side of the helicopter body, and the Z^b axis pointing down so that it forms a right-handed orthogonal frame. Since the Rmax inertial sensors are quite close to the helicopter's center of gravity it is possible to consider the navigation frame and the body frame as having the same origin point.

In order to transform a vector from the body system to the navigation

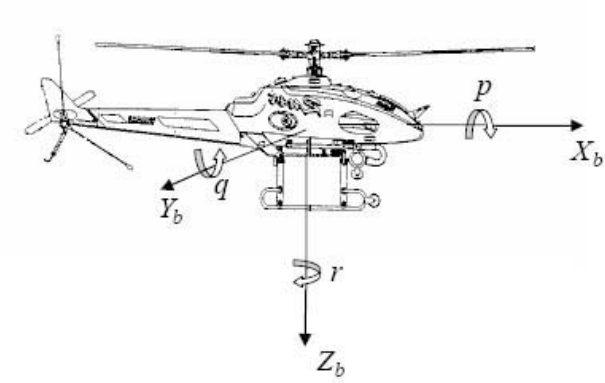


Figure 3.4: Body reference system.

system a rotation matrix has to be applied:

$$\vec{r}^{\,n} = C_b^n \vec{r}^{\,b} \tag{3.1}$$

The rotation matrix C_b^n is defined as:

$$C_b^n = \begin{bmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\phi cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{3.2}$$

where ϕ, θ and ψ are the Euler angles roll, pitch and heading. Since the rotation matrix is orthogonal, the transformation from the navigation system to the body system is given by:

$$C_n^b = (C_b^n)^T \tag{3.3}$$

The transformation matrix 3.2 and 3.3 will be used later in the thesis.

3.5 The augmented Rmax dynamic model

The Rmax helicopter model presented in this thesis includes the bare helicopter dynamics and the YACS control system dynamics. The code of the

YACS is strictly Yamaha proprietary so the approach used to build the relative mathematical model has been that of *black-box* model identification. With the use of this technique, it is possible to estimate the mathematical model of an unknown system (the only hypothesis about the system is that it has a linear behavior) just by observing its behavior. This is achieved in practice by sending an input signal to the system and measuring its output. Once the input and output signals are known there are several methods to identify the mathematical structure of the system.

The YACS model identification is not part of this thesis and details can be found in [20]. In the following section the transfer functions of the augmented attitude dynamics will be given. These transfer functions will be used to build the augmented Rmax helicopter dynamic model used in the simulator.

3.5.1 Augmented helicopter attitude dynamics

As previously stated the YACS and helicopter attitude dynamics have been identified through black-box model identification.

The Equations in 3.4 represent the four input/output transfer functions in the Laplace domain:

$$\begin{aligned} \Delta\phi &= \frac{2.3(s^2+3.87s+53.3)}{(s^2+6.29s+16.2)(s^2+8.97s+168)}\Delta ail \\ \Delta\theta &= \frac{0.5(s^2+9.76s+75.5)}{(s^2+3s+5.55)(s^2+2.06s+123.5)}\Delta ele \\ \Delta r &= \frac{9.7(s+12.25)}{(s+4.17)(s^2+3.5s+213.4)}\Delta rud \\ \Delta a_Z &= \frac{0.0828s(s+3.37)}{(s+0.95)(s^2+13.1s+214.1)}\Delta thr \end{aligned} \tag{3.4}$$

where $\Delta\phi$ and $\Delta\theta$ are the roll and pitch angle increments (deg), Δr the body yaw angular rate increment (deg/sec) and Δa_Z the acceleration increment g along the Z^b body axis. Δail, Δele, Δrud and Δthr are the control input increments taken relative to a trimmed flight condition. The control inputs are in YACS units and range from -500 and +500.

These transfer functions describe not only the dynamic behavior of the YACS but also the dynamics of the helicopter (rotor and body) and the dynamics of the actuators.

Since the chain composed by the YACS, actuators and helicopter is quite complex it is important to remember that the estimated model has only picked up a simple reflection of the system behavior. The identification was done near the hovering condition so it is improper to use the model for different flight conditions. In spite of this, simulations up to ∼10 m/s have shown good agreement with the experimental flight-test results.

3.5.2 Helicopter equations of motion

The aircraft equations of motion can be expressed in the body reference frame with three sets of first order differential equations [24]. The first set represents the translational dynamics along the three body axes:

$$\begin{aligned} X &= m(\dot{u} + q\,w - r\,v) + m\,g\,sin\theta \\ Y &= m(\dot{v} + r\,u - p\,w) - m\,g\,cos\theta\,sin\phi \\ Z &= m(\dot{w} + p\,v - q\,u) - m\,g\,cos\theta\,cos\phi \end{aligned} \tag{3.5}$$

where X, Y, Z represent the resultants of all the aerodynamic forces; u, v, w the body velocity components; p, q, r the body angular rates; m and g the mass and the gravity acceleration; ϕ and θ the pitch and roll angles.

The second set of equations represents the aircraft rotational dynamics:

$$\begin{aligned} L &= I_x\dot{p} - (I_y - I_z)\,q\,r \\ M &= I_y\dot{q} - (I_z - I_x)\,r\,p \\ N &= I_z\dot{r} - (I_x - I_y)\,p\,q \end{aligned} \tag{3.6}$$

where L, M, N represent the moments generated by the aerodynamic forces acting on the helicopter; I_x, I_y, I_z the inertia moments of the helicopter.

The third set of equations represents the relation between the body angular rates and the Euler angles:

$$\begin{aligned} \dot{\phi} &= p + q\,sin\phi\,tan\theta + r\,cos\phi\,tan\theta \\ \dot{\theta} &= q\,cos\phi - r\,sin\phi \\ \dot{\psi} &= q\,sin\phi\,sec\theta + r\,cos\phi\,sec\theta \end{aligned} \tag{3.7}$$

These three sets of nonlinear equations are valid for a generic aircraft. The transfer functions Equations 3.4 can be used now in the motion equations. From the Laplace domain of the transfer functions, it is possible to pass to the time domain. This means that from the first three equations in 3.4 we derive $\phi(t)$, $\theta(t)$ and $r(t)$ which can be used in Equations 3.7 in order to find the other parameters $p(t)$, $q(t)$, $\psi(t)$.

The Equations in 3.6 will not be used in the model because the dynamics represented by these equations is contained in the first three transfer functions in 3.4. The motion Equations in 3.5 can be rewritten as follows:

$$\begin{aligned} \dot{u} &= F_x - q\,w + r\,v - g\,sin\theta \\ \dot{v} &= F_y - r\,u + p\,w + g\,cos\theta\,sin\phi \\ \dot{w} &= F_z - p\,v + q\,u + g\,cos\theta\,cos\phi \end{aligned} \tag{3.8}$$

where F_x, F_y, F_z are the forces per unit of mass. In this set of equations some of the nonlinear terms are small and can be neglected for our flight envelope, although for simulation purposes, it does not hurt to leave them there. Later, when the model will be used for control purposes the necessary simplifications will be made.

The tail rotor force is included in F_y and it is balanced by a certain amount of roll angle. In fact every helicopter with a tail rotor must fly with a few degrees of roll angle in order to compensate for the tail rotor force which is directed sideways. For the Rmax helicopter the roll angle is 4.5 deg in hovering condition with no wind. The yaw dynamics in our case is represented by the third transfer function in 3.4. For this reason we do not have to model the force explicitly. By doing that we find in our model a zero degree roll angle in hovering condition which does not affect the dynamics of our simulator. Of course a small coupling effect between

the lateral helicopter motion and yaw channel is neglected during a fast yaw maneuver due to the consistent increase of the tail rotor force.

The Equations in 3.8 are then rewritten as follows:

$$\begin{aligned} \dot{u} &= X_u\, u - q\, w + r\, v - g\, sin\theta \\ \dot{v} &= Y_v\, v - r\, u + p\, w + g\, cos\theta\, sin\phi \\ \dot{w} &= Z_w\, w + T - p\, v + q\, u + g\, cos\theta\, cos\phi \end{aligned} \tag{3.9}$$

where X_u, Y_v and Z_w are the aerodynamic derivatives (accounting for the aerodynamic drag). The values used for the aerodynamic derivatives are $X_u = -0.025$, $Y_v = -0.1$ and $Z_w = -0.6$. These values have been chosen using an empirical best fit criteria using flight-test data.

The main rotor thrust T is given by:

$$T = -g - \Delta a_Z \tag{3.10}$$

where Δa_Z is given by the fourth transfer function in 3.4.

Comparing the Equations in 3.9 with other work in the literature, for example [44], it can be observed that the rotor flapping terms are missing. The rotor flapping represents the possibility of the helicopter rotor disk to tilt relative to the helicopter fuselage. On the other hand the flapping dynamics is contained in the transfer functions in 3.4. It was not possible to model rotor flapping explicitly in the Equations in 3.9 because it is not observable from the black-box model identification approach used. The fact that the flapping terms are not included in Equations 3.9 does not have strong consequences on the low frequency dynamics. On the other hand, the high frequency helicopter dynamics is not captured correctly. Therefore the helicopter model derived here cannot be used for high bandwidth control system design. In any case, the model is good enough for position and velocity control loop design. In [50] the mathematical formulation for rotor flapping can be found.

3.6 Simulation results

The validation of the Rmax helicopter model can be found in [20]. In this section, the validation of the simulation procedure with the control system in the loop will be described. The results of two simulation tests of the PFCM, which will also be described later in the thesis, will be provided. The PFCM is a control function that enables the helicopter to follow 3D path segments. The path segments will be given to the PFCM which is in a closed loop with the simulator. The simulation results will be compared to the same PFCM implemented on the helicopter platform. This is not a validation of the simulator since the control function is in the loop. What is interesting is the analysis of the differences of the closed loop behavior between the flight-test and the simulation. The helicopter is commanded to fly a segment path starting from a hovering condition until reaching a target speed. The path also presents a curvature change. The same control function has been used for the two tests. In Figure 3.5 and Figure 3.6 the simulation (dashed line) and flight-test (continuous line) are overlapped on the same graph. The upper-left plot of Figure 3.5 represents the helicopter path as seen from above. The difference in position between the simulation and flight-test is very small, below one meter and cannot be seen in detail from the plot. In the same diagram the velocity components and the total velocity are plotted. This shows that the simulated velocity and the real velocity are quite close. In this test, the helicopter was accelerated to 5 m/s forward speed. In Figure 3.6 the results for the pitch and roll inputs and the attitude angles are presented. The pitch and roll inputs present a steady state error compared to the simulation while the pitch and roll angles are in good agreement.

In Figures 3.7 and 3.8, the same path segment is tested but the helicopter accelerates until 10 m/s velocity is reached. The simulated position and velocity are still in good agreement with the flight-test experiment while the pitch and roll angles are worse. This is the limitation of assuming a linear dynamic model. Since the model has been identified near hovering conditions, when operating far from such conditions the model will not perform equally well. It is interesting to notice that the simulation can predict quite accurately the saturation of the roll input (Figure 3.8) at around 845 sec. This could be a sign that the helicopter is flying too

fast for the current path curvature, and an adjustment in the PFCM in the generation of the velocity reference might be required. This kind of problem can be analyzed very efficiently with the simulator tool developed.

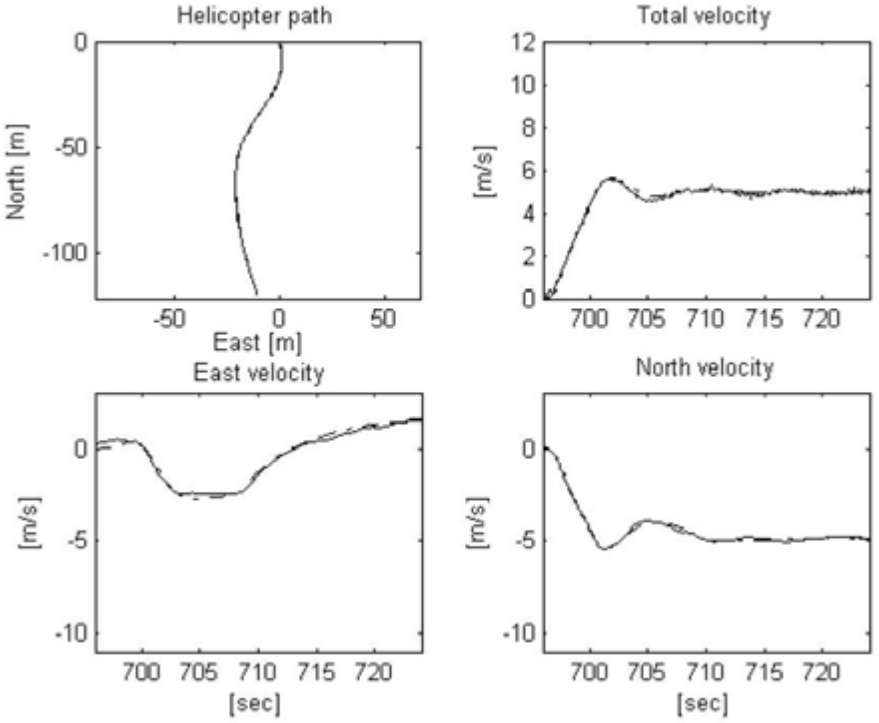


Figure 3.5: **Results 1st flight.** Comparison between flight-test position and velocity data (solid) with simulation data (dash-dot). The helicopter accelerates to 5 m/s target velocity.

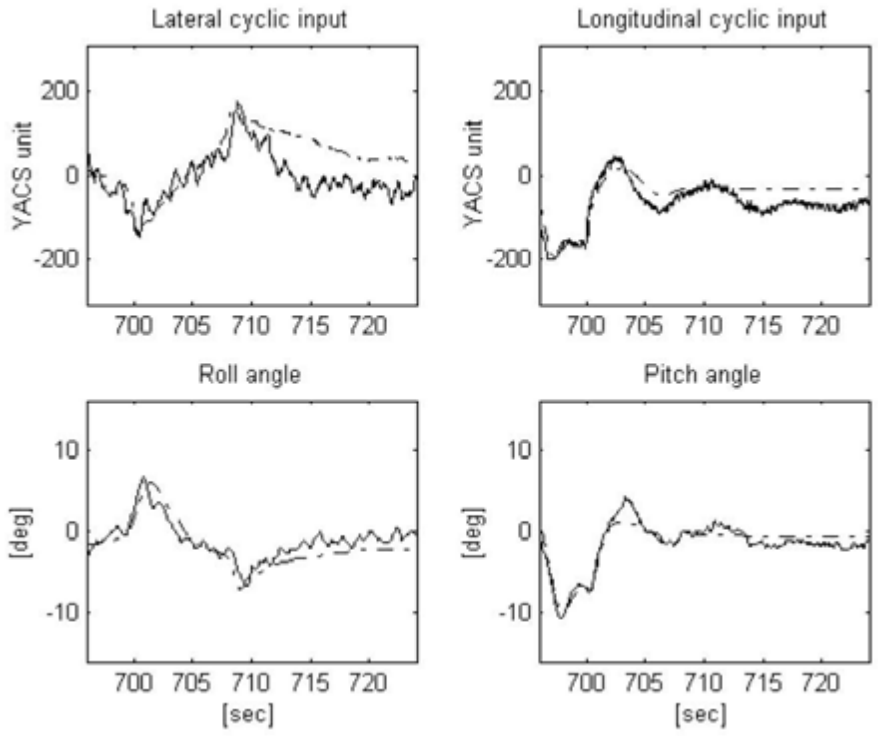


Figure 3.6: **Results 1st flight.** Comparison between flight-test helicopter inputs and attitude data (solid) with simulation data (dash-dot). The helicopter accelerates to 5 m/s target velocity.

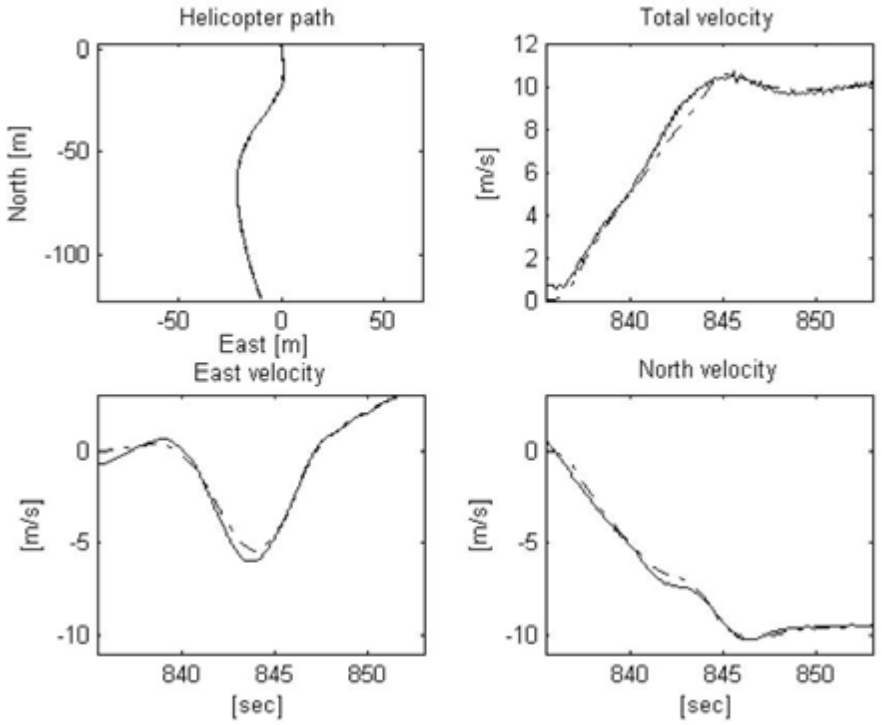


Figure 3.7: **Results 2nd flight.** Comparison between flight-test position and velocity data (solid) with simulation data (dash-dot). The helicopter accelerates to 10 m/s target velocity.

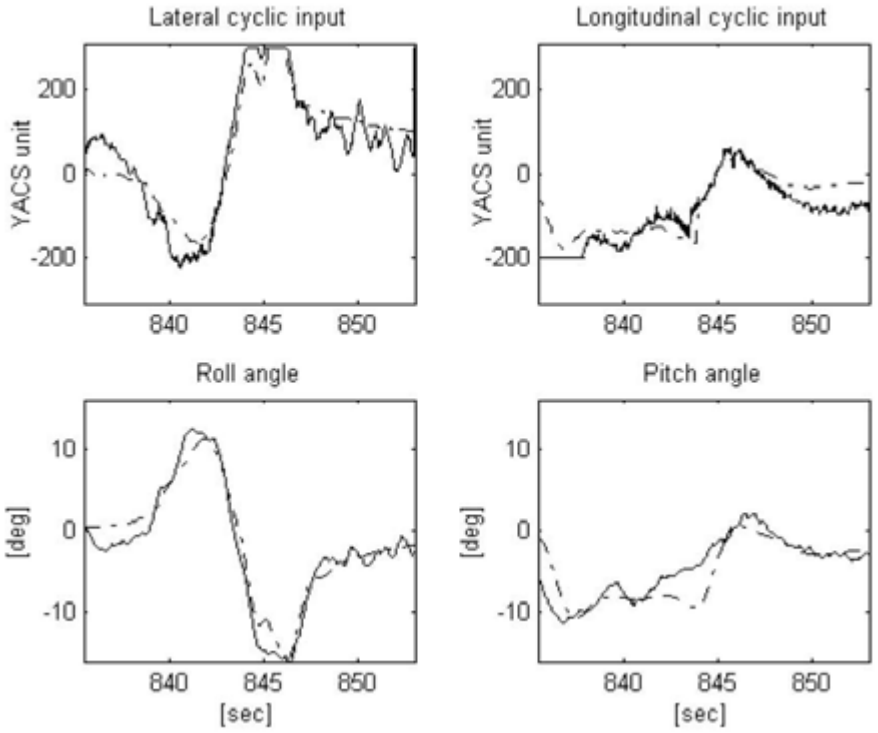


Figure 3.8: **Results 2nd flight.** Comparison between flight-test helicopter inputs and attitude data (solid) with simulation data (dash-dot). The helicopter accelerates to 10 m/s target velocity.

3.7 Conclusion

In this chapter, the simulation tool used to test and develop the UAV software architecture, including the control system has been described. The simulator has been a useful tool for the development of the control modes. The hardware-in-the-loop version of the simulator is a useful tool to test a complete mission in the field. In addition, the fact that the helicopter servos are in the simulation loop provides a rapid verification that the correct signals arrive from the control system. This verification is used for a final decision on the field in order to proceed or not with a flight-test.

Chapter 4

Path Following Control Mode

4.1 Introduction

The Path Following Control Mode (PFCM) described in this chapter and published in [14], has been designed to navigate an autonomous helicopter in an area cluttered with obstacles, such as an urban environment. The path planning problem is not addressed in this thesis, it can be found in [47]. It is assumed that a path planner generates a collision-free path. Then the task which will be solved here is to find a suitable guidance and control law which enables the helicopter to follow the path robustly. The path planner calculates the path geometry which is then input to the PFCM. Before starting with the description of the PFCM, some basic terminology will be provided.

Guidance is the process of directing the movements of an aeronautical vehicle with particular reference to the selection of a flight path. The term of guidance or *trajectory generation* in this thesis addresses the problem of generating the desired reference position and velocity for the helicopter at each control cycle.

The *outer control* is a feedback loop which takes as inputs the reference

position and velocity generated by the trajectory generator and calculates the output for an inner control loop.

The *inner control* is a feedback loop which stabilizes the helicopter attitude and the vertical dynamics. As mentioned previously, the inner control loop used here has been developed by Yamaha Motor Company and is part of the YACS.

4.2 Related work

Several methods have been proposed to solve the problem of generation and execution of a state-space trajectory for an autonomous helicopter [23, 27]. In general this is a hard problem, especially when the trajectory is time dependent. The solution adopted here is to separate the problem into two parts: first to find a collision-free path in the space domain and than to add a velocity profile later. In this way the position of the helicopter is not time dependent which means that it is not required for the helicopter to be in a certain point at a specific time. A convenient approach for such a problem is the path following method. By using the path following method the helicopter is forced to fly close to the geometric path with a specified forward speed. In other words, the path is always prioritized and this is a requirement for robots that for example have to follow roads and avoid collisions with buildings. The method developed for PFCM is weakly model dependent and computationally efficient.

The path following method has also been analyzed in [58, 53]. The guidance law derived there presents singularity when the cross-track error is equal to the curvature radius of the path so that it has a restriction on the allowable cross-track error. The singularity arises because the guidance law is obtained by using the Serret-Frenet formulas for curves in the plane [58].

The approach used in this thesis does not use the Serret-Frenet formulas but a different guidance algorithm similar to the one described in [22] which is also known as virtual leader. In this case the motion of the control point on the desired path is governed by a differential equation containing error feedback which gives great robustness to the guidance method. The control point (Figure 4.1) is basically a point lying on the path where the helicopter ideally should be.

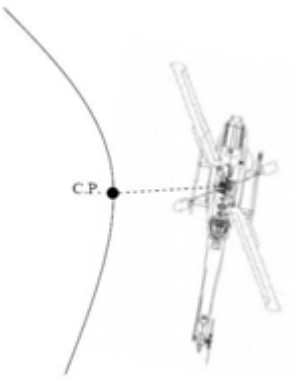


Figure 4.1: Control point on the reference path.

The guidance algorithm developed uses information from the model in 3.9 described in section 3.5.2 in order to improve the path tracking error while maintaining a reasonable flight speed. The experimental results presented show the validity of the control approach.

Figure 4.2 represents the different components of the control system, from the path planner to the inner loop that directly sends the control inputs to the helicopter actuators. The PFCM described in this thesis includes the trajectory generator and the outer control loop.

4.3 Trajectory generator

In this section, a description of the guidance algorithm developed for the PFCM is provided. The trajectory generator function takes as input a set of parameters describing the geometric path calculated by the path planner and calculates the reference position, velocity and heading for the inner control loop. First the analytic expression of the 3D path is calculated, then a feedback algorithm calculates the control point on the path. Finally the reference input for the outer loop is calculated using the kinematic

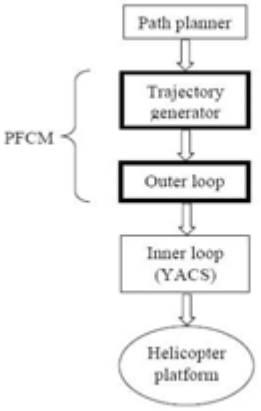


Figure 4.2: The PFCM includes two modules: trajectory generator and outer loop.

helicopter model described in chapter 3.

4.3.1 Calculation of the path geometry

The path planner generates 3D geometric paths described by a sequence of segments. Each segment is passed from the high-level part of the software architecture, where the path planner is located, to the low-level part where the control modes including the PFCM are implemented.

The path segment is generated in the navigation frame with the origin fixed at the initial point of the path. We will use the superscript n to indicate a vector in the navigation frame. Each segment is described by a parameterized 3D cubic curve represented by the following equation in vectorial form:

$$\vec{p}^{\,n}(s) = As^3 + Bs^2 + Cs + D \tag{4.1}$$

where A, B, C and D are 3D vectors calculated from the boundary conditions of the segment and s the path segment parameter.

Figure 4.3 depicts a path segment with the relative boundary conditions. The segment is defined by the starting point coordinates $\vec{p}(0)$, the end point coordinates $\vec{p}(1)$ and two vectors which represent the direction of the segment tangent at the starting point $\dot{\vec{p}}(0)$ and at the end point $\dot{\vec{p}}(1)$ so that the 3D path segment is defined by 12 parameters. The path planner calculates the 12 parameters ensuring the continuity of the path and of the first order derivative at the segment joints.

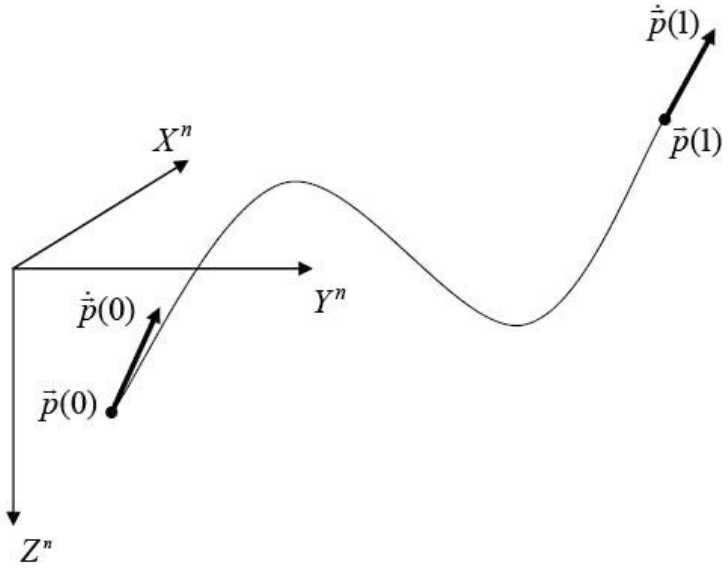


Figure 4.3: Boundary conditions for a path segment.

Actually, by imposing only these boundary conditions (continuity of the path and continuity of the first order derivative at the segment joints) the segment has two degrees of freedom undefined which are represented by

the magnitude of the tangent vectors $\dot{\vec{p}}(0)$ and $\dot{\vec{p}}(1)$.

The magnitude of the tangent vector affects the curvature of the path. If the continuity of the second order derivative at the joints (e.g. the curvature) is imposed then all the 12 parameters would be found and in this case the path would be a cubic spline [51]. The two degrees of freedom are chosen by the path planner in order to satisfy other conditions which will not be mentioned here. For more details the reader is referred to [47]. The path generated in this way can have discontinuity of the second order derivative at the segment joints. This can lead to a small path tracking error especially at high speed.

The 12 parameters are passed as input arguments to the PFCM which then generates the reference geometric segment used for control purposes. The generation of the path segment in the PFCM is done using the matrix formulation in 4.2 with the boundary condition vector explicitly written on the right hand side. The parameter s ranges from $s = 0$ which corresponds to the starting point $\vec{p}(0)$ to $s = 1$ which corresponds to the end point $\vec{p}(1)$ of the same segment. When the helicopter enters the next segment the parameter is reset to zero.

$$\vec{p}^{\,n}(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}(0) \\ \vec{p}(1) \\ \dot{\vec{p}}(0) \\ \dot{\vec{p}}(1) \end{bmatrix} \tag{4.2}$$

The vectors $\vec{p}(0)$, $\vec{p}(1)$, $\dot{\vec{p}}(0)$ and $\dot{\vec{p}}(1)$ are expressed in the navigation reference system. For control purposes the tangent and the curvature need to be calculated. The path tangent $\vec{t}^{\,n}$ is:

$$\vec{t}^{\,n}(s) = \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}(0) \\ \vec{p}(1) \\ \dot{\vec{p}}(0) \\ \dot{\vec{p}}(1) \end{bmatrix} \tag{4.3}$$

The path curvature $\vec{k}^{\,n}$ is:

$$\vec{q}^{\,n}(s) = \begin{bmatrix} 6s & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}(0) \\ \vec{p}(1) \\ \dot{\vec{p}}(0) \\ \dot{\vec{p}}(1) \end{bmatrix} \tag{4.4}$$

$$\vec{k}^{\,n}(s) = \frac{\vec{t}^{\,n}(s) \times \vec{q}^{\,n}(s) \times \vec{t}^{\,n}(s)}{|\vec{t}^{\,n}(s)|^4} \tag{4.5}$$

In the guidance law the curvature radius which is $\vec{r}^{\,n} = 1/\vec{k}^{\,n}$ will be used. Since $\vec{t}^{\,n}$ and $\vec{r}^{\,n}$ are expressed in the navigation frame, in order to be used in the guidance law, they have to be transformed into the body frame using the rotation matrix C_n^b defined in section 3.4.

At this point the geometric parameters (tangent and curvature) of the path segment are known. Now these parameters can be used in the guidance law provided that the path segment parameter s is known. The method as to how to find s will be discussed in the next section.

4.3.2 Feedback method

When the dynamic model of the helicopter is known, it is in principle possible to calculate beforehand at what point in the path the helicopter should be at a certain time. By doing this the path segment would be time dependent. In this way at each control cycle the path parameters (position, velocity and attitude) would be known and they could be used directly for control purposes. Then the helicopter could be accelerated or slowed down if it is behind or ahead of the actual control point (which is the point of the path where the helicopter should be at the relative time).

The generation of a time dependent trajectory is usually a complex problem. An additional complication is that the trajectory has to satisfy obstacle constraints (to find a collision free path in a cluttered environment).

The alternative approach used here is the following. Instead of accelerating or slowing down the helicopter, the control point will be accelerated or slowed down using a feedback method. In this way the path is not time

dependent anymore and so the problem of generating a collision free path can be treated separately from the helicopter dynamics. Of course the path generated must be smooth enough to be flown with a reasonable velocity and this has to be taken into account at the path planning level. The fact that the helicopter kinematic and dynamic constraints are not taken into account at the path planning level might lead to a path which forces the helicopter to abrupt brake due to fast curvature change. This problem can be attenuated using some simple rules in the calculation of the segment boundary conditions [47].

The algorithm implemented in this thesis finds the control point by searching for the closest point of the path to the helicopter position. The problem could be solved geometrically simply by computing an orthogonal projection from the helicopter position to the path. The problem which arises in doing this is that there could be multiple solutions. For this reason a method has been adopted which finds the control point incrementally and searches for the orthogonality condition only locally.

The reference point on the nominal path is found by satisfying the geometric condition that the scalar product between the tangent vector and the error vector has to be zero:

$$\vec{e} \cdot \vec{t} = 0 \tag{4.6}$$

where the error vector $\vec{e}$ is the helicopter distance from the candidate control point. The control point error feedback is calculated as follows:

$$e_f = \vec{e} \cdot \vec{t}/|\vec{t}| \tag{4.7}$$

with e_f the magnitude of the error vector projected on the tangent $\vec{t}$. The vector $\vec{e}$ is calculated as follows:

$$\vec{e} = \vec{p}_{cp,n-1} - \vec{p}_{heli,n} \tag{4.8}$$

where $\vec{p}_{cp,n-1}$ is the control point position at the previous control cycle and $\vec{p}_{heli,n}$ is the actual helicopter position. The control point is updated using the differential relation:

$$d\vec{p} = \dot{\vec{p}} \cdot ds \tag{4.9}$$

Equation 4.9 is applied in the discretized form:

$$s_n = s_{n-1} + \frac{e_f}{|\frac{d\vec{p}_{cp}}{ds}|_{n-1}} \tag{4.10}$$

where s_n is the new value of the parameter. Equations 4.7 and 4.10 can also be used iteratively in order to find a more accurate control point position. In this application, it was not necessary. Once the new value of s is known, all the path parameters can be calculated.

4.3.3 Outer loop reference inputs

In this section, the method used to calculate the reference input or set-point for the outer control loop will be described in detail. Before proceeding, a description as to how the PFCM takes into account some of the helicopter kinematic constraints will be provided.

PFCM kinematic constraints

The model in 3.9 will be used to derive the guidance law which enables the helicopter to follow a 3D path.

The sine and cosine can be linearized around $\theta = 0$ and $\phi = 0$ since in our flight condition, the pitch and roll angles are between the interval ± 20 deg. This means that we can approximate the sine of the angle to the angle itself (in radians) and the cosine of the angle to 1. By doing this from the first and second equation in 3.7 it is possible to calculate the body angular rate p and q:

$$\begin{aligned} q &= \dot{\theta} + r\phi \\ p &= \dot{\phi} - r\theta \end{aligned} \tag{4.11}$$

where the product between two or more angles has been neglected because it is small compared to the other terms. Using the same considerations and using the Equations in 4.11 it is possible to rewrite the system in 3.9 in the following form:

$$\begin{aligned} \dot{u} &= X_u u - (\dot{\theta} + r\,\phi)w + r\,v - g\,\theta \\ \dot{v} &= Y_v v - r\,u + (\dot{\phi} - r\,\theta)w + g\,\phi \\ \dot{w} &= Z_w w + T - (\dot{\phi} - r\,\theta)v + (\dot{\theta} + r\,\phi)u + g \end{aligned} \tag{4.12}$$

At this point we can add the condition that the helicopter has to fly with the fuselage aligned to the path (in general this condition is not necessary for a helicopter, it has been adopted here to simplify the calculation). The constraints which describe this flight condition (under the assumption of relatively small pitch and roll angles) are:

$$\begin{aligned} r &= \frac{u}{R_y^b} \\ \dot{\theta} &= \frac{u}{R_z^b} \\ v &= \dot{v} = 0 \\ w &= \dot{w} = 0 \end{aligned} \tag{4.13}$$

where R_y^b and R_z^b are the components of the curvature radius along the body axes Y^b and Z^b, respectively.

The equations in 4.12 can finally be rewritten as:

$$\begin{aligned} r &= \frac{u}{R_y^b} \\ \theta &= \frac{X_u u - \dot{u}}{g} \\ \phi &= \frac{u^2}{g R_y^b} \\ T &= -\frac{u^2}{R_z^b} - \frac{u^4}{g(R_y^b)^2} - g \end{aligned} \tag{4.14}$$

In the right hand side of 4.14 we have the four inputs that can be given as a reference signal to an inner control loop (such as the YACS) which

controls the yaw rate r, the attitude angles ϕ, θ and the rotor thrust T. These inputs only depend on the desired velocity and acceleration u and $\dot{u}$ and the path curvature R_y^b and R_z^b.

If the geometry of the path, which is represented by R_y^b and R_z^b, is then assigned, in principle it is possible to assign a desired velocity and acceleration u and $\dot{u}$ and calculate the four inputs for the inner loop. The problem is that these inputs cannot be assigned arbitrarily because they have to satisfy the constraints of the dynamic system composed by the helicopter plus the inner loop.

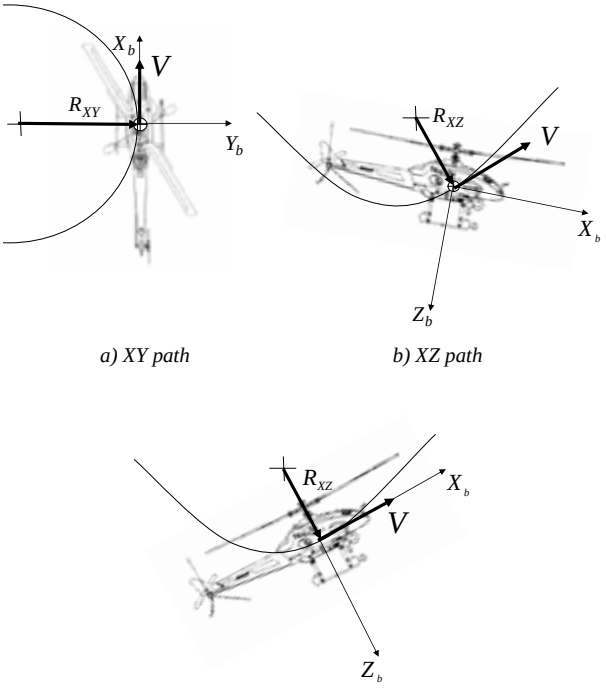
A solution of 4.14 which does not involve the dynamic constraints is represented by a stationary turn on the horizontal plane with constant radius (picture (a) of Figure 4.4). The solution for this flight condition is given by 4.14 where $R_z^b = \infty$, $\dot{u} = 0$ and $R_y^b = constant$. This condition is called *trimmed flight* because the first derivative of the flight parameters are zero ($\dot{\phi} = \dot{\theta} = \dot{r} = \dot{T} = \dot{u} = 0$). For this flight condition it is straightforward to calculate the maximum flight speed allowed. Since the maximum values of r, ϕ, θ and T are limited for safety reasons, the maximum path velocity u can be calculated from the system 4.14:

$$\begin{aligned} u_1 &= |R_y^b r_{max}| \\ u_2 &= |\frac{g\theta_{max}}{X_u}| \\ u_3 &= \sqrt{|\phi_{max} g R_y^b|} \\ u_4 &= (|-g(T_{max}+g)(R_y^b)^2|)^{\frac{1}{4}} \end{aligned} \tag{4.15}$$

The minimum of these four velocities can be taken as the maximum speed for the path:

$$u_{max} = min(u_1, u_2, u_3, u_4) \tag{4.16}$$

The path generated by the path planner is represented by a cubic polynomial. The curvature radius in general is not constant for such a path,



a) XY path *b) XZ path*

c) XZ path with fuselage aligned

Figure 4.4: Representation of the several ways in which the helicopter can follow a 3D path.

which means that the helicopter never flies in trimmed conditions but it flies instead in *maneuvered flight* conditions.

Let now examine a maneuver in the vertical plane ($R_y^b = \infty$, $R_z^b = constant$). From the second equation in 4.13 and the second equation in 4.14 we can observe that there is no constant velocity solution ($\dot{u} = 0$) which satisfies both. This means that when the helicopter climbs, it loses velocity.

To make the PFCM more flexible in the sense of allowing vertical climbing and descending at a constant speed, we have to remove the constraints $\dot{\theta} = u/R_z^b$ and $w = \dot{w} = 0$. In other words the fuselage will not be aligned to the path during a maneuver in the vertical plane as it is shown in Figure 4.4 (c). The helicopter instead will follow the path as it is shown in Figure 4.4 (b).

Calculation of the outer loop inputs

We can finally address the problem of generating the reference inputs for the outer control loop. The inputs will be calculated in the form of control errors (difference between the current helicopter state and the desired one) as follows.

1. Calculation of the position error vector $\delta\vec{p}$.

   The position error vector is the difference between the control point position $\vec{p}_{cp}^{\,n}$ and the helicopter position given by the INS/GPS system $\vec{p}_{heli}^{\,n}$. In order to be used in the outer loop control equations the vector must be rotated from the navigation frame to the body frame using the rotation matrix C_n^b:

   $$\delta\vec{p}^{\,b} \;=\; C_n^b(\vec{p}_{cp}^{\,n} - \vec{p}_{heli}^{\,n})$$

   As explained in section 4.3.2, the control point position is calculated using feedback from the helicopter position. The method does not search for the control point along the whole path segment but it remembers the value of the parameter s (e.g. the previous control

point) from the previous control cycle and starts the search from there. By doing this, the search is very fast since the new control point will not be far from the previous one (the control function is called with a frequency of 50Hz). At the beginning of each segment, the parameter value is set to zero.

Once the new value of s is found, the position error vector $\delta\vec{p}^{\,b}$ can be calculated together with the local path tangent $\vec{t}$ and curvature $\vec{k}$.

2. Calculation of the velocity error vector $\delta\vec{v}$.

The velocity error vector is the difference between the target velocity v_{targ} and the helicopter velocity given by the INS/GPS system $\vec{v}^{\,n}_{heli}$. The target velocity is obviously tangent to the geometric path. The direction of the tangent vector is given by:

$$\vec{\tau}^{\,n} = \frac{\vec{t}^{\,n}}{|\vec{t}|} \tag{4.17}$$

In order to be used in the control equations in 4.25, the velocity error vector must be expressed in the body frame in the same way as the position error vector:

$$\delta\vec{v}^{\,b} = C^b_n(v_{targ} \cdot \vec{\tau}^{\,n} - \vec{v}^{\,n}_{heli}) \tag{4.18}$$

The calculation of v_{targ} (desired helicopter velocity along the path) must take into account the helicopter kinematic constraints, the limitation due to the maximum allowable vertical velocity and the particular phase of the flight path that is *acceleration, cruising* and *braking*.

Let us call v_{targ1} the velocity calculated according to the acceleration, cruising and braking condition, v_{targ2} the velocity calculated according to the helicopter kinematic constraints and v_{targ3} the velocity according to the vertical speed limitation. These three velocities will

be calculated in the following part of this section and the minimum value among the three will be used as v_{targ}. This procedure is repeated at each control cycle and in this way the velocity profile for the path is shaped.

The calculation of v_{targ1} is done considering the acceleration, cruising and braking conditions. The calculation scheme can be represented by the state-machine in Figure 4.5.

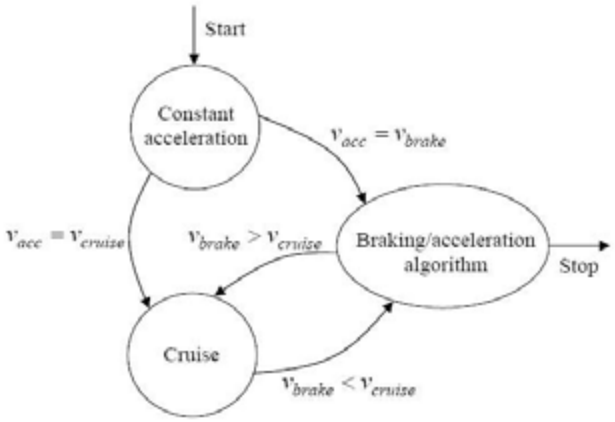


Figure 4.5: State-machine representing the calculation of the velocity v_{targ1}.

The *constant acceleration* phase is activated only at the beginning of the first segment of the path and it ends when $v_{acc} = v_{cruise}$ or $v_{acc} = v_{brake}$. v_{cruise} is set by the path planner while the calculation of v_{brake} will be explained below. During the acceleration phase v_{acc} is increased at a constant rate of 1.2 m/s^2. The *cruising* phase is active when the braking and the acceleration phases are off. The *braking* condition is activated when the following condition becomes true:

$$v_{brake} < v_{cruise} \tag{4.19}$$

with

$$v_{brake} = \sqrt{|(2 \cdot l_{end} \cdot Acc + v_{end}^2)|} \tag{4.20}$$

where l_{end} is the distance, calculated along the path segment, that the helicopter has to fly to reach the end of the segment and Acc is the desired acceleration during the braking phase (its value is set to 1.2 m/s). v_{end} is the desired velocity at the end of each path segment and it is assigned by the path planner. The condition 4.19 means that the helicopter must start to brake when the distance to the end of the segment is equal to the distance necessary to bring the helicopter from the velocity v_{cruise} to v_{end} with the desired acceleration. If v_{end} is greater than v_{cruise} the helicopter increases the velocity instead. The method used for the calculation of l_{end} is explained in [14].

The calculation of v_{targ2} takes into account the kinematic constraints described by the equations in 4.15. For safety reasons the flight envelope has been limited to: $r_{max} = 40$ rad/sec maximum yaw rate, $\phi_{max} = 15$ deg maximum roll angle, $\theta_{max} = 15$ deg maximum pitch angle and for the vertical acceleration a load factor of $Nz_{max} = T_{max}/g = 1.1$. The value of Nz_{max} has been chosen using the fact that a load factor of 1.1 means increasing the helicopter weight 10 percent. The maximum takeoff weight of the Rmax is 94 kg and the Rmax weight used in this experimental test is around 80 kg, so a load factor of 1.1 ensures enough safety. The second equation in 4.15 represents the forward velocity u_2 achievable with the maximum pitch angle. It is not considered as a constraint here since the cruise velocity assigned by the path planner will always be smaller than u_2. u_{max} is calculated using 4.16. Finally we can calculate v_{targ2} as:

$$v_{targ2} = \frac{u_{max}}{\tau_x^b} \tag{4.21}$$

where τ_x^b is the projection of the vector in 4.17 on the helicopter X_b body axis.

Figure 4.6 shows the three velocities u_1, u_3 and u_4, depending on the path curvature radius, calculated from equations in 4.15 using the limitation r_{max}, ϕ_{max} and Nz_{max}. The most limiting velocity is u_3, which means the maximum roll angle ϕ_{max} is the most limiting factor for the maximum velocity u_{max}. The situation can change if the takeoff weight is more than 80 kg, in this case Nz_{max} could become the limiting factor.

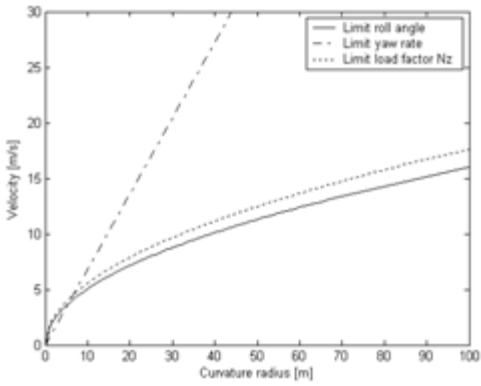


Figure 4.6: Velocity constraint due to the maximum yaw rate, roll angle and load factor.

It is important to mention that the velocity u_{max} is calculated using Equation 4.15 which is derived from a trimmed flight condition. As previously mentioned, using a path described by a cubic polynomial,

the curvature changes continuously and the helicopter almost never flies in trimmed conditions. The resulting u_{max} might not always be consistent with the attitude dynamics $\dot{\theta}$ and $\dot{\phi}$. The faster the speed, the more the attitude dynamics is relevant.

The velocity v_{targ3} is calculated as follows. The vertical velocity must be limited during a descending path because of the vortex ring which can build up around the main rotor in this flight condition. In this case, the helicopter descends into rotor down-wash and enters what is commonly called the vortex ring state (VRS). This situation can cause loss of helicopter control and might be difficult to recover. For this reason a limitation on the descending velocity component is necessary. The flow state diagram of Figure 4.7 shows the combination of horizontal and vertical speed where VRS occurs. This diagram was developed at the Aviation Safety School, Monterey CA, in the late 1980s for use by mishap investigators in their analysis of several of these events. In this thesis, this diagram has been used as a guideline in choosing the right combination of vertical and horizontal speed in order to avoid a VRS situation. More details on VRS phenomenon can be found in [50].

The flow state diagram axes are parametrized with the hovering induced velocity which is calculated in feet/minutes as follows:

$$V_i = 60\sqrt{\frac{DL}{2\rho}} \tag{4.22}$$

where DL is the rotor disk loading (lbs/ft^2) and ρ the air density ($\frac{SLUGS}{ft^3}$). This diagram will be used to calculate a safe vertical speed.

For the Rmax helicopter, the induced hovering velocity calculated using 4.22 and converted in m/s results in $V_i = 6.37$ m/s. The value has been calculated using the air density at sea level $\rho = 0.002377 \frac{SLUGS}{ft^3}$, Rmax weight of 80 kg and rotor diameter $R = 3.115$ meters. From the diagram, the vertical speed when the light turbulence first occurs

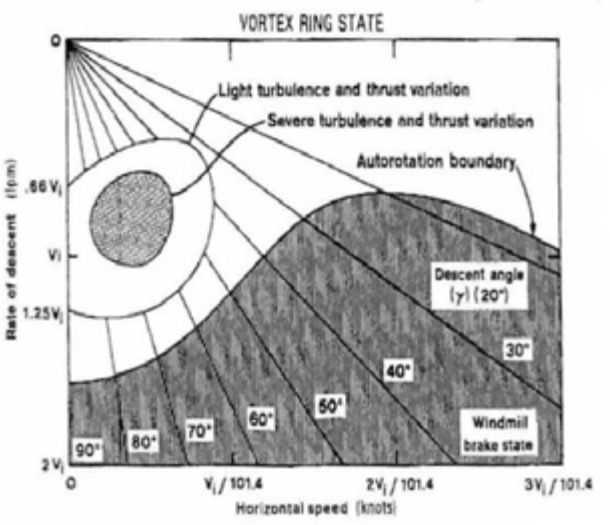


Figure 4.7: Flow states in descending forward flight (from "The vortex ring state fallacy" by R.E. Joslin, 2003).

is around $w_1 = 0.5V_i = 3.18$ m/s. It can also be observed that for a descent angle smaller than 30 deg the VRS area is avoided completely.

The maximum vertical velocity profile chosen for the Rmax is shown in Figure 4.8 (dashed line) where for safety reasons w_1 has been reduced to 1.5 m/s for a descent angle γ greater than 30 deg, while for γ smaller then 30 deg the descending velocity has been limited to $w_2 = 3$ m/s.

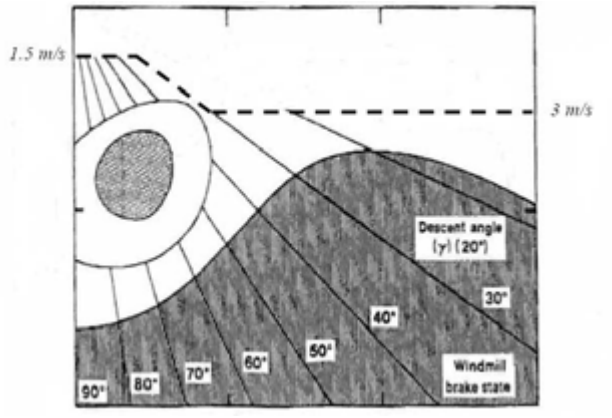


Figure 4.8: Maximum descent velocity used in the PFCM for the Rmax helicopter.

The calculation of v_{targ3} is then:

$$\begin{aligned} \gamma &= atan(\frac{\tau_z^n}{\tau_x^n}) \\ w_{MAXdescent} &= 1.5 \quad 90^\circ > \gamma \geq 30^\circ \\ w_{MAXdescent} &= 3 \quad 30^\circ > \gamma > 0^\circ \\ v_{targ3} &= \frac{w_{MAXdescent}}{\tau_z^n} \end{aligned} \tag{4.23}$$

Finally, the helicopter velocity profile is given by:

$$v_{targ} = min(v_{targ1}, v_{targ2}, v_{targ3})$$

3. Calculation of the heading error $\delta\psi$.

   The heading error is given by:

$$\delta\psi = atan2(t_y^n, t_x^n) - \psi_{heli}$$

   where ψ_{heli} is the helicopter heading given by the INS/GPS system.

4. Calculation of the feed forward control terms r_{ff}, ϕ_{ff}.

   The terms r_{ff} and ϕ_{ff} are calculated from the first and third equations in 4.14 where the component of the curvature radius R_y^b is calculated as follows:

$$\begin{aligned} \vec{k}^b &= C_b^n \vec{k}^n \\ R_y^b &= \frac{1}{k_y^b} \end{aligned} \tag{4.24}$$

   The feed forward terms will be used in the outer control loop to enhance the tracking precision.

4.4 Outer loop control equations

The PFCM control equations implemented on the Rmax are the following:

$$\begin{aligned} YAW_{yacs} &= r_{ff} + K_1^y \delta\psi \\ PITCH_{yacs} &= K_1^p \delta p_x + K_2^p \delta v_x + K_3^p \frac{d}{dt}\delta v_x + K_4^p \int \delta v_x dt \\ ROLL_{yacs} &= \phi_{ff} + K_1^r \delta p_y + K_2^r \delta v_y \\ THR_{yacs} &= K_1^t \delta p_z + K_2^t \delta v_z + K_3^t \int \delta v_z dt \end{aligned} \tag{4.25}$$

where the K's are the control gains, r_{ff} and ϕ_{ff} are the feed-forward control terms resulting from the model in 4.14. The other two terms θ_{ff} and T_{ff} relative to the pitch and throttle channels have not been implemented. These channels are controlled by the feedback loop only. $\delta\psi$ is the heading error, $\delta\boldsymbol{p}$ is the position error vector (difference between the control point and helicopter position), $\delta\boldsymbol{v}$ is the velocity error vector (difference between target velocity and helicopter velocity).

Adding the feed forward control terms, especially on the roll channel, results in a great improvement in the tracking capability of the control system compared to a PID feedback loop only. Results of the control approach are shown in the next section.

4.5 Experimental results

This section presents experimental results of the PFCM implemented on the Rmax helicopter. In Figure 4.9, a 3D path is flown starting from an altitude of 40 meters and finishing at 10 meters. The path describes a descending spiral and the velocity v_{cruise} given by the path planner was set to 10 m/s. In Figure 4.10, the velocity profile of the path is represented and it can be observed that as soon as the helicopter reaches 10 m/s it slows down in order to make the turn with the compatible velocity. This was an early test, where the roll angle limitation was quite strict (around 8 deg). This explains the consistent decrease in velocity. In addition, the acceleration phase was missing. In fact, the commanded target velocity, represented by the dashed line, starts at 10 m/s. This resulted in an abrupt pitch input at the beginning of the flight.

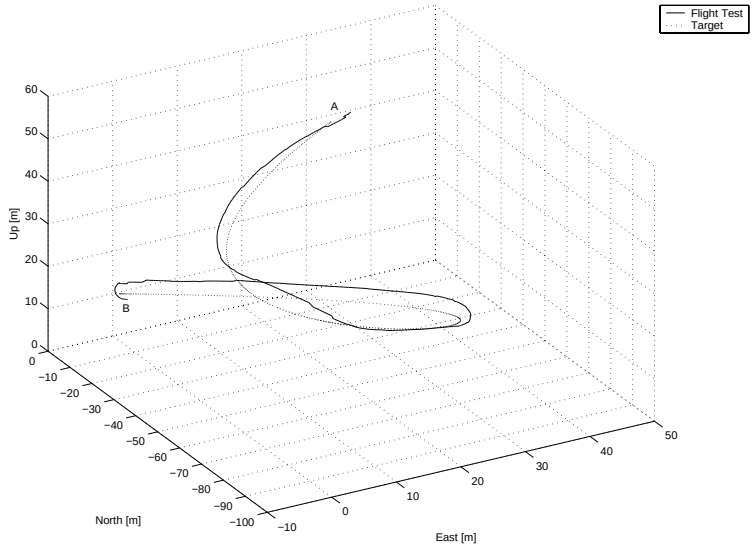


Figure 4.9: Target and actual 3D helicopter path.

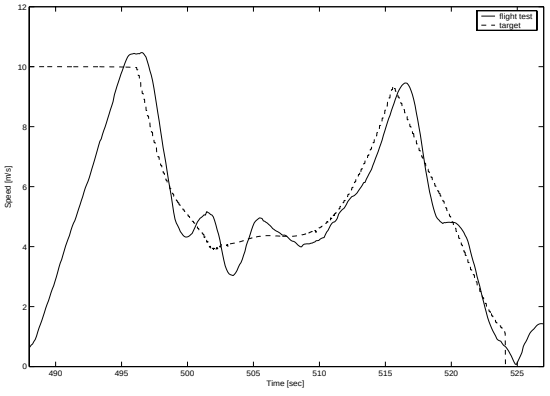


Figure 4.10: Target and actual helicopter speed.

Table 4.1 shows the results of several paths flown with different wind conditions and different velocities. The table reports three flight sessions (separated by horizontal lines) flown on three different days so as to cover three different wind conditions. In order to give more generality to the results, representative paths of typical flight maneuvers have been chosen. In the HR path the helicopter describes a complete turn in the horizontal plane turning right, in the DR path the helicopter makes the same turn while it is descending from 40 to 10 meters and in the CR path the helicopter turns while climbing from 10 to 40 meters. The same flights are repeated turning left instead. Figure 4.9 for example is a DL path.

The first column of the table shows the kind of path flown, the second, third and fourth column are the average error, maximum error and standard deviation error, and the fifth and sixth columns show the maximum ground speed reached and the average wind speed. The error is the distance of the helicopter to the reference path and is calculated using the INS/GPS signal, which is also used as control signal during flight (an independent source would have been a better reference for the purpose of this statistics).

To summarize the results of the table, the first session gives the worst results because of the wind, moreover the right turn gave better results than the left one because the wind was blowing from the side. In the second session the overall performance increases because of less wind. In the third session several straight lines of 170 meters at low speed were flown, during the test the wind was negligible.

Although the PFCM just described has exhibited a satisfactory performance in terms of robustness, the tracking capabilities in case of maneuvered flight (when path curvature change rapidly) were not satisfactory. For this reason, the possibility to improve the tracking performance was investigated in the case of maneuvered flight without a major redesign of the control system.

The lateral control has been modified by adding an extra control loop on the roll channel besides the YACS control system. The new lateral control loop is depicted in Figure 4.11 b). From the diagram one can compare the difference between the previous control scheme, Figure 4.11 a), and the new one Figure 4.11 b).

The inner compensator that was added provides a phase lead compensation with an integral action and has the following structure:

Path [-]	Av Err [m]	Max Err [m]	St Dev [m]	Speed [m/s]	Wind [m/s]
HR	1.2	3.4	0.7	10	4
HL	1.9	4.1	1.3	10	4
DR	1.5	2.8	0.7	10	4
DL	1.8	3.5	1.1	10	4
CR	1.7	3.3	0.7	10	4
CL	1.9	4.1	1.3	10	4
HR	1.1	2.7	0.8	10	2
HL	0.8	2.2	0.6	10	2
DL	0.9	1.8	0.5	10	2
SLN	0.3	0.8	0.2	3	≈ 0
SLN	0.5	1.4	0.3	3	≈ 0
SLN	0.5	1.9	0.5	3	≈ 0
SLN	0.6	1.4	0.3	3	≈ 0
SLN	0.4	1.3	0.3	3	≈ 0

HR = Horizontal Right HL = Horizontal Left
DR = Descending Right DL = Descending Left
CR = Climbing Right CL = Climbing Left
SLN = Straight Line

Table 4.1: Experimental data

$$C(s) = K(\alpha\frac{1+s}{1+\alpha s}) + K_I\frac{1}{s} \tag{4.26}$$

The phase lead compensation increases the bandwidth and, hence, makes the closed loop system faster, but it also increases the resonance frequency with the danger of undesired amplification of system noise. The control system has been tuned in simulation but a second tuning iteration was needed on the field due to the presence of damped oscillations on the roll channel.

An experimental comparison between the modified control system and the previous one is shown in Figure 4.12. The target velocity in both cases was set to 10 m/s. Figure 4.13 depicts the target velocity and the actual helicopter velocity relative to the path on the right side of Figure 4.12. The diagram on the right side in Figure 4.12 depicts the path flown with the basic PFCM controller (Figure 4.11 a). One can observe that in the dynamic part of the path, where the curvature changes rapidly, the controller is slow. This results in a relevant tracking error.

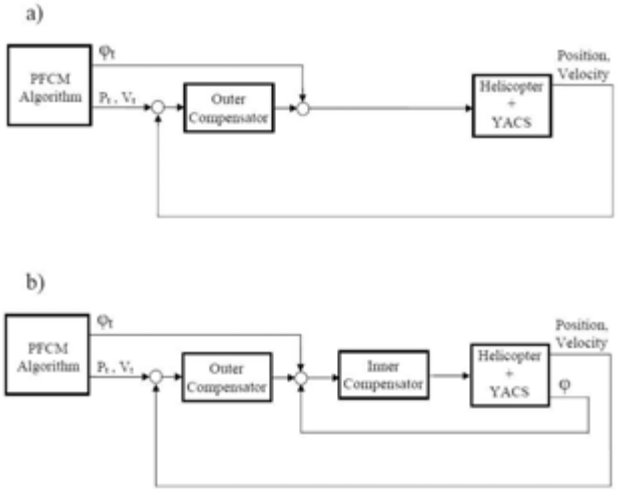


Figure 4.11: a)Previous lateral control. b)Modified lateral control loop using a lead compensator.

The diagram on the left in Figure 4.12 depicts a test of the same path flown with the modified roll control loop (Figure 4.11 b). The new lateral control scheme improves the tracking capability in the presence of fast curvature change. The helicopter can follow the dynamic part of the path with considerably lower tracking error.

4.6 Conclusions

The PFCM developed here has been integrated in the helicopter software architecture and it is currently used in a number of flight missions carried out in an urban area used for flight-tests. The goal has been the development of a reliable and flexible flight mode which could be integrated robustly with a path planner. Safety mechanisms have been built-in to the PFCM in order to handle communication failures with the path planner

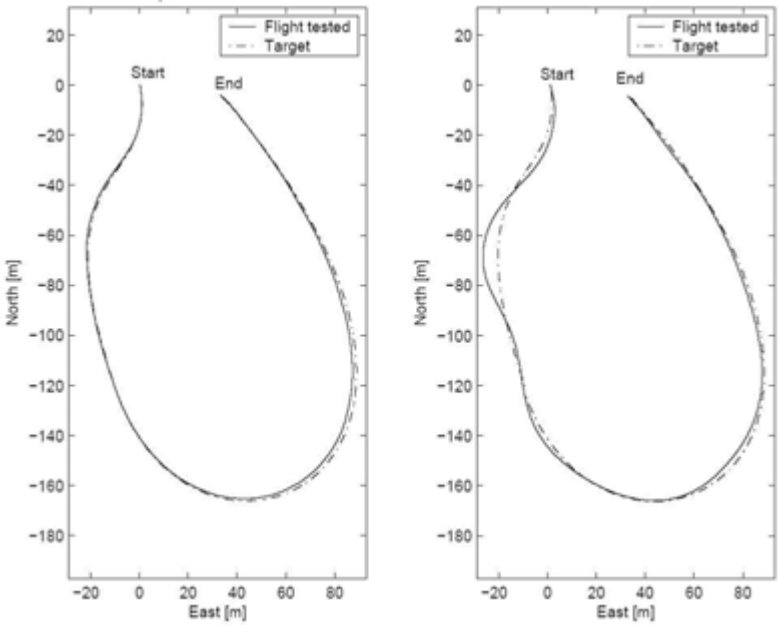


Figure 4.12: Comparison of path tracking performances using two different roll control strategy. On the right side is depicted the flight-test of the modified roll control loop with the lead compensator added. On the left the same test is done using the old roll control configuration. The flight-tests were performed at 36 km/h velocity for both paths.

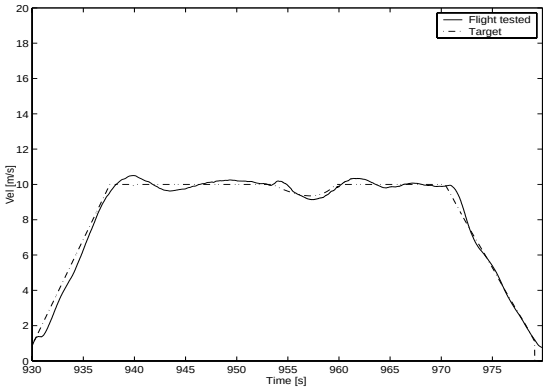


Figure 4.13: Target velocity and actual helicopter velocity.

(this can happen since the path planner is implemented on a separate computer). More details on this topic can be found in [14]. Moreover, since the path planner was developed before the PFCM, a number of constraints have been inherited and have shaped the development of the PFCM. For example, the fact that a geometric segment was precomputed and given to the control system without taking into account the dynamic constraints of the helicopter, has led to the development of the feedback algorithm to update the control point on the reference path.

Part II

Vision-based Navigation

Chapter 5

Terrain Relative Navigation Based on Geo-reference Imagery

5.1 Introduction

In this chapter, an approach to vision-based state estimation using geo-referenced aerial imagery to navigate in outdoor environments without GPS is presented. The content of this chapter has been published in [13].

One of the main concerns which prevents the use of UAV systems in populated areas is the safety issue. State of the art UAV systems are still not able to guarantee an acceptable level of safety to convince aviation authorities to authorize their use in populated areas (except in rare cases such as war zones).

There are several problems which have to be solved before unmanned aircraft can be introduced into civilian airspace. One of them is GPS vulnerability [32]. A standard UAV navigation system often relies on GPS and inertial sensors (INS). If the GPS signal for some reason becomes unavailable or corrupted, the state estimation solution provided by the INS

alone drifts in time and will be unusable after a few seconds (especially for small-size UAVs which use low-cost INS). The GPS signal can also be unreliable when operating close to obstacles due to multi-path reflections. In addition, jamming of GPS has arisen as a major concern for users due to the availability of GPS jamming technology on the market. Therefore UAVs which rely blindly on a GPS signal are quite vulnerable to malicious actions. For this reason, a navigation system for autonomous UAVs must be able to cope with short and long term GPS fallouts.

The problem addressed in this chapter is concerned with finding techniques for a UAV to be able to navigate to home-base in case its GPS signal is lost ("homing" problem). The Yamaha Rmax helicopter presented in chapter 2 is used as a test-bed for the development and testing of a vision-based navigation architecture which can cope with GPS failures. The approach developed fuses information obtained from an inertial measurement unit (IMU) composed of three accelerometers and three gyros with information extracted from a passive monocular video camera. A passive video camera is an appealing sensor which can be used to solve navigation related problems. Almost every UAV already has a video camera as a standard sensor in its payload package. Compared to other sensors, e.g. laser, video cameras are quite light and less power hungry. On the other hand, passive video cameras are quite sensitive to environmental light conditions. The navigation system proposed here, replaces the GPS signal by combining visual odometry with an algorithm which registers the on-board video to geo-referenced satellite or aerial images. Such images must be available on-board the UAV beforehand or downloaded in flight. The growing availability of high resolution satellite images (for example provided by Google Earth) makes this topic very interesting and timely.

The vision-based architecture proposed is depicted in Figure 5.3. It is composed of an error dynamics Kalman filter (KF) that estimates the navigation errors of the INS and a separate Bayesian filter, called *point-mass filter* (PMF) [6], which estimates the absolute position of the UAV on the horizontal plane fusing information from two image processing techniques called *feature tracking* and *geo-reference image registration*. The 2D position estimated from the PMF together with barometric altitude information obtained from an on-board barometer, are used as position measurement updates for the KF.

Feature tracking and image registration are complementary position information sources. The KLT feature tracker [61] is used to track corner features in an on-board video image from subsequent frames. A homography-based odometry function uses the KLT results to calculate the distance traveled by the UAV. Since the distance calculated by the odometry function is affected by drift, a mechanism which compensates for the drift error is still needed. For this purpose the information from a geo-reference image registration module is used. When the image registration is performed correctly, it is possible to calculate the absolute position of the UAV in a drift-free manner. In other words, the position information obtained is similar to the one provided by a GPS. Visual odometry is used since it provides robustness to the position estimation problem. It is capable of providing position information over rural terrain where the image registration module is usually less reliable.

This work proposes a navigation architecture that, taking advantage of geo-reference information, is capable of providing high-rate and drift-free full state estimation without using a GPS. The state estimation is suitable for autonomous flight control. The approach is validated using flight-test data collected from the Rmax UAV helicopter. The architecture is also implemented on the Rmax helicopter computers and tested on-board in vision-based autonomous flight experiments.

The first contribution of this work is to explore the possibility of using one video camera both as a velocity meter (odometry) and a positioning device (image registration). We believe that this is a very practical and innovative concept. The second contribution is the development of a sensor fusion architecture which combines vision-based information together with inertial information in an original way. The third contribution is a real-time implementation and experimental results in field trials of the approach proposed on the Yamaha Rmax unmanned helicopter.

5.2 Related work

The terrain relative navigation problem is of great interest not only for terrestrial applications but also for future space missions. One of the requirements for the next lunar mission in which NASA/JPL is involved is to

autonomously land within 100 meters of a predetermined location on the lunar surface. Traditional lunar landing approaches based on inertial sensing do not have the navigational precision to meet this requirement [33]. A survey of different terrain navigation approaches can be found in [33] where methods based on passive imaging and active range sensing are described.

Many research groups are focusing on non-GPS navigation problems. One technique which could be applied is Simultaneous Localization and Mapping (SLAM). The goal of SLAM is to localize a robot in the environment while mapping it at the same time. Prior knowledge of the environment is not required. In SLAM approaches, an internal representation of the world is built on-line in the form of a landmarks database. Such a representation is then used for localization purposes. For indoor robotic applications SLAM is already a standard localization technique. More challenging is the use of such technique in large outdoor environments. Some examples of SLAM applied to aerial vehicles can be found in [34, 35, 36].

The use of reference images for aircraft localization purposes is also investigated in [57]. A reference image matching method which makes use of the Hausdorff distance using contour images representation is explored. This work focuses mainly on the image processing issues and gives less emphasis to architectural and fusion schemes which are of focus in this chapter. The results in [57] cannot be properly compared to the work in this thesis as the position error results are obtained at a substantially higher flight altitude.

There are also other kinds of terrain navigation methods which are not based on aerial images but on terrain elevation models. In this case a measurement of the flight altitude relative to the ground is required. Matching the ground elevation profile, measured with a radar altimeter for example, to an elevation database allows for aircraft localization. A commercial navigation system called TERNAV (TERrain NAVigation) is based on such a method. The navigation system has been implemented successfully on some military jet fighters and cruise missiles. In the case of small UAVs and more specifically for unmanned helicopters, this method does not appear to be appropriate. Compared to jet fighters, UAV helicopters cover short distances at very low speed so the altitude variation is quite poor in terms of allowing ground profile matching.

Advanced cruise missiles implement a complex navigation system based

on GPS, TERNAV and DSMAC (Digital Scene Matching Area Correlation). During the cruise phase the navigation is based on GPS and TERNAV. When the missile approaches the target the DSMAC is used to increase the position accuracy. The DSMAC matches the on-board camera image to a reference image using correlation methods.

Although the use of reference images for navigation purposes has already been investigated in previous works, it is not straight forward to transfer the technology to small UAVs. The limited payload capability and sensor accuracy of such platforms, compared to jet fighters or cruise missiles, imposes additional constraints which require a specialized solution to the problem. It has been demonstrated through real-time experimental results that the method developed in this thesis is applicable to small size UAV platforms which uses commercial off-the-shelf computers.

5.3 Assumptions and limitations

The vision-based navigation approach proposed makes use of aerial or satellite 2D images available beforehand on-board the UAV. The fact that 2D reference images are used implies that the observation of the world from the UAV must be approximately 2D. The approximation holds as long as the UAV flight altitude is substantially larger than the typical ground object height. If this is true the world can be assumed to be flat. The assumption adopted here is that the UAV flight altitude satisfies this requirement.

The inherent limitation of using reference images for navigation purposes is that the terrain characteristics must have distinctive features which allow a robust correlation with the reference image. For instance a matching algorithm applied in rural areas will not perform equally well as for urban areas where the features are more structured. Another factor to be considered is the scene stability. Since the reference images are usually taken months or years in advance, changes in the landscape could represent a problem. In addition, repetitive scenes with periodic structures could also lead to false matching locations.

To mitigate the impact of such problems, it could be wise to classify beforehand the reference images according to certain navigability criteria. In [65, 12, 29], methods for selecting scene matching areas suitable for ter-

rain navigation are proposed. Such methods could be used during the flight planning phase in order to detect and avoid areas with poor navigability properties.

5.4 Reference systems and camera projection model

In this section, a brief overview of the coordinate reference systems and transformations between them will be given. In section 3.4, the navigation and body frames together with the transformation between the two systems were introduced. Now, since the camera is used for navigation purposes, the position and displacement computed using the camera must be transformed into the navigation reference system. The position of a feature, imaged by the video camera, is first computed in pixel coordinates in the image reference system. The position is then transformed into the navigation reference system passing through the camera system, the gimbal system and the body system. The camera is mounted on a gimbal system below the helicopter allowing rotations in azimuth (β) and elevation (α) as depicted in Figure 5.1.

The *gimbal reference system* is an orthogonal axis set which has its origin coinciding with the center of rotation of the camera's gimbal, the X^g axis aligned to the camera's optical axis, the Y^g axis pointing right in the image plane and the Z^g pointing down in the image plane (Figure 5.1).

The *camera reference system* is an orthogonal axis set which has its origin coinciding with the camera's optical center, the X^c axis pointing upward in the image plane, the Y^c axis pointing to the right in the image plane and the Z^g in the same direction of the optical axis (Figure 5.1).

The *image reference system* (X^i, Y^i) is represented in Figure 5.2. The projection of a real world point p^{world} in the image plane p^i is expressed in pixel coordinates. The origin of the system is in the upper left corner of the image plane, the X^i axis points to the right direction while the Y^i axis points downward.

The camera projection model used in this thesis is a simple pin-hole model. The transformation between the image reference system and the

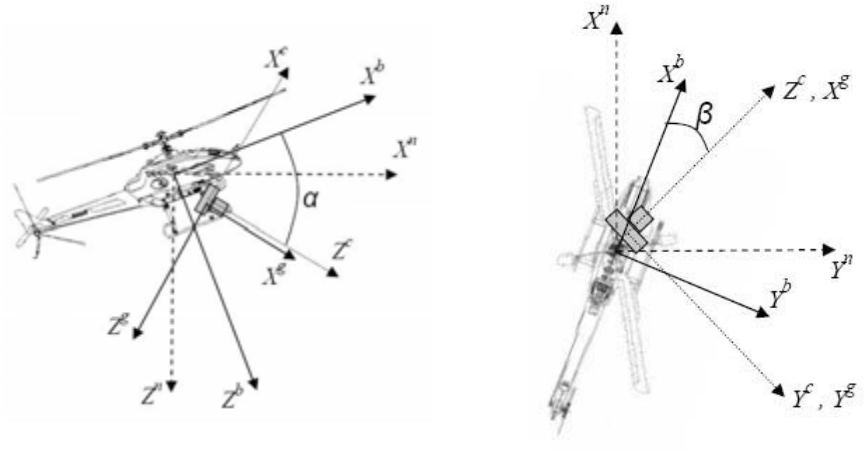


Figure 5.1: Reference system representation.

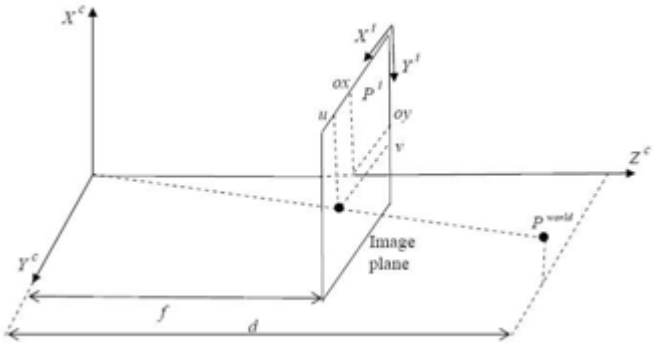


Figure 5.2: Image reference system and camera projection model.

camera reference system is realized through the calibration matrix K:

$$K = \begin{bmatrix} 0 & f_x & ox \\ -f_y & 0 & oy \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

where f_x, f_y are the camera focal lengths in the x and y directions (represented with f in Figure 5.2) and ox, oy is the center point of the image in pixels. The camera's lens distortion compensation is not applied in this approach since the camera being used has a relatively large focal length (approximately 45 degrees horizontal angle of view), therefore the image distortion is considered to be small for the precision required. In addition, the image center ox and oy is considered coinciding with the camera's principal point. A feature in the image plane of coordinates $\vec{p}^{\,i} = (u, v, 1)$ is expressed in the navigation frame with the following relation:

$$\vec{p}^{\,n} = d \left(K \, C_g^c \, C_b^g \, C_n^b \right)^{-1} \vec{p}^{\,i} \tag{5.2}$$

where d is the feature's depth, C_b^g is the body-gimbal rotation matrix given by:

$$C_b^g = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta & \sin\alpha \\ -\sin\beta & \cos\beta & 0 \\ -\sin\alpha\cos\beta & -\sin\alpha\,\sin\beta & \cos\alpha \end{bmatrix} \tag{5.3}$$

and C_g^c, the gimbal-camera rotation matrix given by:

$$C_g^c = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.4}$$

The camera is rigidly mounted on the gimbal system so the relative orientation is fixed. Tha angles α and β are usually given by the gimbal unit. In Equation 5.2 the C_n^b navigation-body rotation matrix is also used. Its definition was given in section 3.4. The offset between the origin of the different reference systems is neglected because it is small for the platform used in this thesis.

5.5 Vision-based navigation architecture

The sensor fusion architecture developed is presented in Figure 5.3 and is composed of several modules. It can work in GPS modality or vision-based modality if the GPS is not available.

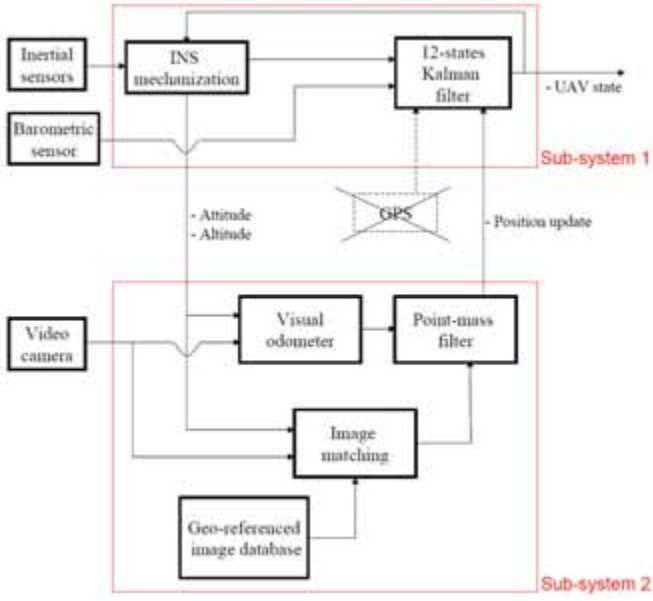


Figure 5.3: Sensor fusion architecture.

A traditional KF (sub-system 1) is used to fuse data from the inertial sensors (3 accelerometers and 3 gyros) with a position sensor (GPS or vision system in case the GPS is not available). An INS mechanization function performs the time integration of the inertial sensors while the KF function estimates the INS errors. The KF is implemented in the error dynamics form and uses 12 states: 3 dimensional position error, 3 dimensional velocity error, 3 attitude angle error (pitch, roll, heading) and 3 accelerometer biases. In appendix A of this thesis, more details on the KF

implementation adopted for this work can be found. The error dynamics formulation of the KF is widely used in INS/GPS integration [40, 59, 56]. The estimated errors are used to correct the INS solution.

The vision system (sub-system 2) is responsible for computing the absolute UAV position on the horizontal plane. This position is then used as a measurement update for the KF. The visual odometry computes the helicopter displacement by tracking a number of corner features in the image plane using the KLT algorithm [61]. The helicopter displacement is computed incrementally from consecutive image frames. Unfortunately the inherent errors in the computation of the features location accumulate when old features leave the frame and new ones are acquired, producing a drift in the UAV position. Such drift is less severe than the position drift derived from a pure integration of typical low-cost inertial sensors used on small UAVs. The drift of pure INS integration will be compared to the odometry drift in the experimental section. The image matching module provides drift-free position information which is integrated in the scheme through a grid-based Bayesian filtering approach. The matching between reference and on-board image is performed using the normalized cross-correlation algorithm [49].

5.6 Visual odometry

Visual odometry for aerial navigation has been an object of great interest during the last decade [4, 28, 41, 10]. The results in [4] have shown how visual odometry is capable of stabilizing an autonomous helicopter in hovering conditions. In that work the odometry was based on a template matching technique where the matching between subsequent frames was obtained through use of the sum of squared differences (SSD) minimization of the gray scale pixel values of the templates. The helicopter's attitude information was taken from an IMU sensor. Specialized hardware was expressly built for this experiment in order to properly synchronize attitude information with the video frame. Most of the recent work [28, 41, 10] on visual odometry for airborne applications is based on homography estimation under a planar scene assumption. In this case, the relation between points of two images can be expressed as $x_2 \approx Hx_1$ where x_1 and x_2 are

corresponding points, expressed in *homogeneous coordinates*, observed from two image views 1 and 2 and H is the 3x3 homography matrix. The symbol $\approx$ indicates that the relation is valid up to a scale factor. A point is expressed in homogeneous coordinates when it is represented by equivalence classes of coordinate triples $(k\,x,\ k\,y,\ k)$ where k is a multiplicative factor. The camera rotation and displacement between two camera positions c1 and c2, can be computed from the homography matrix decomposition [30]:

$$H = K\left(C_{c1}^{c2} + \frac{1}{d}\vec{t}^{\,c2}\vec{n}^{\,c1T}\right)K^{-1} \tag{5.5}$$

where K is the camera calibration matrix introduced in section 5.4, $\vec{t}^{\,c2}$ is the camera translation vector expressed in the camera 2 reference system, C_{c1}^{c2} is the rotation from camera 1 to camera 2, $\vec{n}^{\,c1}$ is the unit normal vector to the plane being observed and expressed in the camera 1 reference system and d is the distance of the principal point of the camera 1 to the plane.

The visual odometry system implemented in this work is based on robust homography estimation. Comprehensive theoretical and practical details on robust homography estimation can be found in [30]. The background theory on robust homography estimation has been reported in appendix B of this thesis.

The homography matrix H is estimated from a set of corresponding corner features being tracked from frame to frame. H can be estimated using the direct linear transformation algorithm (DLT) with a minimum number of four feature points in a non-degenerate configuration (see appendix B). In practice the homography is estimated from a higher number of corresponding feature points (50 or more) and the random sample consensus (RANSAC) algorithm [26] is used to identify and then discard incorrect feature correspondences. The RANSAC is an efficient method to determine among the set of tracked features S the subset S_i of inlier features and the outlier subset S_o so that the estimate H is unaffected by outliers in the data. The RANSAC algorithm is described in appendix B.

The feature tracker used in this work is based on the KLT algorithm. The algorithm selects a number of features in an image according to a "goodness" criteria described in [55]. Then it tries to re-associate the same

features in the next image frame. The association is done by minimizing the sum of squared differences over patches taken around the features in the second image. Such association criteria gives good results when the feature displacement is not too large. Therefore it is important that the algorithm has a low execution time. The faster the algorithm is, the more successful is the association process.

In Figure 5.4 the RANSAC algorithm has been applied on a set of features tracked in two consecutive frames. On the left are represented the feature displacements computed by the KLT algorithm while on the right the set of outlier features has been detected and removed using RANSAC.

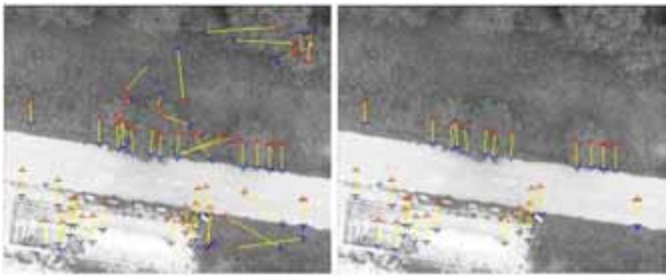


Figure 5.4: On the left is displayed the set of features tracked with the KLT algorithm. On the right the outlier feature set has been identified and removed using the RANSAC algorithm.

Once the homography matrix has been estimated it can be decomposed into its rotation and translation components in the form of Equation 5.5 using singular value decomposition as described in [30]. However, homography decomposition is a poorly-conditioned problem especially when using cameras with a large focal length [45]. The problem also arises when the ratio between the flight altitude and the camera displacement is high. For this reason it is recommended to use inter-frame rotation information from other sensor sources if available. If the camera rotation is available the homography can be reduced to the rotation-free case of planar homology. The known camera rotation transformation C can be applied to the set of points of the first image, then the homology decomposition can be expressed as follows:

$$H = I - \vec{t}\,\vec{n}^{\,T} \tag{5.6}$$

Equation 5.6 has six degrees of freedom, three degrees associated with the translation vector $\vec{t}$, two degrees with the unitary normal vector $\vec{n}$ and the last one associated with the unknown absolute scale (usually measured with an altimeter sensor). The reader can find information about the homology decomposition solution in [45, 30].

The odometry presented in this work makes use of inter-frame rotation information coming from the KF but does not compute the homology decomposition. The solution presented in the remainder of this section makes use of the knowledge of the terrain slope, in other words the direction of the normal vector of the terrain is assumed to be known. In the experiment presented here the terrain will be considered non-sloped. Information about the terrain slope could also be extracted from a database if available.

The goal then is to compute the helicopter translation in the navigation reference system from Equation 5.5. The coordinate transformation between the camera and the INS sensor is realized with a sequence of rotations. The translation between the two sensors will be neglected since the linear distance between them is small.

The rotations sequence (5.7) aligns the navigation frame (n) to the camera frame (c) passing through intermediate reference systems named helicopter body frame (b) and camera gimbal frame (g) as follows:

$$C_c^n \quad = \quad C_b^n C_g^b C_c^g \tag{5.7}$$

C_b^n is given by the KF described in section 5.5, C_g^b is measured by the pan/tilt camera unit, and C_c^g is constant since the camera is rigidly mounted on the pan/tilt unit. The transformation between the camera reference system and the image reference system (pixels) is given by the calibration matrix K (5.1). The rotation matrices are defined according to the reference system definitions and are given in section 5.4.

If c1 and c2 are two consecutive camera positions, then considering the following relationships:

$$\begin{aligned} C_{c1}^{c2} &= C_n^{c2} C_{c1}^n \\ \vec{t}^{\,c2} &= C_n^{c2} \vec{t}^{\,n\,T} \\ \vec{n}^{\,c1} &= C_n^{c1} \vec{n}^{\,n\,T} \end{aligned}$$

and substituting in Equation 5.5 give:

$$H = K C_n^{c2} \left(I + \frac{1}{d} \vec{t}^{\,n} \vec{n}^{\,n\,T} \right) C_{c1}^n K^{-1} \tag{5.8}$$

Considering that the rotation matrices are orthogonal, implies that the inverse coincides with the transposed, so Equation 5.8 can be rearranged as:

$$\vec{t}^{\,n} \vec{n}^{\,n\,T} = d \left(C_n^{c2\,T} K^{-1} H^* K C_{c1}^{n\,T} - I \right) \tag{5.9}$$

where, in order to eliminate the scale ambiguity [30], the homography matrix has been normalized with the ninth element of H as follows:

$$H^* = \frac{H}{H_{3,3}}$$

The distance to the ground d can be measured from a laser altimeter mounted on the helicopter. In case of flat ground, the differential barometric altitude measured from an on-board barometric sensor between the take-off and the current flight position, can be used. Since in our environment the terrain is considered to be flat then $\vec{n}^{\,n} = [0, 0, 1]$. Using RHS to indicate the right hand side of Equation 5.9, the north and east helicopter displacement are given by:

$$\begin{aligned} t_{north} &= RHS_{1,3} \\ t_{east} &= RHS_{2,3} \end{aligned} \tag{5.10}$$

5.6.1 Visual odometry error analysis

The error characteristics of the odometry displacement calculated from Equation 5.9 is evaluated through Monte Carlo simulations. It is assumed that the error distribution for the location of each point feature is Gaussian with zero mean and standard deviation σ_π, therefore the effect of outliers in the data is not represented in this analysis. In addition, a Gaussian error distribution with zero mean and standard deviation σ_α is assumed for the camera's attitude angles (pitch, roll, heading) represented in Equation 5.9 through the rotation matrices C^n_{c1} and C^{c2}_n. It can also be assumed that the ground altitude d is affected by Gaussian noise. However, the influence of such an error is neglected since it has low impact on the analysis. This is due to the fact that the camera is pointing mostly perpendicular downward and that the flight altitude is substantially larger than the measurement noise of the barometric sensor used.

A uniform distribution of n features is first generated in image 1. The same feature distribution is considered in image 2 so that the homography matrix transformation between the two images is the identity. Each feature is then perturbed with zero mean Gaussian error and standard deviation σ_π in each image. Subsequently, the normalized DLT algorithm (see appendix B) is applied to the noisy point correspondences and a homography matrix H is estimated.

The rotation matrices C^n_{c1} and C^{c2}_n are obtained from $C^c_n = C^c_g C^g_b C^b_n$ (see sections 3.4 and 5.4). A Gaussian error with zero mean and standard deviation σ_α is added to the pitch, roll, heading angles in the matrix C^b_n while C^c_g and C^g_b are taken as constant transformations. In reality, the camera platform used in this thesis is connected to the helicopter body through a set of dampers so that C^g_b should also be affected by random error (due to vibrations). In this analysis, the value used for the standard deviation σ_α takes into account both the UAV attitude error and the camera mounting error caused by the vibrations.

In the first analysis it will be assumed a uniform distribution of point features on the whole image. Subsequently, the same analysis is repeated using features distributed only on a limited part of the image.

1. Uniform feature distribution in the whole image.

A set of n feature points are generated in a 360x288 image using a uniform random distribution as shown in Figure 5.5.

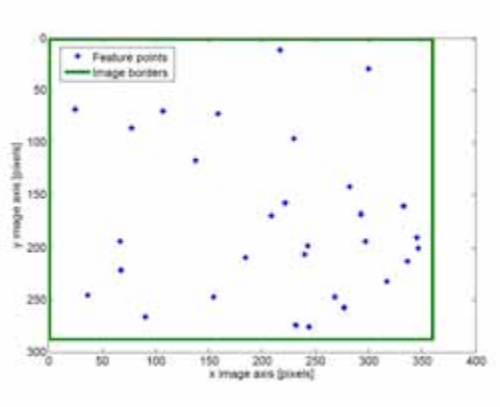


Figure 5.5: Example of a uniform random feature distribution in a 360x288 image.

After the features have been generated in the image, Equation 5.9 is evaluated N times adding Gaussian distributed zero mean noise at every iteration to the feature locations and attitude angles. The result is the odometry error sample distribution in the two directions t^i_{north} and t^i_{east}, computed from Equation 5.10. Since the features are uniformly distributed in the image, the sample distributions t^i_{north} and t^i_{east} will be similar in the two directions. In addition, from Equation 5.9 it can be observed that the ground altitude d can be considered as a final multiplicative factor to the analysis. Therefore, the results from the Monte Carlo simulation will be represented in terms of normalized standard deviation error in one of the direction, for example t^i_{north}:

$$\sigma_\tau = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t^i_{north} - \bar{t}_{north})^2} \tag{5.11}$$

where $\bar{t}_{north}$ indicates the mean of the distribution and N the number of samples. The term normalized standard deviation indicates that the computation of Equation 5.9 has been performed using the unitary depth value $d = 1$.

Figure 5.6 shows σ_τ for several pixel noise levels σ_π and for several numbers of point features used in the image. Each point of the plot has been extracted using $N = 1000$ iterations. To separate the effects of the noise relative to the point feature and to the attitude angles, the results displayed in Figure 5.6 have been obtained applying Gaussian noise only on the pixels but without noise on the attitude angles ($\sigma_\alpha = 0$). From the graph, one can observe how the output noise σ_τ increases with the increasing of the feature noise level and decreases with the increasing of the feature number per image. The slope of the graph tends to zero with the increasing of the feature number, which means that the advantage of increasing the feature number becomes less and less relevant.

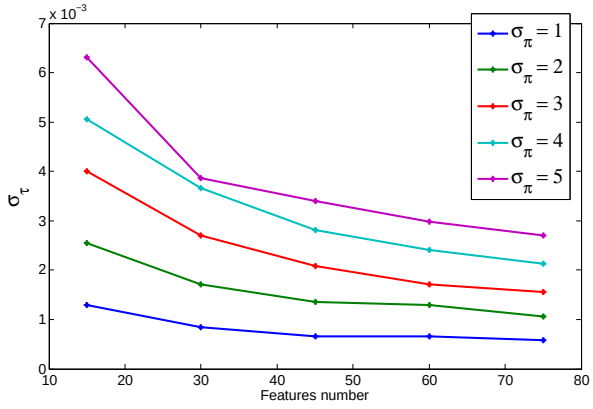


Figure 5.6: Effect of the feature noise level σ_π (in pixels) and the number of features used per image on the normalized odometry noise σ_τ.

Figure 5.7 shows the value of σ_τ for different noise levels on the at-

titude angles. In this case the pixels are not corrupted by noise and the homography matrix is the identity. The values of σ_τ due to the attitude noise appear to be larger than the values obtained from noisy point features location. Assuming $\sigma_\alpha = 0.5$ and $\sigma_\pi = 2$, the effect of σ_α on σ_τ is one order magnitude larger than the effect of σ_π, in other words the odometry is affected predominantly from the attitude noise.

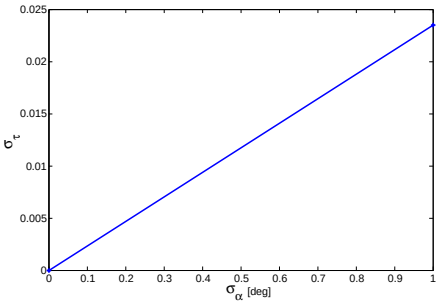


Figure 5.7: Effect of the attitude noise level σ_α on the normalized odometry noise σ_τ.

The final odometry noise characteristic can be extracted assuming the realistic standard deviation values of $\sigma_\pi = 2$ pixels for the feature noise, $\sigma_\alpha = 0.5$ degrees for the camera attitude angles and an average number of 50 features each image. Figure 5.8 shows the odometry noise characteristics function of the ground altitude (the normalization has been removed). The result represents the odometry standard deviation error calculated for one axis.

2. Non-uniform feature distribution in the image.

   This case analyzes the effects of having a non-uniform feature distribution in the image. An example of non-uniform feature distribution is represented in Figure 5.9 where the features are concentrated in the left part of the image.

   The results of the Monte Carlo analysis, as just described at the

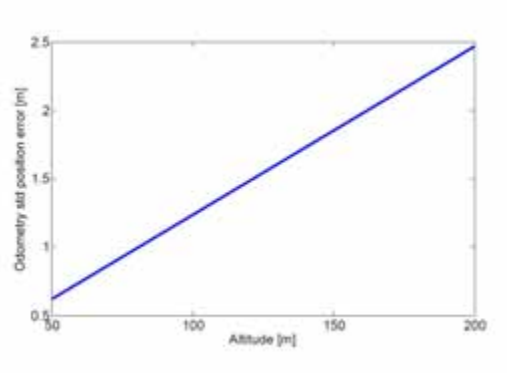


Figure 5.8: Odometry standard deviation noise (for one axis), function of the altitude. A standard deviation of $\sigma_\pi = 2$ pixel has been used for the feature noise and $\sigma_\alpha = 0.5$ for the camera attitude noise. 50 features have been used for each image.

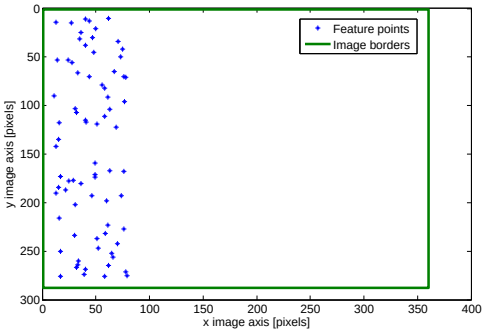


Figure 5.9: Example of a non-uniform feature distribution in a 360x288 image.

previous point, are represented in Figure 5.10. It can be observed how a non-uniform feature distribution leads to worse results compared to

a uniform distribution (Figure 5.6). It can also be observed that the non-symmetry of the odometry noise along the two image axes is as expected. The feature distribution used is similar to the one displayed in Figure 5.9 where the features lie on the left part of the image. As intuition suggests, the odometry behavior along the x image axis is worst than the y direction. It can also be observed that the behavior is unstable when using a small feature number.

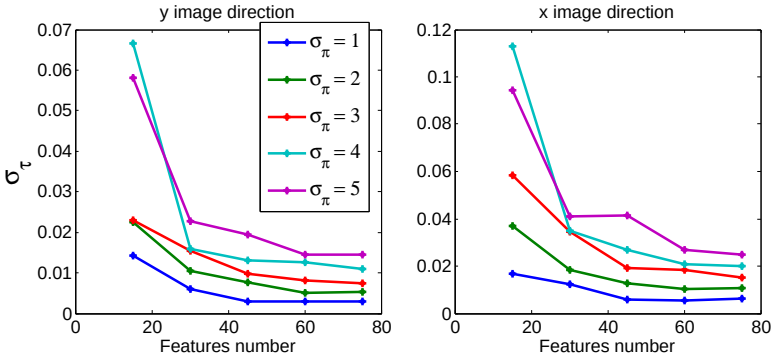


Figure 5.10: Effect of the feature noise level σ_π (in pixels) and the number of features used per image on the normalized odometry noise σ_τ. A non-uniform feature distribution in the image was used, therefore σ_τ is different along the two image axes direction.

Using the same noise level and number of features as for the previous analysis ($\sigma_\pi = 2$, $\sigma_\alpha = 0.5$ and $n = 50$), the final odometry noise characteristic can be plotted for the two image directions (Figure 5.11). The results are slightly worse than the previous case (Figure 5.8) but not as worse as one would expect thanks to the use of camera rotation information from the inertial sensors. This case would be much worse if external rotation information would not be used but had to be extracted from homography decomposition.

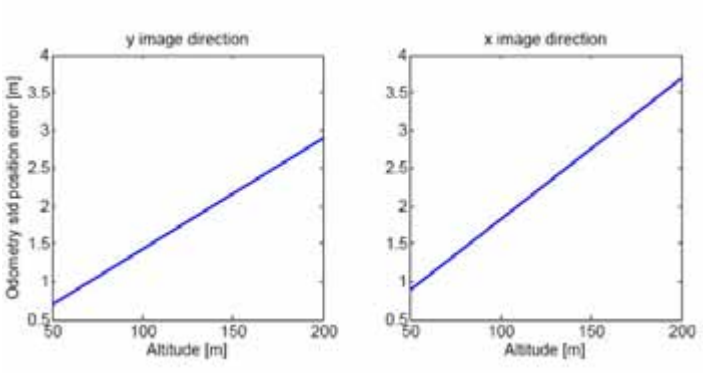


Figure 5.11: Odometry standard deviation noise (for both axes), function of the altitude. A standard deviation of $\sigma_\pi = 2$ pixel has been used for the feature noise and $\sigma_\alpha = 0.5$ for the camera attitude noise. 50 features non-uniformly distributed have been used for each image.

To conclude this section, some considerations about the planar scene assumption will be made. Even if the terrain is not sloped, altitude variations between the ground level and roof top or tree top level are still present in the real world. The planar scene assumption is widely used for airborne applications, however a simple calculation can give a rough idea of the error being introduced.

For a UAV in a level flight condition with the camera pointing perpendicularly downwards, the 1D pin-hole camera projection model can be used to make a simple calculation:

$$\Delta x_p = f \frac{\Delta x_c}{d} \tag{5.12}$$

where f is the camera focal length, d is the distance to the ground plane, Δx_p is the pixel displacement in the camera image of an observed feature and Δx_c the computed odometry camera displacement. If the observed feature is not on the ground plane but at a δd from the ground, the true

camera displacement Δx_c^t can be expressed as a function of the erroneous camera displacement Δx_c as:

$$\Delta x_c^t = \Delta x_c(1 - \frac{\delta d}{d}) \tag{5.13}$$

with $\epsilon = \frac{\delta d}{d}$ being the odometry error due to the depth variation. Then, if δd is the typical roof top height of an urban area, the higher the flight altitude d the smaller is the error ϵ. Considering an equal number of features picked on the real ground plane and on the roof tops, the reference homography plane can be considered as at average roof height over the ground and the odometry error can be divided by 2. If a UAV is flying at an altitude of 150 meters over an urban area with a typical roof height at 15 meters, the odometry error derived from neglecting the height variation is about 5%.

5.7 Image registration

Image registration is the process of overlaying two images of the same scene taken at different times, from different viewpoints and by different sensors. The registration geometrically aligns two images: the *reference* and *sensed* images. Image registration has been an active research field for many years and it has a wide range of applications. A literature review on image registration can be found in [8, 66]. In this context it is used to extract global position information for terrain relative navigation.

Image registration is performed with a sequence of image transformations, including rotation, scaling and translation which bring the sensed image to overlay precisely with the reference image. In this work, the reference and sensed images are aligned and scaled using the information provided by the KF. Once the images are aligned and scaled, the final match consists in finding a 2D translation which overlays the two images. The orientation between the sensed and reference images can also be found using image processing methods. An approach based on the Hough transform is proposed in [15]. In such a case the UAV heading information extracted from the image registration process could be used as a measurement to

update the KF. The approach proposed in [15] is not adopted here since it requires a structured environment (i.e. road network) in order to perform properly.

Two main approaches can be adopted to image registration: *correlation-based matching* and *pattern matching*. In correlation-based matching, the sensed image is placed at every pixel location in the reference image, then, a similarity criteria is adopted to decide which location gives the best fit. In pattern matching approaches on the other hand, salient features (or landmarks) are detected in both images and the registration is obtained by matching the set of features between the images. Both methods have advantages and disadvantages.

Methods based on correlation can be implemented very efficiently and are suitable for real-time applications. They can be applied also in areas with no distinct landmarks. However they are typically more sensitive to differences between the sensed and reference image than pattern matching approaches.

Methods based on pattern matching do not use image intensity values directly. The patterns are information on a higher level typically represented with geometrical models. This property makes such methods suitable for situations when the terrain presents distinct landmarks which are not affected by seasonal changes (i.e. roads, houses). If recognized, even a small landmark can make a large portion of terrain unique. This characteristic makes these methods quite dissimilar from correlation-based matching where small details in an image have low influence on the overall image similarity. On the other hand these methods work only if there are distinct landmarks in the terrain. In addition, a pattern detection algorithm is required before any matching method can be applied. A pattern matching approach which does not require geometrical models is the Scale Invariant Feature Transform (SIFT) method [37]. The reference and sensed images can be converted into feature vectors which can be compared for matching purposes. Knowledge about altitude and orientation of the camera relative to the terrain is not required for matching. Correlation methods are in general more efficient than SIFT because they do not require a search over image scale. In addition SIFT features do not have the capability to handle variation of illumination condition between reference and sensed images [62, 33].

This work makes use of a matching technique based on a correlation approach. The normalized cross-correlation of intensity images [49] is utilized. Before performing the cross-correlation, the sensed and reference images are scaled and aligned. The cross-correlation is the last step of the registration process and it provides a measure of similarity between the two images.

The image registration process is represented in the block diagram in Figure 5.13. First, the sensed color image is converted into gray-scale, then it is transformed to the same scale of the reference image. Scaling is performed converting the sensed image to the resolution of the reference image. The scale factor s is calculated using Equation 5.14:

$$\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \begin{pmatrix} \frac{1}{f_x} \\ \frac{1}{f_y} \end{pmatrix} d \cdot I_{res} \tag{5.14}$$

where d, as for the odometry, is the UAV ground altitude and I_{res} is the resolution of the reference image. The alignment is performed using the UAV heading estimated by the KF.

After the alignment and scaling steps, the cross-correlation algorithm is applied. If S is the sensed image and I is the reference image, the expression for the two-dimensional normalized cross-correlation is:

$$C(u,v) = \frac{\sum_{x,y}[S(x,y) - \mu_S][I(x-u, y-v) - \mu_I]}{\sqrt{\sum_{x,y}[S(x,y) - \mu_S]^2 \sum_{x,y}[I(x-u, y-v) - \mu_I]^2}} \tag{5.15}$$

where μ_S and μ_I are the average intensity values of the sensed and the reference image respectively. Figure 5.12 depicts a typical cross-correlation result between a sensed image taken from the Rmax helicopter and a restricted view of the reference image of the flight-test site.

Image registration is performed only on a restricted window of the reference image. The UAV position predicted by the filter is used as center of the restricted matching area. The purpose is to disregard low probability areas to increase the computational efficiency of the registration process. The window size depends on the position uncertainty estimated by the filter.

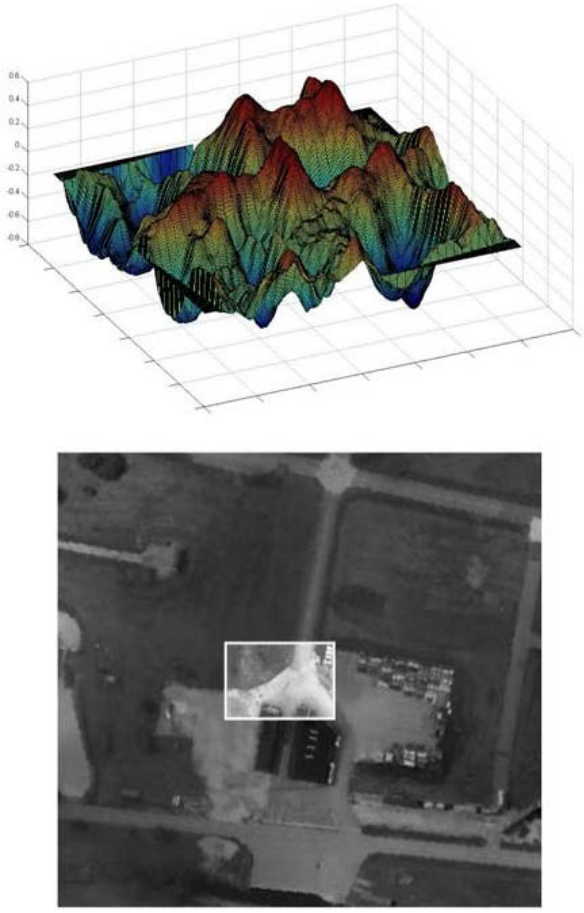


Figure 5.12: At the top of the figure is depicted the normalized cross-correlation result and, at the bottom, the sensed image (white square) matched to the reference image.

5.8 Sensor fusion algorithms

In this work, the UAV state is computed fusing inertial sensor data with vision data coming from the odometry and image registration modules. The

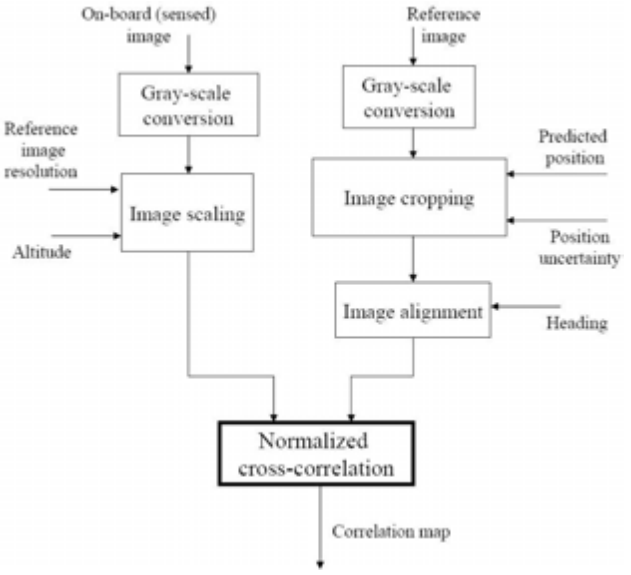


Figure 5.13: Image registration schematic.

state estimation problem has been split into two parts. The first part is represented by a standard Kalman filter (KF) which estimates the full UAV state (position, velocity and attitude), the second part is represented by a point-mass filter (PMF) which estimates the absolute 2D UAV position. The two filters are interconnected as shown in Figure 5.3.

Basics concepts on the probabilistic approach to filtering and on the Bayesian recursion equations will be given in the following two sections. A good and accessible reference on this topic is available in [60].

5.8.1 Basic concepts

In general it is impossible to know the state of a robotic system with infinite accuracy. In fact, when one tries to measure for example the position, a certain degree of uncertainty is always present due to unavoidable measurement errors. It is natural than to represent quantities such as the state

of a robot in probabilistic terms.

In this work, the Bayesian estimation framework is used to fuse data from different sensor sources and estimate the UAV state. Suppose that $\vec{x}$ is a state vector and $\vec{y}$ is a measurement vector which is used to measure $\vec{x}$, in the Bayesian estimation framework, such vectors are random variables and they are represented in terms of *probability density functions* (PDF) $p(\vec{x})$ and $p(\vec{y})$. The *conditional* or *posterior* PDF $p(\vec{x}|\vec{y})$ represents the PDF of the state vector $\vec{x}$ after the measurement $\vec{y}$ and represents the solution to the Bayesian filtering problem.

For on-line applications, $p(\vec{x}|\vec{y})$ must be computed recursively each time a new measurement update is available. If a dynamic model describing the time evolution of the system's states is available, it can be used in the estimation process to make a time prediction of the system state $p(\vec{x}_{t+1}|\vec{x}_t)$.

The Bayesian filtering problem aims at finding the posterior PDF $p(\vec{x}_t|\vec{y}_t)$ given the PDF for the state transition $p(\vec{x}_{t+1}|\vec{x}_t)$ and the measurement *likelihood* $p(\vec{y}_t|\vec{x}_t)$, which is the probability of measuring $\vec{y}$ from a given state $\vec{x}$. Once the posterior is known, an estimate for the state can be computed from it.

5.8.2 Bayesian filtering recursion

In general, the state transition PDF $p(\vec{x}_{t+1}|\vec{x}_t)$ and the measurement likelihood $p(\vec{y}_t|\vec{x}_t)$ are computed from a state-space model. A non-linear state-space model with additive process noise $\vec{v}$ and measurement noise $\vec{e}$ can be represented with the following form:

$$\begin{aligned} \dot{\vec{x}}_t &= f(\vec{x}_{t-1}, \vec{u}_{t-1}) + \vec{v} \\ \vec{y}_t &= h(\vec{x}_t, \vec{u}_t) + \vec{e} \end{aligned} \qquad t = 1, 2, ... \tag{5.16}$$

If the state vector $\vec{x}$ is n-dimensional, the recursive Bayesian filtering solution for such a model is:

$$p(\vec{x}_t|\vec{y}_{1:t-1}) = \int_{\Re^n} p_v(\vec{x}_t - f(\vec{x}_{t-1}, \vec{u}_{t-1}))p(\vec{x}_{t-1}|\vec{y}_{1:t-1})d\vec{x}_t \quad (5.17)$$

$$\alpha_t = \int_{\Re^n} p_e(\vec{y}_t - h(\vec{x}_t, \vec{u}_t))p(\vec{x}_t|\vec{y}_{1:t-1})d\vec{x}_t \quad (5.18)$$

$$p(\vec{x}_t|\vec{y}_{1:t}) = \alpha_t^{-1} p_e(\vec{y}_t - h(\vec{x}_t, \vec{u}_t))p(\vec{x}_t|\vec{y}_{1:t-1}) \quad (5.19)$$

The aim of recursion (5.17)-(5.19) is to estimate the posterior density $p(\vec{x}_t|\vec{y}_{1:t})$ of the filter. Equation 5.17 represents the time update while Equation 5.19 is the measurement update. p_v and p_e represent the probability density functions of the process and measurement noise respectively. The iterations must be initialized with $p(\vec{x}_1|\vec{y}_0) = p(\vec{x}_0)$. $p(\vec{x}_0)$ is also called the *prior*.

Once the posterior $p(\vec{x}_t|\vec{y}_{1:t})$ is known, the estimation of the minimum variance (MV) system state and its uncertainty are computed from Equations 5.20 and 5.21 respectively.

$$\hat{\vec{x}}_t^{MV} = \int_{\Re^n} \vec{x}_t p(\vec{x}_t|\vec{y}_{1:t})d\vec{x}_t \quad (5.20)$$

$$P_t = \int_{\Re^n} (\vec{x}_t - \hat{\vec{x}}_t^{MV})(\vec{x}_t - \hat{\vec{x}}_t^{MV})^T p(\vec{x}_t|\vec{y}_{1:t})d\vec{x}_t \quad (5.21)$$

In general, it is very hard to compute the analytical solution of the integrals in (5.17)-(5.21). A closed form solution exists only for special cases. If the state-space system (5.16) is linear, the noises $\vec{v}$ and $\vec{e}$ are Gaussian and the prior $p(\vec{x}_0)$ normally distributed, the closed form solution is represented by the popular Kalman filter. The Kalman filter represents PDFs in terms of mean and covariance and it is therefore addressed as a *parametric* solution. In case of non-linearity of the state-space model, an approximate solution can be obtained through a linearization of the model usually via Taylor expansion. After linearization the KF solution can be applied. This class of filters are addressed as *extended Kalman filters*.

If, besides the linearity, the Gaussian hypothesis is also not fulfilled, other methods must be employed to solve the Bayesian recursion. Several

methods have been developed for this purpose, the most popular belong to the class of *non-parametric* solutions. The *particle filter* (PF) is an example of a non-parametric solution in which the posterior is represented by a set of random state samples. The solution is approximate but can represent more general probability distributions than the Gaussian one. Another non-parametric solution which can be applied to non-linear and non-Gaussian systems is represented by the point-mass filter (PMF). The key idea is to solve the Bayesian recursion over a grid. The integrals in Equations (5.17)-(5.21) are discretized and approximated with finite sums over the grid. Particle filters and point-mass filters have the limitation that they are efficient only for low-dimension state vectors (in general up to dimension 2 for PMF and up to dimension 5 for PF). For higher dimensions, the computational complexity of the algorithms requires specialized hardware solutions for real-time applications.

The vision-based navigation problem addressed in this chapter is non-linear and non-Gaussian therefore a solution based only on a Kalman filter cannot be applied. The main source of non-linearity comes from the video camera measurements. The relation between a point in the image plane and the corresponding point in the real world is represented by Equation 5.2. Such a relation is non-linear due to the presence of the UAV's attitude angles (in C_n^b) and the depth d. In addition, the problem is non-Gaussian since the image registration likelihood does not have a Gaussian distribution (see top of Figure 5.12).

It appears that the use of the PMF or PF techniques might be appropriate for this case. However, since the state dimension used in this work is 12 (3-position, 3-velocity, 3-attitude and 3-accelerometer bias) a direct application of such techniques poses computational problems for on-line applications. In order to make PF techniques suitable for on-line applications even for larger state vectors, the research community has developed a *marginalized particle filter* (MPF) approach. The idea is to explore any linear Gaussian substructure in the system in a way that the Kalman filter can be applied to such substructure, while for the remaining part the PF is applied. Preliminary results of the MPF approach applied to UAV navigation can be found in [34].

In this thesis a different filtering solution has been explored. A standard 12-state error dynamics Kalman filter is used to fuse inertial data with an

absolute position sensor. The position sensor is a 2-state point-mass filter which fuses the visual odometry data with the absolute position information coming from the image registration module (Figure 5.3). A *certainty equivalence approximation* has been used in the scheme since the PMF uses the estimated attitude information from the Kalman filter as deterministic values. The approach has shown excellent results as will be shown later.

5.8.3 Kalman filter

The KF implemented is based on the standard structure in airborne navigation systems and is used to estimate the error states from an integrated navigation system. The KF implementation, together with the INS mechanization is described in detail in appendix A.

The linear state-space error dynamic model used in the KF is derived from a perturbation analysis of the motion equations [7] and is represented with the model (5.22)-(5.23) where system (5.22) represents the process model while system (5.23) represents the measurement model.

$$\begin{pmatrix} \delta\dot{\vec{r}}^{\,n} \\ \delta\dot{\vec{v}}^{\,n} \\ \dot{\vec{\epsilon}}^{\,n} \end{pmatrix} = \begin{bmatrix} \mathbf{F_{rr}} & \mathbf{F_{rv}} & \mathbf{0} \\ & & 0 \quad -a_d \quad a_e \\ \mathbf{F_{vr}} & \mathbf{F_{vv}} & a_d \quad 0 \quad -a_n \\ & & -a_e \quad a_n \quad 0 \\ \mathbf{F_{er}} & \mathbf{F_{ev}} & -(\vec{\omega}_{in}^{\,n}\times) \end{bmatrix} \begin{pmatrix} \delta\vec{r}^{\,n} \\ \delta\vec{v}^{\,n} \\ \vec{\epsilon}^{\,n} \end{pmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{C_b^n} & \mathbf{0} \\ \mathbf{0} & -\mathbf{C_b^n} \end{bmatrix} \vec{u} \tag{5.22}$$

$$\begin{pmatrix} \varphi_{ins} - \varphi_{vision} \\ \lambda_{ins} - \lambda_{vision} \\ h_{ins} - (\Delta h_{baro} + h_0) \end{pmatrix} = \begin{bmatrix} \mathbf{I\,0\,0} \end{bmatrix} \begin{pmatrix} \delta\vec{r}^{\,n} \\ \delta\vec{v}^{\,n} \\ \vec{\epsilon}^{\,n} \end{pmatrix} + \vec{e} \tag{5.23}$$

with:

$\delta\vec{r}^{\,n}$ = $(\delta\varphi\ \delta\lambda\ \delta h)^T$ = latitude, longitude and altitude error states;

$\delta\vec{v}^{\,n}$ = $(\delta v_n\ \delta v_e\ \delta v_d)^T$ = north, east and down velocity error states;

$\vec{\epsilon}^{\,n}$ = $(\delta\phi\ \delta\theta\ \delta\psi)^T$ = pitch, roll and heading error states;

$\vec{u}$ = $(\delta\vec{f}^{\,T}_{acc}\ \delta\vec{\omega}^{\,T}_{gyro})$ = accelerometers and gyros system noise;

$\vec{e}$ = represents the measurement noise;

$\vec{\omega}^{\,n}_{in}$ = rotation rate of the navigation frame;

a_n, a_e, a_d = north, east and vertical acceleration components;

$\mathbf{F_{xx}}$ = elements of the system's dynamics matrix;

The KF implemented uses 12 states including 3 accelerometer biases as mentioned before. However, the state-space model (5.22)-(5.23) uses only 9 states without accelerometer biases. The reason for using a reduced representation is to allow the reader to focus only on the relevant part of the model for this section. The accelerometer biases are modeled as first order Markov processes. The gyro biases can be modeled in the same way but they were not used in this implementation. The acceleration elements a_n, a_e and a_d are left in the explicit form in the system (5.22) as they will be needed for further discussions.

It can be observed how the measurements coming from the vision system in the form of latitude and longitude (φ_{vision}, λ_{vision}) are used to update the KF. The altitude measurement update is done using the barometric altitude information from an on-board pressure sensor. In order to compute the altitude in the WGS84 reference system required to update the vertical channel, an absolute reference altitude measurement h_0 needs to be known. For example if h_0 is taken at the take-off position the barometric altitude variation Δh_{baro} relative to h_0 can be obtained from the pressure sensor and the absolute altitude can finally be computed. This technique works if the environmental static pressure remains constant. The state-space system, as represented in (5.22)-(5.23) is fully observable. The elements of the matrix $\mathbf{F_{er}}$, $\mathbf{F_{ev}}$ and $(\vec{\omega}^{\,n}_{in}\times)$ are quite small as they depend on the Earth's rotation rate and the rotation rate of the navigation frame

due to the Earth's curvature. These elements are influential in the case of a very sensitive IMU or high speed flight conditions. For the flight conditions and typical IMU used in a small UAV helicopter, such elements are negligible, therefore observability issues might arise for the attitude angles. For example in case of hovering flight conditions or, more generally, in case of non-accelerated flight conditions, the elements a_n and a_e are zero. It can also be observed that the heading angle error $\delta\psi$ (third component of $\vec{\epsilon}^n$) is not observable. Therefore, for helicopters that are supposed to maintain the hovering flight condition for an extended period of time, an external heading aiding (i.e. compass) is required. On the other hand, the pitch and roll angle are observable since the vertical acceleration a_d is usually different from zero (except in case of free fall but this is an extreme case).

It should be mentioned that the vision system uses the attitude angles estimated by the KF for the computation of the absolute position, as can be observed in Figure 5.3 (sub-system 2), where this can be interpreted as a linearization of the measurement update equation of the KF. This fact might have implications on the estimation of the attitude angles since an information loop is formed in the scheme. The issue will be discussed later in section 5.8.5.

An alternative solution is to estimate the attitude angles independently using the assumption of non-accelerated flight condition. In this case the direction of the gravity vector can be measured by the IMU and consequently the attitude angles can be estimated. In any case the attitude estimation fails in situations where the vehicle flies in a persistent accelerated flight conditions. For this reason, this work does not rely on this assumption.

5.8.4 Point-mass filter

The point-mass filter (PMF) is used to fuse measurements coming from the visual odometry system and the image matching system. The PMF computes the solution to the Bayesian filtering problem on a discretized grid [9]. In [6] such a technique was applied to a terrain navigation problem where a digital terrain elevation model was used instead of a digital 2D image as in the case presented here. The PMF is particularly suitable for this kind of problem since it handles general probability distributions and

non-linear models.

The problem can be represented with the following state-space model:

$$\vec{x}_t = \vec{x}_{t-1} + \vec{u}_{t-1}^{odom} + \vec{v} \tag{5.24}$$

$$p(\vec{y}_t|\vec{x}_t, \vec{z}_t) \tag{5.25}$$

Equation 5.24 represents the process model where $\vec{x}$ is the two-dimensional state (north-east), $\vec{v}$ the process noise and $\vec{u}^{odom}$ is the position displacement between time $t-1$ and t computed from the visual odometry ($\vec{u}^{odom}$ was indicated with $\vec{t}$ in section 5.6). For the process noise $\vec{v}$, a zero mean Gaussian distribution has been used and a standard deviation value according to Figure 5.8.

The observation model is represented by the PDF (5.25) and represents the probability of measuring $\vec{y}_t$ given the state vector $\vec{x}_t$ and $\vec{z}_t$. The latter is a parameter vector given by $\vec{z}_t = (\hat{\phi}\ \hat{\theta}\ \hat{\psi}\ d)$. The first three components are the estimated attitude angles from the Kalman filter while d is the ground altitude measured from the barometric sensor. A certainty equivalence approximation has been used here since the components of vector $\vec{z}_t$ are taken as deterministic values. The measurement PDF (5.25) is computed from the cross-correlation (5.15) and its distribution is non-Gaussian. The distribution given by the cross-correlation (5.15) represents the correlation of the on-board camera view with the reference image. In general, there is an offset between the position matched on the reference image and the UAV position due to the camera view angle. The offset must be removed in order to use the cross-correlation as probability distribution for the UAV position. The attitude angles and ground altitude are used for this purpose.

As discussed earlier, in the PMF approximation the continuous state-space is discretized over a two-dimensional limited size grid so the integrals are replaced with finite sums over the grid points. The grid used in this application is uniform with N number of points and resolution δ. The

Bayesian filtering recursion (5.17)-(5.21) can be approximated as follows:

$$p(\vec{x}_t(k)|\vec{y}_{1:t-1}) = \sum_{n=1}^{N} p_v(\vec{x}_t(k) - \vec{u}_{t-1}^{odom} - \vec{x}_{t-1}(n)) p(\vec{x}_{t-1}(n)|\vec{y}_{1:t-1}) \delta^2 \quad (5.26)$$

$$\alpha_t = \sum_{n=1}^{N} p(\vec{y}_t(n)|\vec{x}_t(n), \vec{z}_t) p(\vec{x}_t(n)|\vec{y}_{1:t-1}) \delta^2 \quad (5.27)$$

$$p(\vec{x}_t(k)|\vec{y}_{1:t}) = \alpha_t^{-1} p(\vec{y}_t(n)|\vec{x}_t(n), \vec{z}_t) p(\vec{x}_t(k)|\vec{y}_{1:t-1}) \quad (5.28)$$

$$\hat{\vec{x}}_t^{MV} = \sum_{n=1}^{N} \vec{x}_t(n) p(\vec{x}_t(n)|\vec{y}_{1:t}) \delta^2 \quad (5.29)$$

$$P_t = \sum_{n=1}^{N} (\vec{x}_t(n) - \hat{\vec{x}}_t^{MV})(\vec{x}_t(n) - \hat{\vec{x}}_t^{MV})^T p(\vec{x}_t(n)|\vec{y}_{1:t}) \delta^2 \quad (5.30)$$

Before computing the time update (5.26) the grid points are translated according to the displacement calculated from the odometry:

$$\vec{x}_t(k) = \vec{x}_{t-1}(k) + \vec{u}_{t-1}^{odom} \qquad k = 1, 2, ..., N \quad (5.31)$$

The convolution (5.26) requires N^2 operations and computationally is the most expensive operation of the recursion. In any case, due to the problem structure, the convolution has separable kernels. For this class of two-dimensional convolutions, there exist efficient algorithms which reduce the computational load. More details on this problem can be found in [6].

Figure 5.14 depicts the evolution of the filter PDF. From the prior (t_0), the PDF evolves developing several peaks. The images and flight data used were collected during a flight-test campaign and for the calculation a 200x200 grid of 1 meter resolution was used.

The interface between the KF and the PMF is realized by passing the latitude and longitude values computed from the PMF to the KF measurement model (Equation 5.23). The PMF position covariance is computed with Equation 5.30 and it could be used in the KF for the measurement update. However, the choice of an empirically determined covariance for the KF measurement update has been preferred. The constant value of

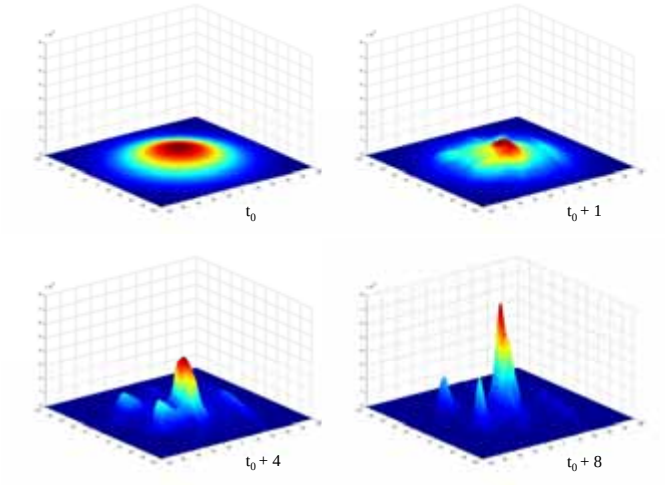


Figure 5.14: Evolution of the filter's probability density function (PDF). The capability of the filter to maintain the full probability distribution over the grid space can be observed.

2 meters for the horizontal position uncertainty and 4 meters for the vertical position uncertainty (derived from the barometric altitude sensor) have been used.

5.8.5 Stability analysis from Monte Carlo simulations

In this section, the implications of using the attitude angles estimated by the KF to compute the vision measurements will be analyzed. As mentioned before, since the vision measurements are used to update the KF, an information loop is created which could limit the performance and stability of the filtering scheme. First, the issue will be analyzed using a simplified dynamic model. Then, the complete KF in closed loop with the odometry will be tested using Monte Carlo simulations.

The velocity error dynamic model implemented in the KF is derived

from a perturbation of the following velocity dynamic model [52]:

$$\dot{\vec{v}}^{\,n} = C_b^n \vec{a}^{\,b} - (2\vec{\omega}_{ie}^{\,n} + \vec{\omega}_{en}^{\,n}) \times \vec{v}^{\,n} + \vec{g}^{\,n} \tag{5.32}$$

with $\vec{a}^{\,b}$ representing the accelerometers output, $\vec{\omega}_{ie}^{\,n}$ and $\vec{\omega}_{en}^{\,n}$ the Earth rotation rate and the navigation rotation rates which are negligible, as discussed in section 5.5, and $\vec{g}^{\,n}$ the gravity vector. Let us analyze the problem for the 1D case. The simplified velocity dynamics for the x^n axis is:

$$\dot{v}_x^n = a_x^b \cos\theta + a_z^b \sin\theta$$

where θ represents the pitch angle. Suppose that the helicopter is in hovering but the estimated pitch angle begins to drift due to the gyro error. An apparent velocity will be observed due to the use of the attitude angles in the measurement equation from the vision system. Linearizing the model around the hovering condition results in the following velocity dynamics and observation equations:

$$\dot{v}_x^n = a_x^b + a_z^b \theta \tag{5.33}$$
$$h\,\dot{\theta} = v_x^n \tag{5.34}$$

where h is the ground altitude measured by the barometric sensor. The coupling between the measurement equation (5.34) with the dynamic equation (5.33) can be observed (θ is in fact used to disambiguate the translation from the rotation in the measurement equation). Substituting Equation 5.34 in Equation 5.33 results in:

$$h\ddot{\theta} = a_x^b + a_z^b \theta \tag{5.35}$$

It can be observed that Equation 5.35 is a second order differential equation similar to the equation for a spring-mass dynamic system (a_z^b has a negative value) where the altitude h plays the role of the mass. One should expect that the error introduced by the information loop has an

oscillatory behavior which changes with the change in altitude. Since this is an extremely simplified representation of the problem, the behavior of the Kalman filter in closed loop with the odometry was tested through Monte Carlo simulations.

Gaussian distributed accelerometer and gyro data were generated together with randomly distributed features in the images. Since the analysis has the purpose of finding the effects of the attitude coupling problem, the following procedure was applied. First, Kalman filter results were obtained using the noisy inertial data and constant position measurement update. Than, the KF was fed with the same inertial data as the previous case but the measurement update came from the odometry instead (the features in the image were not corrupted by noise since the purpose was to isolate the coupling effect).

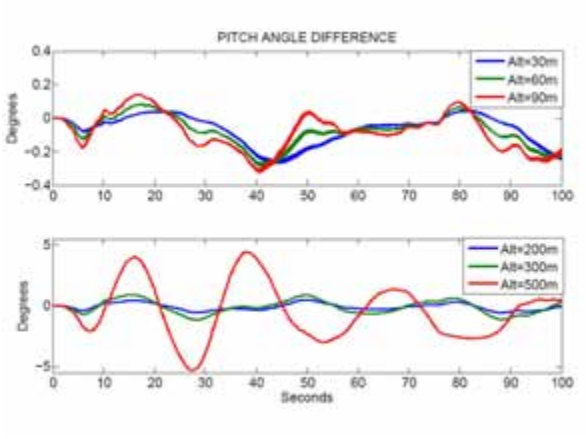


Figure 5.15: Difference between the pitch angle estimated by the KF with a constant position update and the KF updated using odometry position. It can be noticed how, the increase in the flight altitude, makes the oscillatory behavior of the system more evident with an increase in amplitude.

The KF results for the first case (constant update) were considered as the reference, then the result for the second case was compared to the ref-

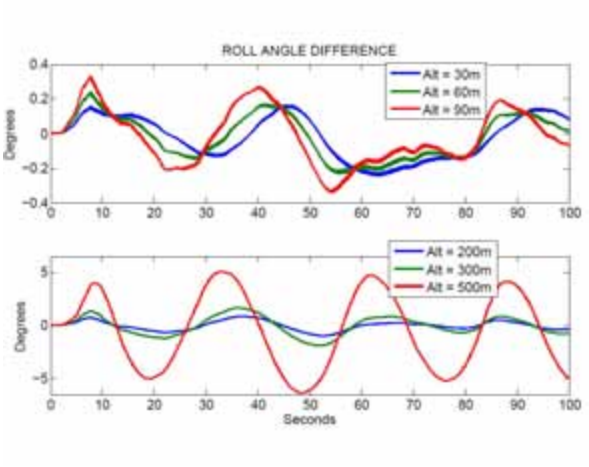


Figure 5.16: Difference between the roll angle estimated by the KF with a constant position update and the KF updated using odometry position. It can be noticed how, the increase in the flight altitude, makes the oscillatory behavior of the system more evident with an increase in amplitude.

erence case. What it is shown in the plots is the difference of the estimated pitch angle (Figure 5.15) and roll angle (Figure 5.16) between the two cases at different altitudes (obviously the altitude influences only the results of the closed loop case).

For altitudes up to 100 meters the difference between the two filter configurations is less than 0.5 degree. However, the increase in the flight altitude makes the oscillatory behavior of the system more evident with an increase in amplitude (as expected from the previous simplified analysis). A divergent oscillatory behavior was observed from an altitude of about 700 meters.

This analysis shows that the updating method used in this work introduces an oscillatory dynamics in the filter state estimation. The effect of such dynamics has a low impact for altitudes below 100 meters while becomes more severe for larger altitudes.

5.9 Experimental results

In this section, the performance of the vision-based state estimation approach described will be analyzed. The proposed filter architecture has been tested on real flight-test data and on-board the autonomous UAV helicopter described in chapter 2. Experimental evaluation based on off-line real flight data as well as on-line on-board test results are presented. The flight-tests were performed in a rural area of the Swedish country side (the reference images of the area were downloaded from Google Earth) but mainly in an emergency services training area in the south of Sweden. The reference image of this area is an ortho-rectified aerial image of 1 meter/pixel resolution with a sub-meter position accuracy (Figure 1.1).

The video camera sensor is a standard CCD analog camera with approximately 45 degrees horizontal angle of view. The camera frame rate is 25Hz and the images are reduced to half resolution (384x288 pixels) at the beginning of the image processing pipeline to reduce the computational burden. During the experiments, the video camera was looking downwards and fixed with the helicopter body. The PMF recursion was computed in all experiments on a 80x80 meters grid of one meter resolution. The IMU used is provided by the Yamaha Motor Company and integrated in the Rmax platform. Table 5.1 provides available specification of the sensors used in the experiment.

Sensor	Output Rate	Resolution	Bias
Accelerometers	66 Hz	1 mG	13mg
Gyros	200 Hz	0.1 deg/s	< 0.1deg/s
Barometer	40Hz	0.1 m	-
Vision	25 Hz	384x288 pixels	-

Table 5.1: Available characteristics of the sensor used in the navigation algorithm.

The results of the vision-based navigation algorithm are compared to the navigation solution given by an on-board INS/GPS KF running on-line. The KF fuses the inertial sensors with GPS position data and provides the full helicopter state estimate. The position data used as reference are

provided by a real-time kinematic GPS receiver with a sub-meter position accuracy.

5.9.1 Performance evaluation using off-line flight data

Sensor data and on-board video were recorded during an autonomous flight. The helicopter autonomously flew a pre-planned path using a path following functionality implemented in the software architecture. The video was recorded on-board and synchronized with the sensor data. The synchronization is performed by automatically turning on a light diode when the sensor data begins to be recorded. The light diode is visible in the camera frame. The video is recorded on tape using an on-board video recorder and the synchronization with the sensor data is done manually off-line. The video sequence is recorded at 25Hz frame rate. For the experiment described here, the video frames were sampled at 4Hz. The on-board sensor data was recorded at different sample rates.

The idea behind this experiment is to show the capability of the UAV to fly back to home-base once the GPS is lost. A GPS failure is simulated by disconnecting the GPS from the navigation filter. Subsequently the vision-based filter is engaged with the initial UAV state taken from the last GPS reading. The position update to the KF is then provided by the vision system. The inertial data is sampled and integrated at a 50Hz rate, while the vision system provides a position update rate at 4Hz.

Figure 5.17 shows the UAV flight path reconstructed using the navigation approach described in this work without using the GPS. The helicopter flew a closed loop path of about 1 kilometer length at 60 meters constant altitude above the ground. The left picture of Figure 5.17 presents a comparison between the helicopter path computed by the vision-based system (PMF output) with the GPS reference. The vision-based position is always close to the GPS position indicating a satisfactory localization performance. The odometry position results are also displayed in the left picture. The right picture of Figure 5.17 shows the position uncertainty along the path estimated from the PMF (Equation 5.30) and graphically represented with ellipses. The ellipse axes are oriented along the north-east directions (the uncertainty is in fact estimated along such axes by the PMF). The uncertainty would be better represented with ellipses oriented along the direction

of maximum and minimum uncertainty. In any case it is interesting to notice that above a road segment the uncertainty increases along the road and decreases along the direction perpendicular to the road. This can be noticed along the first path's leg (between points 1 and 2) and at point 4 of the path. This fact is a consequence of the impossibility for the image registration module to find the proper matching location along the road direction. It can also be noticed how the uncertainty increases when the helicopter passes above the pond (leg 3-4). This is a sign that this area does not have good properties for image registration. The position uncertainty was initialized to 25 meters at the beginning of the path (point 1).

The pictures in the top row of Figure 5.18 shows the north and east vision-based KF velocity estimation while the bottom right picture shows the horizontal velocity error magnitude. The bottom left picture shows the horizontal position error magnitude of the PMF, visual odometry and stand alone inertial solution. It can be observed how the PMF position error is always below 8 meters during the flight while the odometry accumulates 25 meters error in about 1 km of flight path length. The inertial solution has a pronounced drift showing the low performance characteristics of the IMU used and giving an idea of the improvement introduced just by the odometry alone.

Figure 5.19 shows the attitude angles estimated by the KF (left column) and the attitude angle errors (right column). From the pitch and roll error plots, does not appear to be any tendency to drift during the flight. The heading error plot shows instead a tendency to drift away. The low acceleration of the helicopter during the flight experiment makes the heading angle weakly observable (section 5.5), therefore the use of external heading information (i.e. compass) is required in the KF in order to allow for a robust heading estimate.

In Figure 5.20 the influence of the number of features tracked and the effects of the RANSAC algorithm are analyzed. The plots in the left column are relative to the case of the KLT tracking a maximum number of 50 features per frame while the right column shows the case of KLT tracking a maximum number of 150 features. When the maximum number of features is set, the tracking algorithm tries to track them but usually, a number of features will be lost each frame so the number of features tracked are less than the maximum number. In addition, when the RANSAC algorithm is

running, an additional number of features are discarded because they are considered outliers. In the top left plot, the odometry error is shown in the cases with and without the use of RANSAC and with a maximum of 50 features tracked. It can be observed how the use of RANSAC does not improve the odometry drift rate but it helps to filter out the position jumps. It has been observed that the jumps occur during sudden helicopter attitude variation resulting in a large feature displacement in the image. Such a correlation can be observed comparing the pitch plot with the odometry error plot. As the images were sampled at a 4Hz rate, the problem could be mitigated by increasing the feature tracking frame rate. The figure also shows the number of features tracked and discarded from the RANSAC algorithm. In the top right plot the same test is done with a maximum of 150 features tracked in each frame. If the top left and top right plots are compared, it can be observed that increasing the number of features tracked produces a worse result (comparing the blue plots without RANSAC). This effect is somehow predictable as the KLT selects the feature quality in a decreasing order. Comparing the two cases with RANSAC (red plots) there are basically no relevant differences meaning that the RANSAC detects the great part of outliers introduced by tracking a larger number of features (compare the two plots at the bottom of Figure 5.20). The fact that the cases without RANSAC finish with a lower position error does not have any relevance as it is mostly a random effect. From these considerations the solution with 50 features and RANSAC is preferable.

Figure 5.21 and Figure 5.22 show the results based on the same flight data set but the vision-based navigation system has been initialized with 5 degrees error in pitch, roll and heading angles. It can be observed in Figure 5.22 how the pitch and roll converge rapidly to the right estimate while the heading was between 5 to 10 degrees off during the whole test. It is interesting to notice from Figure 5.21 the effects of the off-heading condition. The odometry is rotated by approximately the heading angle error. The PFM solution is degraded due to the increased odometry error and due to the fact that there is an image misalignment error between the sensed and reference images. Despite the image misalignment, the PMF position estimate closes the loop at the right location (point 5) thanks to the robustness of the image cross-correlation in respect to small image misalignment. It appears that the image correlation was especially robust around corner

4 of the path. This fact is verified by the sudden decrease of the position uncertainty which can be observed in the left plot of Figure 5.21.

5.9.2 Real-time on-board flight-test results

Flight-test results of the vision-based state estimation algorithm implemented on-board the Rmax helicopter platform will be presented here. The complete navigation architecture is implemented on two on-board computers in the C language. The Sub-system 1 (Figure 5.3) is implemented on the primary flight computer PFC (PC104 PentiumIII 700MHz) and runs at a 50Hz rate. The Sub-system 2 is implemented on the image processing computer IPC (also a PC104 PentiumIII 700MHz) and runs at about 7Hz rate. The image processing is implemented using the Intel OpenCV library. The real-time data communication between the two computers is realized using a serial line RS232C. The data sent from the PFC to the IPC is the full vision-based KF state while the data sent from the IPC to the PFC is the latitude and longitude position estimated by the PMF and used to update the KF.

The on-board flight-test results are displayed in Figure 5.23, Figure 5.24 and Figure 5.25. The odometry ran with a maximum number of 50 features tracked in the image without RANSAC as it was not implemented on-board at the time of the experiment. The helicopter was manually put in hovering mode at an altitude of about 55 meters above the ground (position 1 of Figure 5.23). Then the vision-based navigation algorithm was initialized from the ground station. Subsequently the helicopter from manual was switched into autonomous flight with the control system taking the helicopter state from the vision-based KF. The helicopter was commanded to flight from position 1 to position 2 along a straight line path. Observe that during the path segment 1-2, the vision-based solution (red line) resembles a straight line as the control system was controlling using the vision-based data. The real path taken by the helicopter is instead the blue one (GPS reference). The helicopter was then commanded to fly a spline path (segment 2-3). With the helicopter at the hovering point 3 the autonomous flight was aborted and the helicopter taken into manual flight. The reason the autonomous flight was aborted was that in order for the helicopter to exit the hovering condition and enter a new path segment

the hovering stable condition must be reached. This is a safety check programmed in a low-level state machine which coordinates the flight mode switching. The hovering stable condition is fulfilled when the helicopter speed is less then 1 m/s. As can be seen from the velocity error plot in Figure 5.24 this is a rather strict requirement which is at the border of the vision-based velocity estimation accuracy. In any case, the vision-based algorithm was left running on-board while the helicopter was flown manually until position 4. The vision-based solution was always rather close to the GPS position during the whole path. Even the on-board solution confirms what was observed during the off-line tests for the attitude angles (Figure 5.25) with a stable pitch and roll estimate and a non stable heading estimate. Considering the error in the heading estimate the PMF position results (Figure 5.23) confirm a certain degree of robustness of the image matching algorithm with respect to the image misalignment error.

5.10 Conclusions

The experimental results of this work confirm the validity of the approach proposed. A vision-based sensor fusion architecture which can cope with short and long term GPS outages has been proposed and tested on-board an unmanned helicopter. The main contribution of this work can be summarized in the following points.

- Exploring the possibility of using one video camera both as a velocity meter (odometry) and a positioning device (image registration). We believe that this is a very practical and innovative concept.
- Development of a sensor fusion architecture which combines vision-based information together with inertial information in an original way. The envelope of the method proposed has been explored using Monte Carlo simulations. The main findings are the degradation of the performance when increasing the flight altitude.
- Real-time implementation and experimental results in field trials of the approach proposed on the Yamaha Rmax unmanned helicopter.

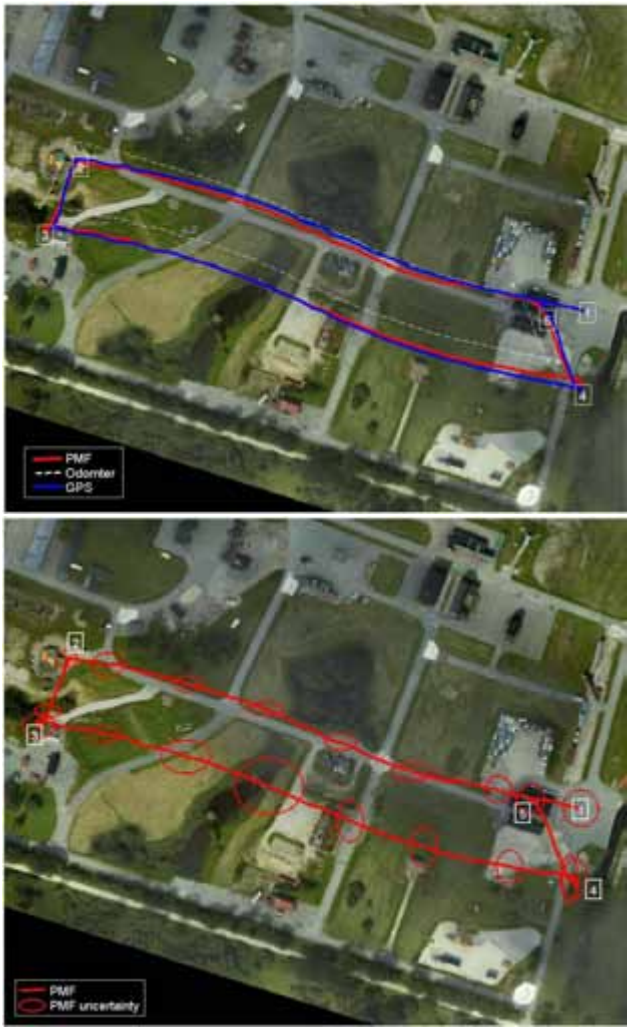


Figure 5.17: On the top is displayed the comparison between the flight path computed with the point-mass filter (red), the odometry (dashed white) and the GPS reference (blue). On the bottom, is shown the PMF result with the position uncertainty represented with ellipses along the path.

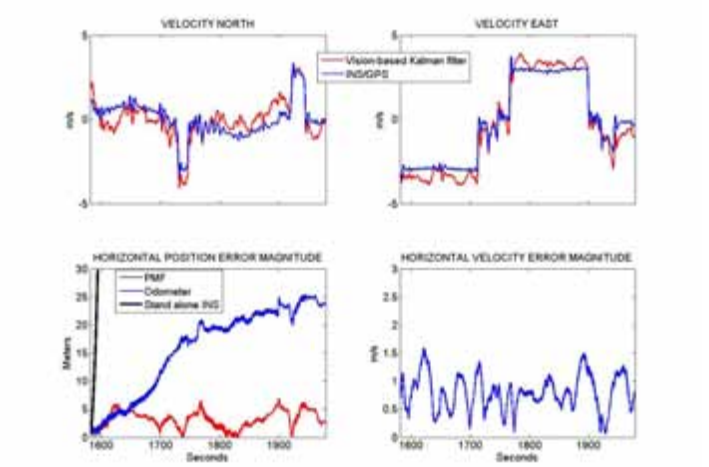


Figure 5.18: The top row depicts the north and east velocity components estimated by the Kalman filter while the bottom right picture depicts the horizontal velocity error magnitude. At the bottom left, the position error comparison between PMF, odometry and stand alone INS is given.

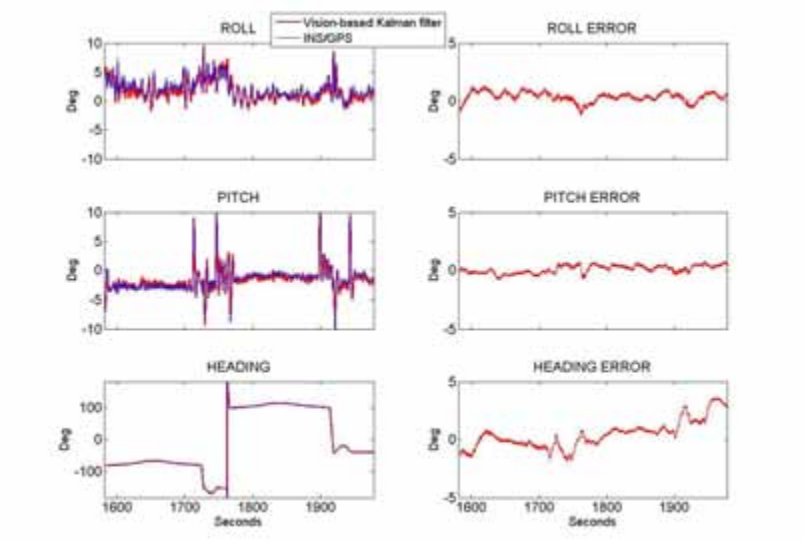


Figure 5.19: Estimated Kalman filter attitude angles with error plots.

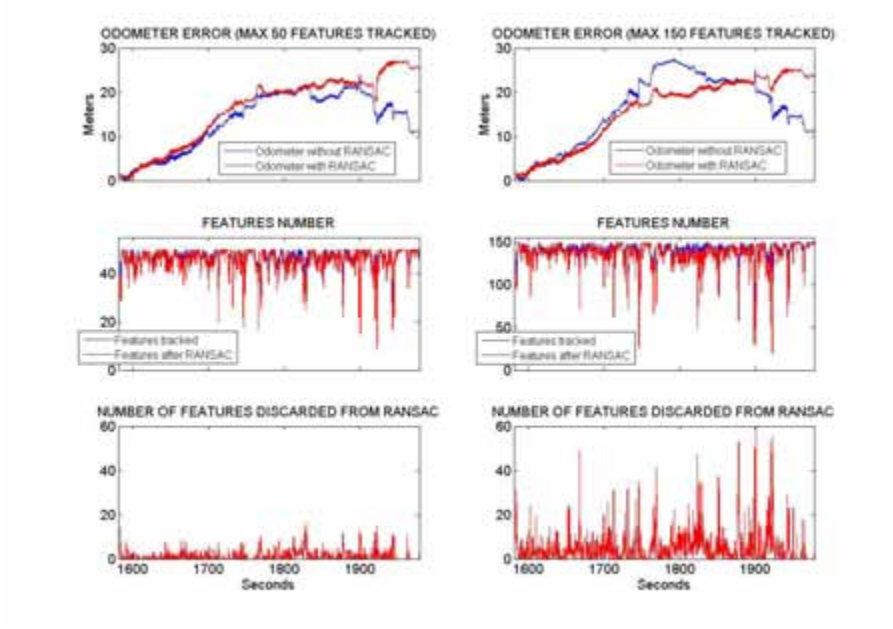


Figure 5.20: Comparison between odometry calculation using at most 50 features (left column) and at most 150 features (right column). The comparison also shows the effects of the RANSAC algorithm on the odometry error.

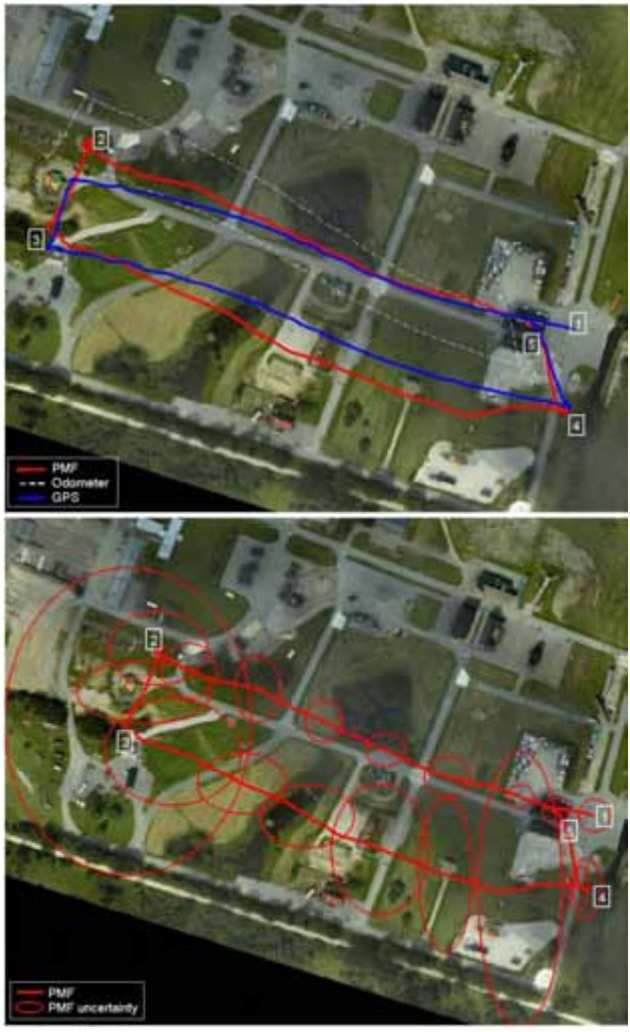


Figure 5.21: Vision-based navigation algorithm results in off-heading error conditions. The navigation filter is initialized with a 5 degree error in pitch, roll and heading. While the pitch and roll angle converge to the right estimate rapidly, the heading does not converge to the right estimate affecting the odometry and the image cross-correlation performed in off-heading conditions.

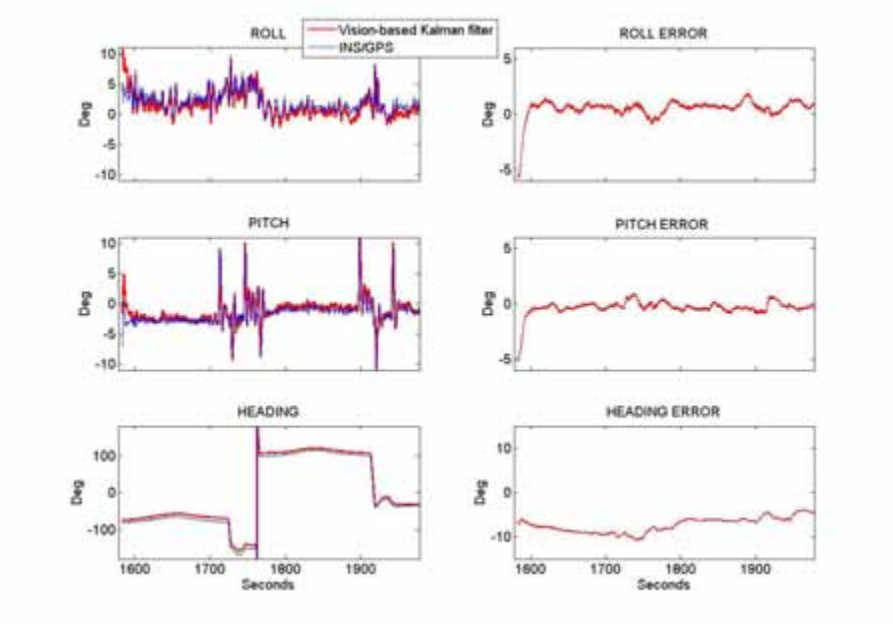


Figure 5.22: Estimated Kalman filter attitude angles with an initial 5 degrees error added for roll, pitch and heading angles. It can be observed how the pitch and roll angles converge quite rapidly, while the heading shows a weak converge tendency.

Figure 5.23: Real-time on-board position estimation results. The comparison between the flight path computed with the PMF (red) and the GPS reference (blue) is displayed.

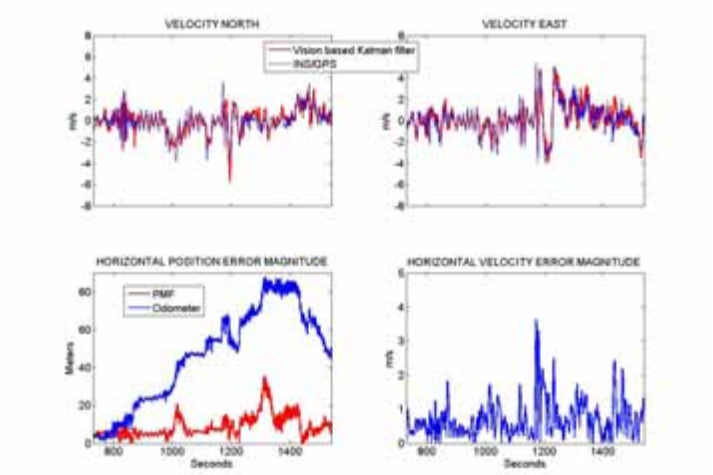


Figure 5.24: Real-time on-board results. In the top row are shown the north and east velocity components estimated by the Kalman filter while the bottom right picture displays the horizontal velocity error magnitude. At the bottom left, the position error comparison between PMF and odometry is given.

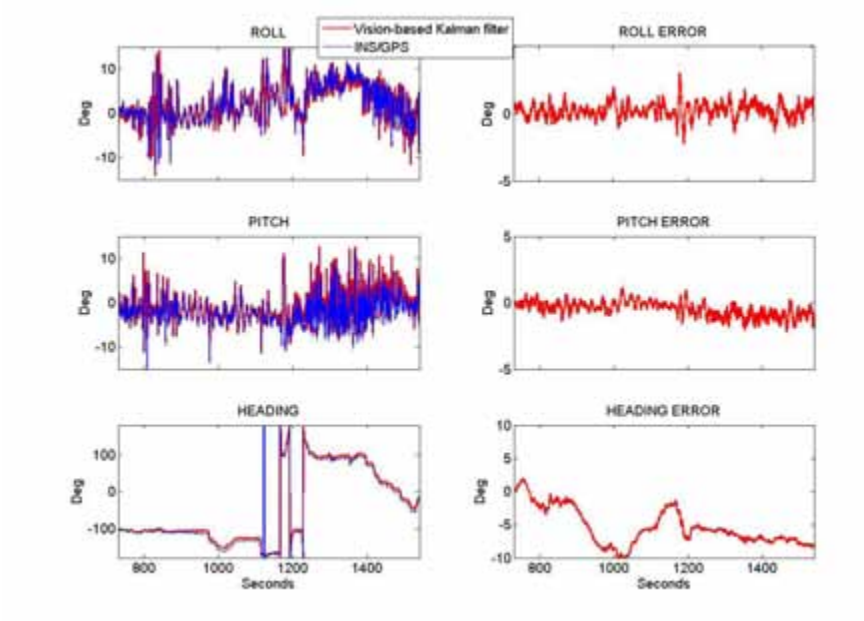


Figure 5.25: Real-time on-board results. Estimated Kalman filter attitude angles with error plots.

Chapter 6

State Estimation for Vision-based Landing

6.1 Introduction

This chapter presents experimental results of an autonomous vision-based landing technique based on vision and inertial sensing. The image processing system is not part of this thesis, details on this topic and on the control approach can be found in [43]. The chapter focuses on the integration between inertial and vision data and emphasizes the benefits derived from fusing these two sensor modalities compared to a pure vision solution.

The vision-based autonomous landing mode developed has been tested on an Rmax helicopter. It allows the helicopter to successfully complete a landing maneuver autonomously from an altitude of about 20 meters using only a single camera and inertial sensors (GPS is not used). An artificial landing pattern has been designed [43] and placed on the ground during the landing maneuver. A picture of the pattern is shown in Figure 6.1. An on-board video camera, mounted on a pan-tilt mechanism, locks on the pattern while an image processing algorithm computes the relative position of the on-board camera with the pattern. The relative position is then used to update a Kalman filter (KF) similar to the one described in appendix A

which estimates the inertial navigation errors.

During the landing phase a pan-tilt controller tracks the pattern keeping it in the middle of the image. This feature increases the robustness of the landing approach described here, minimizing the possibility of losing the pattern from the camera view due to accidental, abrupt helicopter movements.

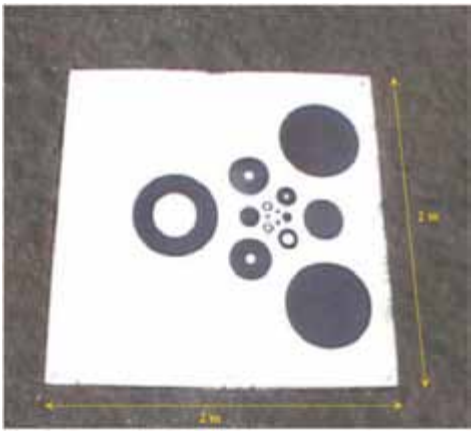


Figure 6.1: Pattern used for the vision-based autonomous landing.

The pose estimation algorithm presented in [43], computes the camera pose using three circles of known dimension placed in a triangular configuration. The special configuration of the landing pad (Figure 6.1) allows a vision-based landing approach from about 20 meters to the touch-down. The image processing algorithm selects always the larger three circle configuration available in the image to provide the best possible accuracy.

The motivations for the development of a vision-based landing mode are similar to those described in section 5.1. It must be pointed out that for an helicopter which lands in proximity of buildings or other structures, the multi-path phenomena can be a serious problem for the correct functioning of the GPS system.

Vision and inertial sensors are combined together because of their com-

plementarity. Vision provides drift-free position data, while inertial sensors provide position, velocity and attitude information at higher frequency but affected by drift. Depending on the price of the sensor, the drift of the inertial sensor can be more or less large. A very expensive inertial navigation unit allows an airplane to navigate for minutes or even hours without large drift. Usually, military and civilian aircraft or military submarines are equipped with such sensors. Small UAVs, such as the Rmax cannot be equipped with such accurate sensors. The reason is that the high costs of these sensors would make the platform too expensive. A second reason is that a small UAV has limited payload capacity and accurate sensors are usually quite heavy to be carried on-board a small UAV. This is the reason why for small UAVs it is common practice to fuse together several relatively cheap sensors with different characteristics. Figure 6.2 shows a classification of inertial sensor performance. The data is taken from [56].

Grade		Navigation	Tactical	Automotive	Consumer
Position error		1.9 (km/hr)	19-38 (km/hr)	≈2 (km/min)	≈3 (km/min)
Gyro	bias (deg/hr)	0.005-0.01	1-10	180	360
	scale factor (ppm)	5-50	200-500		
	noise (deg/hr/$\sqrt{\text{Hz}}$)	0.002-0.005	0.2-0.5		
Accel	bias (μg)	5-10	200-500	1200	2400
	scale factor (ppm)	10-20	400-1000		
	noise (μg/hr/$\sqrt{\text{Hz}}$)	5-10	200-400		

Figure 6.2: Inertial sensor performance classification. The data are taken from [56].

A vision-based landing approach which relies only on a vision system suffers from several problems. The vision system is sensitive to illumi-

nation conditions such as sun reflection or shadows which can partially cover the pattern. In these situations the vision system is blind and does not deliver any position data. A clever landing strategy must choose the proper landing direction in order to avoid these situations taking advantage of the knowledge of the sun position as explained in [43], but this is not always compatible with the wind direction which is also a factor when determining the landing approach. In addition, the pattern view can be accidentally lost shortly before the touch-down since it is hard for the camera pan-tilt control to track the pattern in this situation. Therefore in case the vision system does not deliver position information for a short time, the dead-reckoning capability of the INS can still deliver useful information to continue the landing maneuver. Fusing inertial and vision data together using the Kalman filter technique, allows for high frequency and drift-free helicopter state estimation. Sensor integration also allows low latency velocity estimation which is essential for stable helicopter control. Nevertheless the filter provides more accurate attitude information than the one given by vision only.

The vision-based landing problem for an unmanned helicopter has been addressed by other research groups. A good overview can be found in [54].

The following section provides an overview description of the landing approach.

6.2 System overview

The vision-based landing architecture is depicted in Figure 6.4. The image processing, the camera exposure control and the camera pan/tilt control (PTU) are implemented on a PC104 computer (IPC) while the KF and the helicopter control run on a separate PC104 (PFC). Details about the helicopter hardware architecture can be found in chapter 2. The image processing computes the complete helicopter pose (position and attitude) relative to the pattern. Only the position is used to update the KF though. The sensor fusion architecture follows a standard scheme where the vision position output is used as a measurement to update the KF which estimates the inertial navigation errors. The KF implementation is reported in appendix A.

Figure 6.3: The Rmax helicopter approaching the landing pad.

In Figure 6.4 are also displayed the update rates of the different sensor modalities. The vision system provides a position update rate of 20Hz. The accelerometers and gyros output data at a rate of 66Hz and 200Hz respectively. The KF delivers helicopter states at 50Hz rate.

6.3 Experimental results

The results presented in this section show the benefits resulting from fusing inertial sensors with a vision sensor during an autonomous vision-based landing approach. The sensors used to validate the fused vision-based results are a GPS system (centimeter accuracy) and the Yamaha Attitude System (YAS) for the attitude angles (around 2 deg accuracy).

The plots from Figure 6.5 to Figure 6.10 show flight-test data from an autonomous vision-based landing. In this particular test, the landing procedure starts around 910 sec and finishes with the touch-down around 965 sec.

Figure 6.5 shows the comparison between the filtered position and the raw vision position. Figures 6.6, 6.7 and 6.8 show the velocity components calculated by the filter compared with the GPS velocity. The upper plots of the 3 figures show an attempt at deriving the velocity from the raw

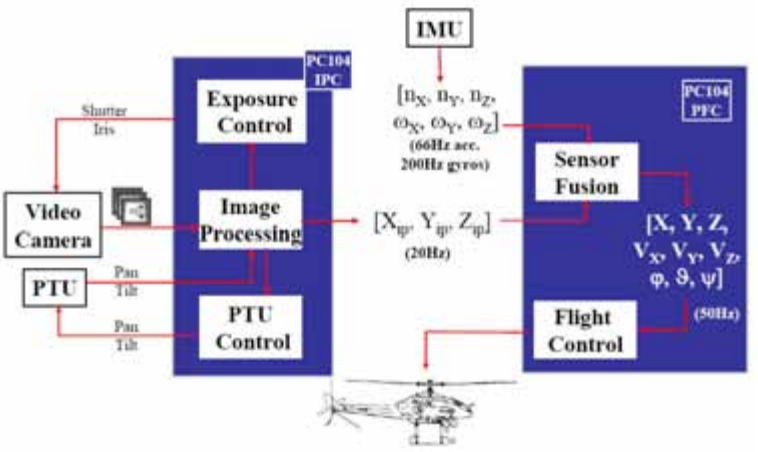


Figure 6.4: Vision-based landing system architecture.

vision position. The resulting velocity is quite noisy at the beginning of the landing due to the large distance from the pattern (around 15 meters). As it is shown in [43], the errors of the vision system are larger when the pattern is far from the helicopter. From the plots, it can be observed that as soon as the helicopter approaches the pattern the velocity derived from the vision position is less noisy. Early attempts have been made in applying a low-pass filter to this velocity to remove the noise but the increased delay made the tuning of the control system more difficult. The velocity data provided by the Kalman filter has low latency as can be observed from the comparison with the GPS velocity. This is due to the fact that the Kalman filter takes advantage of the high frequency and low latency information from the accelerometers. The use of low latency velocity information allows for stable control during the landing approach.

Figures 6.9 and 6.10, show a comparison between the attitude angles provided by the vision system alone, the attitude angles calculated by the Kalman filter and the attitude given by the YAS (attitude sensor built-in the Rmax helicopter used as reference). The attitude calculated by

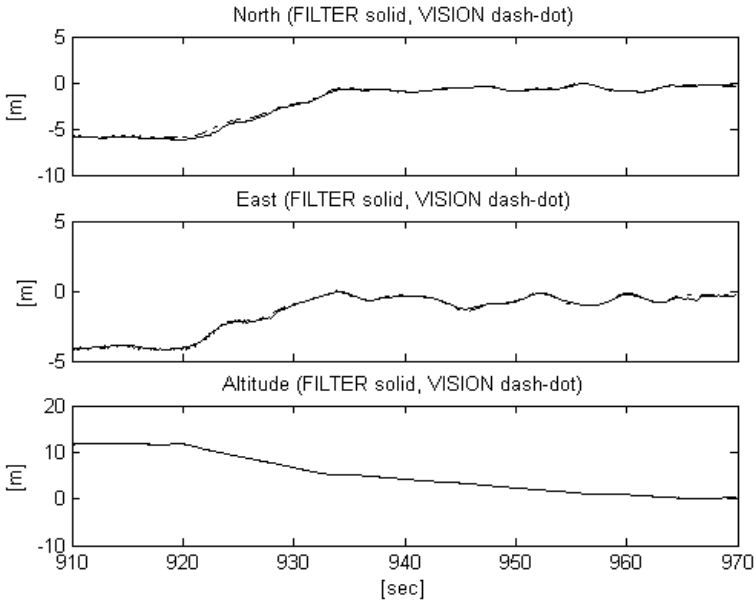


Figure 6.5: Comparison between vision and filtered position data during autonomous landing.

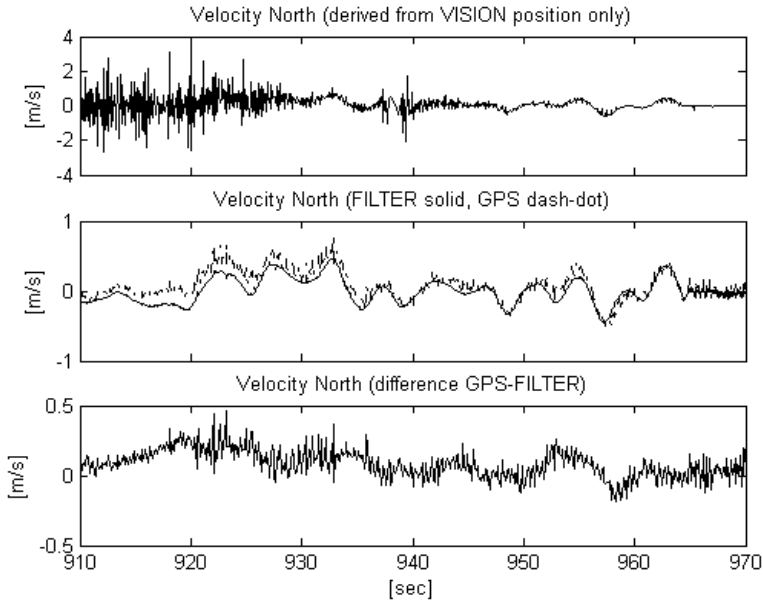


Figure 6.6: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the North component.

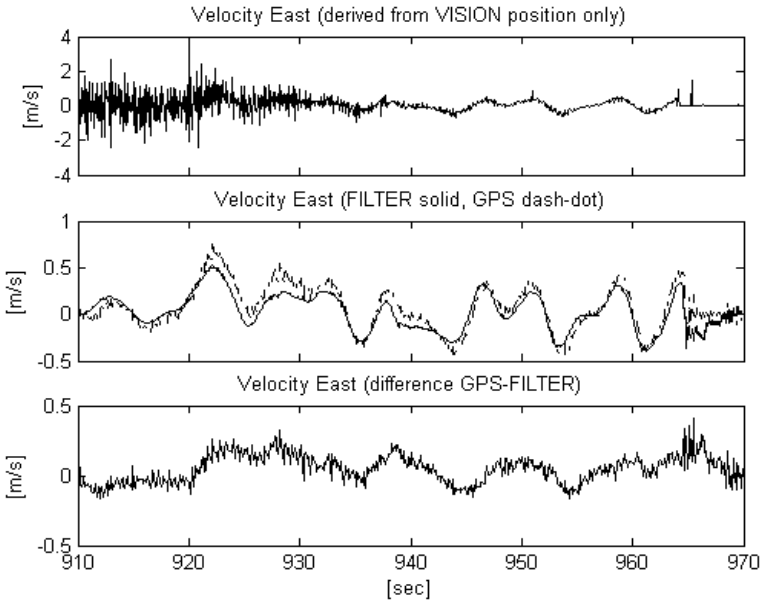


Figure 6.7: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the East component.

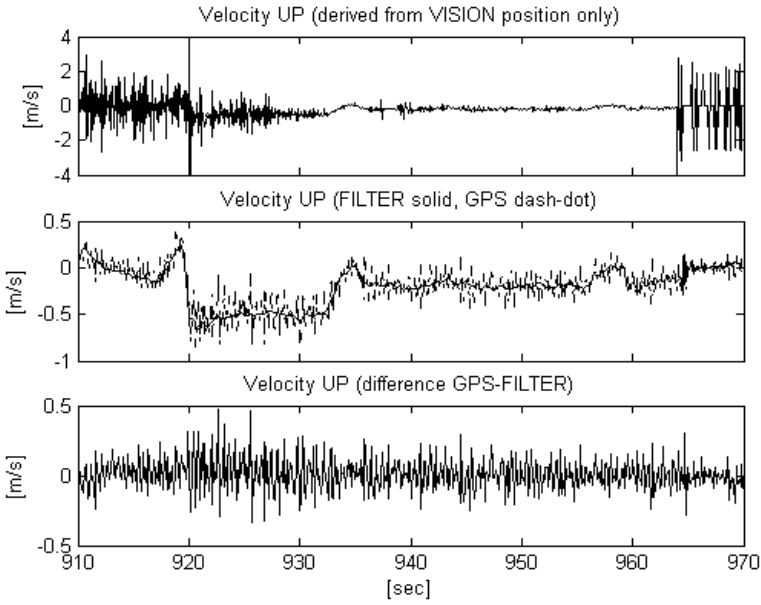


Figure 6.8: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the vertical component.

the vision system alone suffer from bias errors and high noise level when compared to the YAS data. The reason for the bias is the mounting error in angle between the camera platform and the helicopter body. The video camera is fixed on a suspended platform mounted on springs in order to damp helicopter vibrations and this produces alignment errors. The angles given by the filter do not suffer from this problem and are in good agreement with the YAS measurements.

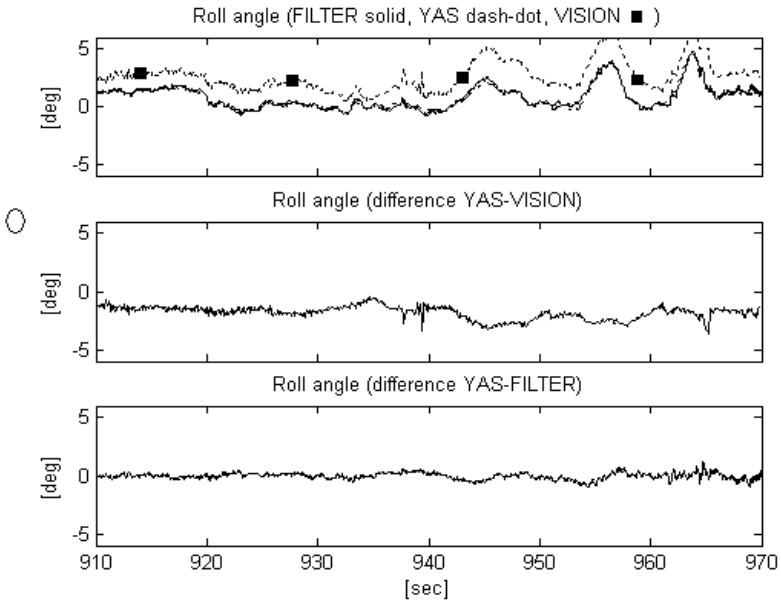


Figure 6.9: Comparison between roll angle calculated by the vision system, sensor fusion and YAS.

Figure 6.11 shows an altitude plot from one of the several landing tests. It is possible to observe that when the helicopter was at about 0.6 meters above the ground the vision system stopped delivering valid data because the pattern disappeared accidentally out of the camera's field of view. The

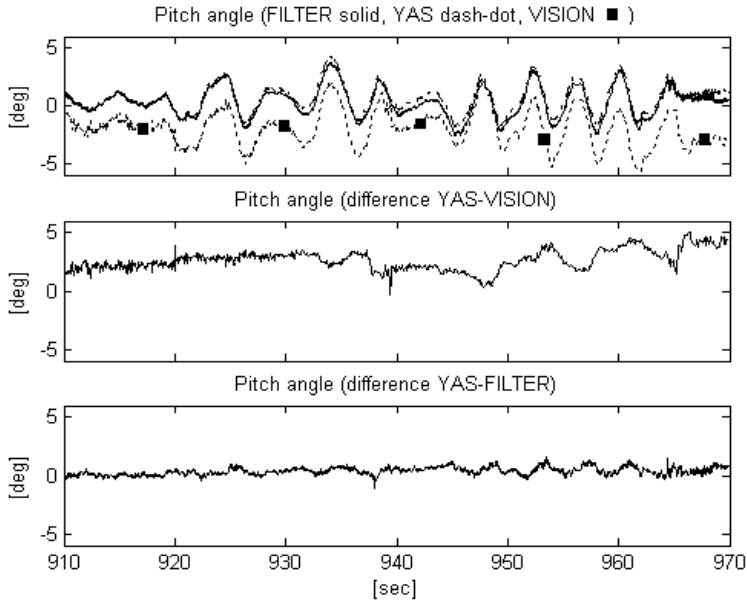


Figure 6.10: Comparison between pitch angle calculated by the vision system, sensor fusion and YAS.

filter continued to deliver position information using its dead-reckoning capability until the landing was terminated safely.

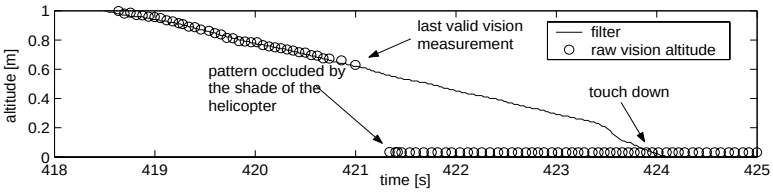


Figure 6.11: Altitude plot of an autonomous landing completed with the vision lost before touch down.

6.4 Conclusion

In this chapter, the benefit of integrating inertial sensors with a vision system as part of a vision-based autonomous landing system for an autonomous helicopter has been shown. The sensor fusion algorithm is based on a Kalman filter where the inertial sensor errors are estimated using position observation from a single camera vision system. The major benefits in fusing inertial sensors with a vision system can be summarized as resulting in a higher frequency state estimation, lower latency velocity estimation, more accurate attitude angle estimation and the possibility of surviving temporary black-out in the vision system.

Chapter 7

Vision-based Ground Target Geo-location

7.1 Introduction

This chapter deals with the problem of vision-based ground target geo-location from a fixed-wing Micro Aerial Vehicle (MAV) platform of a few hundred grams. What makes the MAV platforms interesting is their small size (order of centimeters) and affordable price. Their area of application includes target detection and localization but also more general tasks such as monitoring and surveillance. The method presented makes use of geo-referenced imagery to improve the ground target geo-location accuracy. The content of this chapter has been published in [15].

Since the method used here relies on geo-referenced imagery, the problems related to this method are similar to those already discussed for vision-based navigation in chapter 5. However, the matching method used for this application is different from the one used in chapter 5, it is in fact based on contour images instead of gray-scale images. The reason for this is that in this application the reference and sensed images are aligned using an image processing approach based on contour images. In fact, the inaccuracy of the MAV sensors does not allow for precise image alignment using only

on-board heading information. In any case, it is possible to find proper alignment using the contour image representation and, when aligned, performing the final 2D correlation between the images using the gray scale representation.

Besides working with Rmax size UAVs, the division of Artificial Intelligence and Integrated Computer System (AIICS) at Linköping University began work with MAV platforms in 2004. Since then, a small autonomous coaxial rotorcraft [21] called LinkMav was designed and developed. The LinkMav won the 1st US-European MAV Competition (MAV05) as "best rotary wing". In the beginning of 2007, the AIICS began development of a fixed-wing MAV called the *PingWing* depicted in (Figure 7.1). The PingWing took part and won the 3rd US-European Micro Aerial Vehicle Competition (MAV07) in Toulouse, September 2007. One of the tasks assigned during the MAV07 competition was to search for a vehicle in a specified region and compute its GPS coordinates. During the competition the car was localized with an error of about 4 meters computed by the judges. This turned out to be the best accuracy among the competitors.

Figure 7.1: *PingWing* micro aerial vehicle platform developed at Linköping University.

Precise ground target localization is an interesting problem and relevant not only for military, but also for civilian applications. For example, an UAV that will be used to automatically monitor road traffic behavior must

be able to discriminate if an object (in this specific case a car) is on a road segment or off road. The target geo-location accuracy required to solve this kind of problem must be at least the road width. The ground target localization error of this method can be potentially lower than 3 meters. Achieving such accuracy is a great challenge when using MAV platforms due to the fact that MAVs of a few hundred grams can only carry very light sensors. Such sensors usually have poor performance which prevents localizing a ground target with the necessary accuracy. The most common sensors used to geo-locate ground objects from a MAV platform are passive video cameras. The reason is that such sensors can be very light and also have low power consumption. The drawback is that they are sensitive to illumination conditions and do not provide direct range information. The miniaturization of active devices, such as radars and lasers, has not yet developed enough to support airborne applications on micro platforms.

When a ground target is within an image frame, detection (picking the target out of the clutter) can be performed manually (i.e. by an operator "clicking" on the screen), or automatically using image processing methods. Subsequently, the world coordinates can be calculated using the MAV position, attitude and the camera orientation relative to the MAV body. The MAV position is given by an on-board GPS receiver, while the attitude angles are computed from a navigation filter which integrates the inertial sensors (gyroscopes and accelerometers) and the GPS. The problem is that the measurement of the MAV position, attitude and camera angles are affected by several error sources which lead to a ground target localization error of the order of tens of meters.

The target geo-location method developed in this chapter is based on satellite image registration. Instead of using the MAV and camera state information to compute the geo-location of a ground object, the MAV camera view is registered to a geo-referenced satellite image with the coordinate of the ground object being calculated from the reference image. The fact that the availability of high resolution satellite or aerial images is rapidly growing and accessible to everybody makes this approach very attractive. In any case, even if this information would be missing, it is possible to imagine operational scenarios where the region of interest is photographed in advance. The experiment described in this chapter makes use of satellite images downloaded from Google Earth.

The advantage of this method is that the errors related to the MAV and camera pose do not affect the target localization accuracy because they are not directly used for the calculation. The MAV sensor information is used instead as a first estimate to restrict the search zone in the satellite image. The major advantage of this method is the capability of instantaneous performance versus the asymptotic performance of other methods. In principle it is enough to get one frame shot of the target to obtain a localization accuracy approximately equivalent to the geo-reference image accuracy. On the other hand, other methods usually require many measurement samples to decrease the localization error to an acceptable level. In addition, they require flying with a specific flight path around the target to be able to remove systematic errors. In order to acquire many measurement samples of a target, the loop between image processing and flight control system must be closed. For a platform of only a few hundred grams this still represents a challenge. Last but not least, the synchronization problem between a video image and flight data is not an issue when using the method proposed in this chapter.

7.2 Related work

The problem of target geo-location has been addressed previously in the literature [5, 38, 46, 11, 16] and continues to be of great research interest.

One of the major challenges when using MAV platforms is the estimation of the ground altitude which is essential for solving the ground object geo-location task. The problem lies in the fact that, due to limited payload capabilities, a MAV cannot carry a range sensor capable of measuring distances of hundreds of meters. Most of the methods found in the literature rely on a flat world assumption. In other words, it is assumed that there is no height difference between the take-off site and any other point on the ground. With such an assumption it is possible to compute the ground altitude simply by using a calibrated barometric sensor. The ground altitude is then computed by converting in relative altitude the differential pressure between the take-off position and any other point in space. Of course when the flat world assumption is violated an error will be introduced in the target geo-location calculation.

Target geo-location methods applied to MAV platforms can be divided into the following four categories, as discussed in [38].

- Map-based methods. With these methods the geo-location of a ground object is computed by intersecting the ray starting from the camera center and passing through the target pixel location in the image plane with the ground (Figure 7.2). The camera location and attitude must be known. A digital elevation model (DEM) of the terrain or the flat world assumption hypothesis must be employed for the calculation.

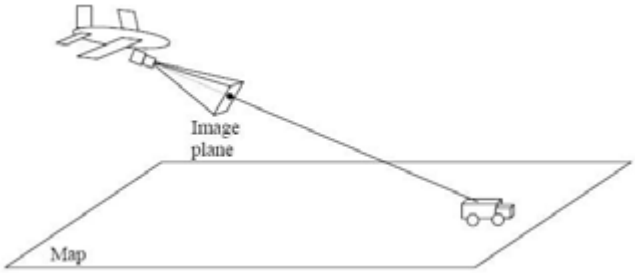


Figure 7.2: Mab-based geo-location geometry.

The accuracy of map-based geo-location methods depends strongly on how accurate the camera attitude is known and also on the accuracy of the camera position. Other error sources arise from the inaccuracy of the DEM or on how well the flat world assumption holds for the area of interest. The sensors typically installed on-board a MAV platform are of low accuracy due to limited payload capabilities, therefore the error of a single measurement sample of the target can be on the order of 40-60 meters when flying at about 100 meters altitude as shown in [5, 38]. The error sources affecting the target measurement can be divided in zero mean error noise, systematic errors and those related to the DEM accuracy.

The influence of the zero mean error noise can be attenuated by making an average on multiple target observations. The systematic errors

are usually biases in the estimation of the camera orientation. They can be the consequence of misalignment between the IMU sensor and the camera gimbal unit, inaccuracy in the calibration of the compass sensor or biases in the MAV's attitude estimation. The influence of systematic errors can be reduced by flying circular paths around the target. In [5] it is shown that with a least square filter applied to multiple observations with a MAV flying in circular paths, the error in the geo-location estimate decreases from 20-40m to less than five meters. The flat world assumption was employed in the experiment. The results were obtained using off-line flight data taken from an in-house built MAV with avionics.

- Line-of-sight filtering. This method is similar to the map-based method in the sense that the camera pose must be known but eliminates the need of having a DEM or using the flat world assumption hypothesis. The idea is to find the intersection point of the rays from multiple observations passing through the camera and the target, such intersection identifies the 3D coordinates of the target. Due to sensor errors the rays might not intersect each other, therefore the problem is to find the closest point to all the rays. Such a point is assumed to be the 3D target position. The problem can be solved using a Kalman filter estimator where bias and noise errors can be modeled explicitly. Multiple target observations are still required though in this case, in order to decrease the target position uncertainty. Such a method is evaluated in [38] and produces a horizontal targeting accuracy on the order of 10 meters at about 100 meters altitude. The results were obtained using off-line flight data taken from a commercial Raven-B MAV.

- Structure From Motion (SFM). This method is potentially more powerful compared to the previously described methods. The camera attitude is estimated purely by vision and not influenced by the attitude errors related to the attitude measurement sensor or misalignment of camera gimbal. This method was tested again in [38] where the GPS camera position was used as input in the algorithm to find the world coordinates of the target. The GPS errors influence the accuracy of this method. SFM requires the tracking of several targets through

several image frames. Therefore the level of complexity is higher than for the other two methods. The authors report a best case accuracy of about 5 meters for this method. The results have been derived using the same flight data as in the previous method.

- Geo-referenced imagery. This method can be very powerful since it is independent from the camera pose accuracy. It consists in registering the on-board aerial video on a geo-referenced image and extracting the target coordinates directly from the reference image. On-board sensor information can be used as a first estimate for the target position, but the final position is computed after the registration process. Therefore, the camera pose errors do not influence the geo-location accuracy. The main limitation here is the dependency on the availability of reference imagery for a particular area. On the other hand, the rapid development of imagery tools such as Google Earth makes this method very promising.

  The accuracy of the method also depends on the accuracy of the reference image. If the reference image is affected by uncompensated distortion or offset in the reference coordinates, the localization accuracy is affected. As an example, for the flight-test site used for the experiment reported below, an offset of about 3 meters was measured between our reference GPS and the Google Earth image. The offset was measured with a stationary high accuracy GPS (which was assumed to be the true position) and removed before flight. The ground altitude could be extracted from the image registration using a search in the image scale. Such an option increases the computational cost of the method and was not investigated in this work. The flat world assumption has been adopted in this case. Although image registration has been and continues to be an active research topic, an experimental evaluation of the accuracy of this method on MAV platforms is not found in the literature. Such evaluation will be the topic of this chapter.

7.3 Ground target geo-location based on image registration

The MAV system used for the experimental tests will be described in detail in section 7.4. It is important to mention that the MAV platform is equipped with an autopilot which allows it to fly autonomously. This allows automated planning of optimized search patterns and relieves the operator from flight control tasks, allowing him to concentrate on detection and classification task (tasks where the human is still better than the machine). In addition, a video camera is placed in the bottom part of the airframe and is mounted on a pitch-roll unit which allows it to rotate around the platform's pitch and roll axes (Figure 7.6). The pitch-roll camera unit is controlled from the MAV autopilot and programmed to counteract the MAV's pitch and roll rotations in order to keep the video camera always looking downwards perpendicularly to the terrain. By doing this, the deformation due to perspective can in practice be neglected and the video frames, after compensating for lens distortion, can be directly matched with ortho-rectified reference images.

The video stream is transmitted to a laptop on the ground which performs the image processing tasks and calculates the geo-location of the ground target. The reference image of the area where the MAV will perform the target identification and localization tasks is stored in the image processing laptop beforehand. The sensed images transmitted from the MAV are grabbed and processed on-line. Subsequently, the images are aligned and matched with the reference image. A second laptop is used to communicate with the MAV autopilot and receive telemetry data. The telemetry data is transmitted through an Ethernet connection to the image processing laptop and used for ground object localization.

The ground object is identified and the tracker initialized manually in the down-linked video from a ground operator. After the initialization, the object is tracked in subsequent image frames automatically using a template tracking algorithm. Details of the tracking method are presented in section 7.4. The result from the tracking algorithm is a pixel coordinate of the object in the image frame. The calculation from pixel to GPS coordinates is done automatically by the system using the method presented in this

section.

7.3.1 Image registration

The discussion about the different approaches to the image-to-map registration problem has been already addressed in section 5.7. In this application a correlation-based matching approach of contour images is used. The image registration scheme used is represented in the block diagram in Figure 7.3.

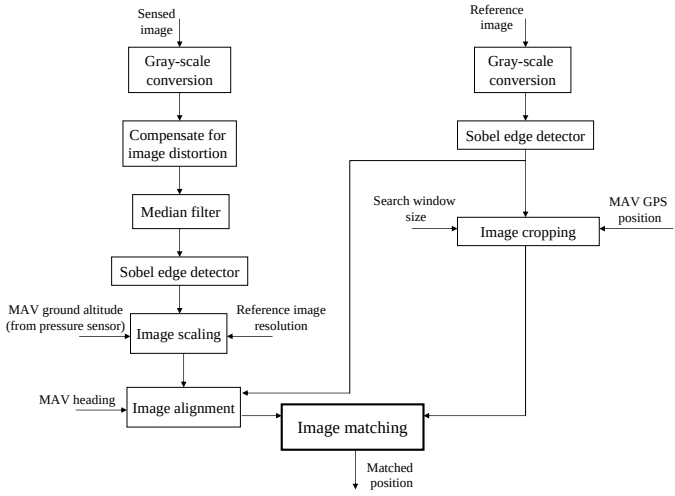


Figure 7.3: Image registration schematic.

The sensed image is pre-processed as follows. The image is converted into gray scale and compensated for the camera lens distortion. A median filter is applied in order to remove small details which are visible from the sensed image but not visible from the reference image. The median filter, has the well-suited property of removing small details while preserving the sharpness of the edges. After filtering, the Sobel edge detector is applied. The edge image must then be scaled and aligned to the reference image. Scaling is performed by converting the edge image to the resolution of the reference image as explained in section 5.7.

The ground distance is calculated using the flat world assumption as already mentioned in section 7.2 using a barometric pressure sensor (the atmospheric pressure at the ground level is taken before take-off so that the differential pressure during flight can be converted into ground altitude).

The sensed image is then aligned to the same orientation as the reference image. The alignment can be made using the heading information available on-board the MAV. Unfortunately, the heading accuracy needed for a successful matching is not enough. Therefore a method as to how to accurately align the sensed images to the reference image was developed and will be described in subsection 7.3.3. The on-board heading is used as an initial guess in the alignment calculation.

The reference image is processed as follows. It is converted into gray scale and the Sobel edge detector is applied. This is done only once, prior to flight, the resulting edge image is then kept in memory and used during flights. The MAV position taken from the on-board GPS receiver is used as the center of a restricted search area in the reference image. The purpose is to speed up the registration process, disregarding areas of the image too far from the ground target.

After both images have been processed and aligned as explained above, a correlation-based matching algorithm computes the 2D image translation which gives the best matching position. Once the match is obtained, the sensed image containing the target can be geo-referenced. A screen shot which shows how the two images are processed and then matched is shown in Figure 7.4.

7.3.2 Image distortion compensation

Prior to acquiring any images during flight the on-board camera is calibrated using a Calibration Toolbox for Matlab [1]. It is a convenient tool for determining the intrinsic camera parameters, namely focal length (f_x,f_y), principal point (c_x,c_y), radial distortion (k_1,k_2,k_3), tangential distortion (k_4,k_5) and the skew coefficient ($alpha$). Distortion from the images acquired by the on-board camera is removed at the beginning of the image processing pipeline.

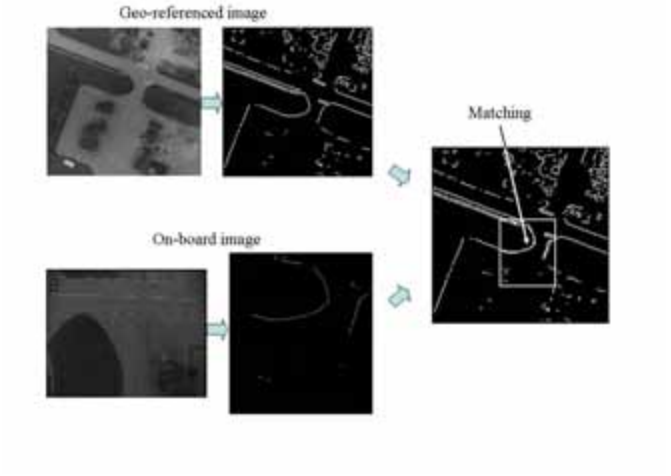


Figure 7.4: Processing of the reference and sensed images with final match.

7.3.3 Image alignment

This subsection describes the algorithm developed to align the sensed image to the reference image. During the experiment the camera is pointing downwards.

Algorithm 1 represents the image alignment block in Figure 7.3. It is based on the *Standard Hough Transform* which is a well known image processing technique used to detect lines in an image. The lines are parametrized as follows:

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta) \tag{7.1}$$

where ρ is the distance from the origin to the line along a vector perpendicular to the line and θ is the angle between the image x axis and this vector. The Hough Transform finds the parameter space θ-ρ which identifies the eventual lines in the image.

Algorithm 1 Algorithm for image alignment

1. Rotate the sensed image after scaling (see Figure 7.3), to the north direction (the reference image is also oriented to north) using the MAV heading information resulting from the telemetry data. The resulting image is called *image1*.
2. Extract a cropped image from the reference image where the center point coincides with the MAV GPS coordinates and window size equal to the size of *image1*. This image is called *image2*.
3. Compute the Standard Hough Transform of *image1* and *image2* and detect the corresponding lines in both images.
4. Compute the angle differences between the corresponding lines of *image1* and *image2* and calculate the average angle difference θ_{avg}.
5. If $|\theta_{avg}|$ is less than 45 degrees, rotate *image1* of θ_{avg} degrees otherwise keep the orientation of *image1* (this step assumes that the maximum heading error resulting from the telemetry data is 45 degrees).

Using Algorithm 1, the matching success rate improves dramatically. Experimental results have shown that using Algorithm 1 to align the images results in up to 3 times less false matches than using only the heading resulting from telemetry.

7.3.4 Ground object position calculation

The final ground object position is obtained as follows. After an object is selected by the ground operator, the template tracking algorithm is initialized and every time the object is in view it is automatically recognized by the tracker and the relative image saved together with the pixel coordinates of the template (the middle point of the template is considered to be the target position). Subsequently, the stored images (sensed images)

are automatically registered to the reference image using the method previously described. After the sensed image has been geo-referenced, the world coordinates of the object can be calculated using the pixel coordinates of the tracked template. Since the sensed image has been rotated before the matching, the template coordinates must be rotated using the same angle before the object world position can be extracted. The final object position is estimated using a recursive least square filter.

7.4 MAV system overview

The following section describes the system which has been used as a flying testbed for the validation of the method presented. All functional subcomponents and interconnections between them are depicted in Figure 7.5. The main components of the MAV system are a fixed-wing MAV flying platform and a ground control station which includes 2 laptop computers. Three types of communication links are used as depicted in Figure 7.5.

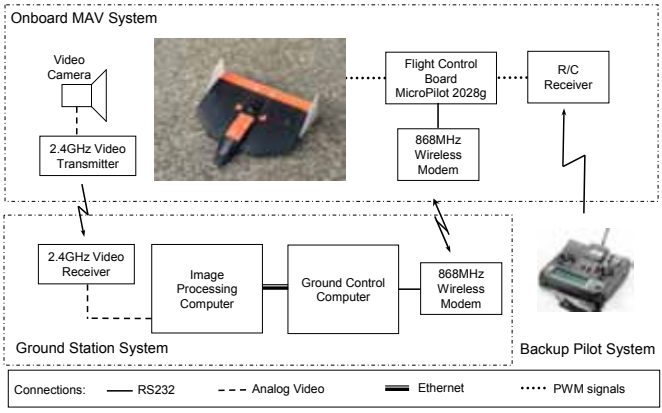


Figure 7.5: MAV system setup.

1. The communication with the autopilot is realized using an AeroCom AC4868 868MHz modem. The communication is bidirectional as can

be seen from Figure 7.5. A ground control computer can interact with the MAV sending flight instructions to the autopilot. The operator can change or add new waypoints dynamically during autonomous flight. The autopilot delivers the MAV state at 5Hz rate which is transmitted to the ground control computer for flight monitoring. In addition, the MAV state is transmitted to the image processing computer through an Ethernet connection and used for image registration.

2. An on-board analog video transmitter working on 2.4GHz frequency transmits the video to the ground. A diversity video receiver is connected to the image processing computer where the video is sent for image analysis. Both transmitter and receiver are from the Black Widow A/V company.

3. The third communication link is between the backup pilot and the MAV. It is realized using a standard R/C transmitter and receiver and used in case of emergency, such as loss of navigation capability (GPS unlock) or failure of the autopilot.

7.4.1 Airframe

The *PingWing* (Figure 7.1) is an in-house designed and manufactured flying wing. It is electrically powered with a brushless engine and Li-Po batteries. It has a wing span of 41 centimeters and a total weight of 434 grams. The flight endurance is 25 minutes with a cruise speed of 19 m/s. The PingWing weight distribution is shown in Table 7.1.

7.4.2 Autopilot

The autopilot used on-board is a commercial flight control board MP2028g manufactured by the Micropilot company. It is a lightweight system equipped with a 3-axis gyro, 3-axis accelerometer, a 4Hz Ublox GPS receiver, a pressure altimeter and a pressure airspeed sensor integrated on a single board. The autopilot is furnished with a standard control system implemented but can also be programmed and personalized by the user using the Xtender^{mp}

Airframe	108 g
Propulsion (incl. batt.)	122 g
Servos/receiver	31 g
Video system (incl. camera gimbal)	62 g
Data link	25 g
Autopilot	86 g
TOTAL	434 g

Table 7.1: PingWing weight distribution.

Image sensor	1/4" CCD
Resolution	490 TV lines
Sensitivity	1 lux
Lens	f=3.6mm (53° FOV)
Dimensions	25 x 25 x 22 mm

Table 7.2: Details of the video camera and lens used for the experiment.

software. The software is used to configure the autopilot telemetry in order to send the flight data from the ground control computer to the image processing computer. The compass sensor was not used in this experiment and heading information is derived from the GPS.

7.4.3 Video system

The PingWing is equipped with a lightweight CCD camera Videotronik VTQ-56. Details of the video camera and lens used are reported in Table 7.2.

The camera is mounted in an in-house developed pitch-roll gimbal (Figure 7.6) which allows automatic image stabilization.

The image processing functionalities have been implemented in the LabVIEW software environment. The software developed is capable of the following tasks: simultaneous recording of camera images and corresponding telemetry, tracking of a ground object, computation of the target's position in world coordinates with a map-based method (see section 7.2) and with

Figure 7.6: Pitch-roll camera gimbal.

the automatic image registration method described in this chapter.

The images coming from the video receiver are grabbed using the IMPERX VCE-PRO frame grabber with a resolution of 720x576 pixels. To reduce the amount of data only those images in which the ground target is visible are recorded. This is accomplished by utilizing a color pattern tracker provided by the NI Vision module available in the LabVIEW environment. For an extensive description of the color tracker the reader may refer to [2, 3].

The ground target geo-location procedure is implemented as follows. The first step consists in creating a template image representing the search object. This is done by the user only at the beginning when the object is first seen in the transmitted camera image. The object to be tracked has to be selected by the operator and a window with a predefined size is automatically drawn with the selected point as middle pixel. Subsequently, to creating the template image the algorithm has to learn the relevant features of the template. The learning process depends on the setup parameters specified for the learning phase. The appropriate parameters used in the learning phase and during the matching process have been determined empirically and depend on the object to be tracked and differing environmental conditions. The pattern matching algorithm searches for the

template in each of the following images using color and shape information. Since searching the entire image for matches is extremely time consuming a coarse-to-fine search strategy first locates a rough position. The location is refined in a second step using a hill climbing algorithm around each match [2]. The result of the template tracker is the position of the template in the image plane (pixel coordinates). The object's position is assumed to be at the center of the template. The size of the template is predefined for convenience and it is chosen according to the predicted object size and the typical flight altitude. The calculation from the pixel coordinates to world coordinates is done using the image registration approach described in section 7.3. Before the sensed images are registered, they are reduced to half resolution 360x288 to speed up the computation. The results are compared with the actual coordinates of the target, in our case measured with a stationary GPS, and employed for error analysis.

The program can be utilized for various scenarios, i.e. in any environment because any geo-referenced image can be loaded. Parameters specifying essential attributes of the reference image, e.g. image dimensions and image resolution, as well as camera parameters, e.g. focal length, principal point, radial and tangential distortion, have to be entered in the user interface.

7.5 Experimental results

This section presents the experimental results of ground object geo-location using the approach described in this chapter. The MAV system used for the experiment has been described in section 7.4.

Two different scenarios will be analyzed. In the first scenario a truck, placed on a structured road system, was geo-located from the PingWing in an automated manner. The results of this experiment are used to evaluate the accuracy of the method. In the second scenario, a target car was placed in an area of 300x300 meters and the PingWing was programmed to scan the area, search for the car and compute its GPS coordinates.

1. **First scenario**

   In this experiment a truck was parked on a road system without mov-

ing. The PingWing was commanded to fly above the truck location and, as soon as the truck was in the camera view, the ground operator selected the template window in the image and started the tracking algorithm (Figure 7.7).

The PingWing flew autonomously in a figure of eight pattern centered on the ground target (Figure 7.7) at a constant altitude of 70 meters. Every time the template tracking algorithm recognized the target, it saved the image together with the target pixel position. Subsequently, the images were registered to a geo-referenced image of the area and the truck position calculated as described in section 7.3. The images containing the target were saved at about 5Hz frame rate which is the rate of the template tracking algorithm.

Figure 7.7: On the left picture is displayed the MAV's flight path during the ground object localization task. On the right, the ground object is automatically tracked in the image (red square).

The geo-referenced image used for this experiment was acquired from Google Earth. The truck position was measured on the ground with a GPS receiver (used as a reference position instrument) which has an accuracy of about 2-3 meters. The measurement samples were averaged to decrease the truck position uncertainty. A bias of 3 meters was found between the measurement taken with the reference GPS and the corresponding position calculated using the Google Earth

image. In order to avoid errors introduced by the reference GPS, the bias of the Google Earth image was compensated using the GPS measurement.

Figure 7.8 shows the results of the target geo-location calculated with the image registration method. It can be observed that there are some outliers due to incorrect matches. The incorrect matches occur because, as can be seen in Figure 7.7, the road system presents repetitive structures which are quite similar to each other. It can be the case that the matching algorithm registers the sensed image to a similar location which is not the right one. In any case, the number of false matches are sporadic and can be filtered out. Figure 7.9 displays the position error of the measurement samples (top) and the result of the recursive least square filter applied to the measurement data (bottom). The error stabilizes after 20-30 samples. The estimated target position error is about 2.3 meters. The outlier measurements are also included in the filtering process but do not have a major influence in the final result.

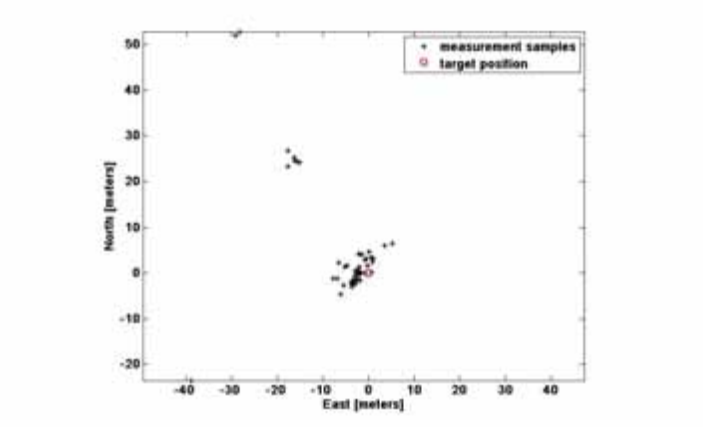


Figure 7.8: Ground target position measured from the MAV system using the image registration technique described in this chapter. The target position is in (0,0).

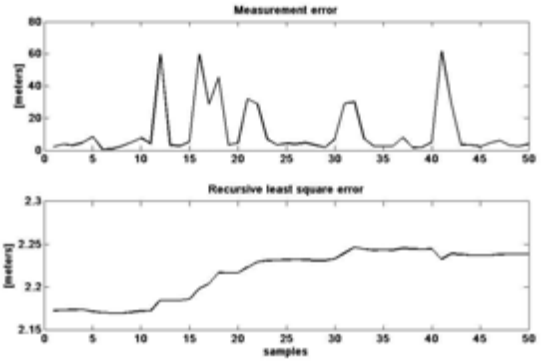


Figure 7.9: Top figure: target collocation error using image registration technique. Bottom figure: estimated target error using recursive least square filter.

As a comparison, using the same data set, the ground target position was calculated with the map-based method described in section 7.2. For this calculation the MAV sensor data coming from the telemetry was used. In Figure 7.10, it can be observed how the measurement samples are spread over a larger area compared to Figure 7.8. In addition the sample distribution has a bias which leads to an estimated target position error of about 22 meters as it is displayed at the bottom of Figure 7.11. The bias could be compensated by choosing a flight path which makes a circle around the target as is shown in [5].

2. **Second scenario**

   The second experiment described was executed as part of the competition mission during the 3rd US-European Micro Aerial Vehicle Competition (MAV07) in Toulouse, September 2007. The specific task was to search for a military car placed at about 1 km distance from the take-off position and compute its geo-location coordinate. The reference image of the competition area, shown at the top of Figure 7.12, was downloaded from Google Earth. The area is less structured compared to the previous experiment.

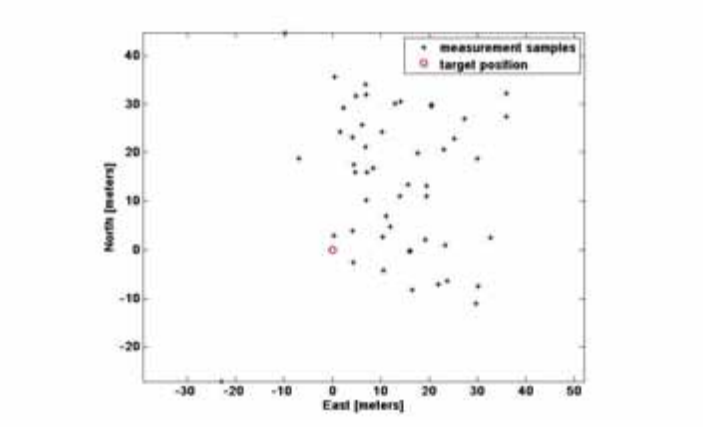


Figure 7.10: Ground target position measured from the MAV system using the map-based method. The target position is in (0,0).

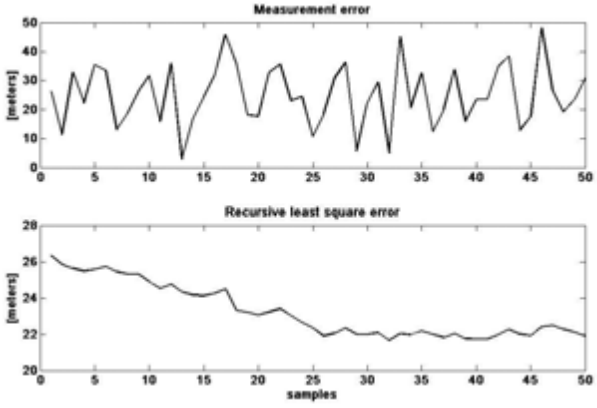


Figure 7.11: Top figure: target collocation error using the map-based method. Bottom figure: estimated target error using recursive least square filter.

Figure 7.12: The upper picture is the reference image of the competition site downloaded from Google Earth. The lower picture is the target car spotted from the PingWing flying at an altitude of 80 meters from the ground.

At the bottom of Figure 7.12 is displayed the car spotted by the PingWing while scanning the assigned search zone. The flight path of the complete mission is depicted in Figure 7.13. The search zone of 300x300 meters was placed at about 1 km from the take-off position and ground station. As can be observed (bottom of Figure 7.12), the on-board video transmitted to the ground station was of low quality due to the large distance and the low power video transmitter installed on board. For this reason the geo-location required a su-

pervision from the ground operator. After selecting the best image available of the target, the registration procedure was applied. Due to the strong image noise, the registration required a further manual adjustment which was in any case simplified from the automatic pre-registration step.

The car coordinate were computed and transmitted to the judges which confirmed an error of 4 meters according to their calculations. The experiment shows how, even in case of few noisy frames (in this case only one frame was used), this technique allows to localize a ground target with a good accuracy.

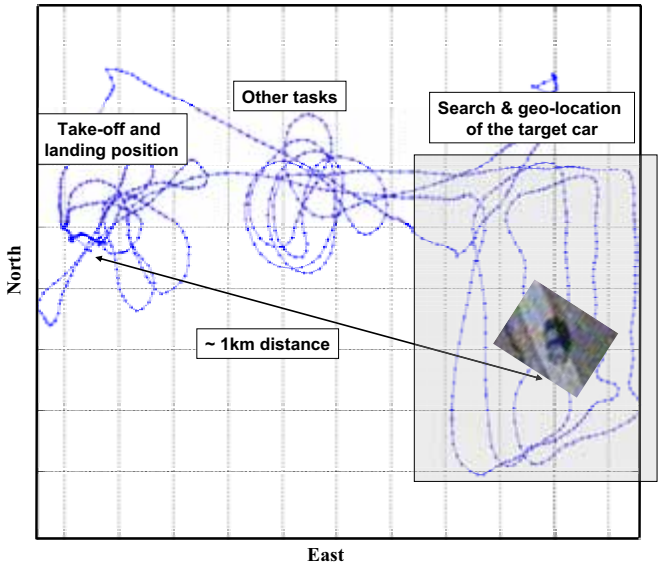


Figure 7.13: Flight path of the PingWing flown during the competition. It can be observed on the right, the scanning pattern during the search for the target car at about 1 km distance from the ground station. The flight was performed in autonomous mode.

7.6 Conclusion

The main advantages using the method presented in this chapter can be summarized as follows:

1. High accuracy ground target localization can be achieved even using low performance sensors.
2. Instantaneous performance (vs. asymptotic). In contrast to other approaches which require many measurement samples, this method allows high accuracy with only a few samples.
3. A specific flight path around the target is not required to remove systematic sensor errors.

The disadvantages of the method can be summarized as follows:

1. A geo-referenced image of the environment is needed to localize the target.
2. The structural properties of the landscape must be suitable for image registration applications.
3. The accuracy of the method depends on the accuracy of the reference image. A calibration of reference image might be required beforehand.

In summary, when the method is applicable, it is very powerful and capable of high accuracy performance. It would be useful to detect in advance the areas of the map where the method can be applied reliably so that it would be possible to build in the system the capability to choose the most appropriate localization method for each area.

The image registration technique used in the experiment presented is based on correlation measure between the reference and sensed image after they are converted into binary contour images. The use of correlation based matching with contour images might not be the best choice from the success rate point of view. More sophisticated methods can be used which give better results on a larger variety of landscapes with the drawback of an increase in computational complexity. It is a user choice to select the method which is more appropriate for the hardware in use.

Chapter 8

Conclusions

This thesis has covered several topics related to navigation of unmanned aircraft systems. The Yamaha Rmax unmanned helicopter has been used as an experimental testbed where the algorithms proposed have been implemented and tested during real flight-tests.

The Rmax helicopter has a built-in digital attitude stabilization system, called YACS, which was used in the control system architecture. A mathematical model of the helicopter dynamics including the YACS dynamics was identified for control purposes. The model was described in chapter 3.

A guidance method which enables the helicopter to follow 3D paths was described in chapter 4. The method is general and can be used to follow any kind of parametrized 3D curve. The guidance method is implemented on the Rmax helicopter and it is currently used in autonomous flight missions.

Chapter 5 addressed the problem of vision-based state estimation for autonomous navigation without GPS. The method proposed fuses information from inertial sensors with position information from a vision system using a Kalman filter technique. The vision system computes the absolute UAV position robustly by fusing relative position information (visual odometry) with absolute position information obtained through registration of the on-board video with geo-referenced imagery. A discretized Bayesian filtering approach, called point-mass filter, is used to fuse the two different sources of visual position information. The method is implemented on the

Rmax helicopter and flight-test trials have demonstrated the capability of such method to provide the necessary state information for navigating a UAV in situations of GPS outage.

Chapter 6 has presented experimental results of a vision-based landing approach which uses inertial sensor data and position data from a vision system relying on an artificial marker placed on the ground.

The last topic of the thesis was presented in chapter 7 where a vision-based ground target localization approach for aerial platforms was proposed. The approach has the capability to localize a ground target with high accuracy even using platforms with poor sensor performance. The method proposed relies on prior landscape information (geo-referenced imagery) and has been applied on a micro aerial vehicle system of a few hundred grams.

Appendix A

A.1 Kalman filter architecture

The Kalman filter (KF) used in chapters 5 and 6 uses a linear model for the measurement and process equations. The filter is implemented in the error dynamics formulation where the filter state represents the errors of the inertial navigation system.

The integration between the inertial sensors and the absolute position sensor is realized through an error feedback scheme depicted in Figure A.1. The INS (inertial navigation system) mechanization process is responsible for the time integration of the gyros and accelerometers sensors providing a full navigation solution (position, velocity and attitude) affected by drift.

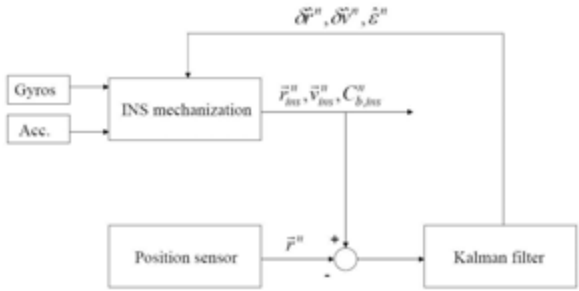


Figure A.1: INS aided integration scheme.

The KF, estimates the position, velocity and attitude errors ($\delta\hat{r}^n$, $\delta\hat{v}^n$, $\hat{\epsilon}^n$) of the INS and feeds them back into the INS mechanization process to avoid unbounded error growth. Provided that the position sensor (usually a GPS) supplies absolute position information (not affected by drift), the scheme of Figure A.1 is capable of computing high-rate and drift-free full state estimation.

The scheme presented in Figure A.1 is used in chapters 5 and 6 where the position update comes from the relative image processing. In the following part of this appendix the basic equations of the INS mechanization process and of the KF will be given. The sensor fusion approach described in this appendix follows a standard scheme widely used in the literature. For more details, the reader can refer to [40, 59, 56].

A.1.1 INS mechanization

The task of the INS mechanization process is the time integration of the inertial sensors (gyros and accelerometers) to provide the navigation parameters (position, velocity and attitudes). The relation between the navigation parameters and the inertial sensor measurements (3 axis accelerations and 3 axis angular rates) are represented with the following differential equations:

$$\begin{aligned} \dot{\vec{r}}^n &= \vec{v}^n \\ \dot{\vec{v}}^n &= C_b^n \vec{f}^b - (2\vec{\omega}_{ie}^{\,n} + \vec{\omega}_{en}^{\,n}) \times \vec{v}^n + \vec{g}^n \\ \dot{C}_b^n &= C_b^n (\vec{\Omega}_{ib}^{\,b} - \vec{\Omega}_{in}^{\,b}) \end{aligned} \tag{A.1}$$

where $\vec{r}^n$ is the position, $\vec{v}^n$ the velocity and C_b^n the direction cosine matrix of the attitude angles. The accelerometer and gyro inputs are represented respectively by $\vec{f}^b$ and $\vec{\Omega}_{ib}^{\,b}$, $\vec{g}^n$ is the gravity vector, $\vec{\omega}_{ie}^{\,n}$ the Earth rotation rate, $\vec{\omega}_{en}^{\,n}$ is the rotation rate of the navigation system relative to the Earth and $\vec{\Omega}_{in}^{\,b}$ is the rotation rate of the navigation system relative to the inertial system. The INS mechanization algorithm performs the time integration of the set of Equations in A.1.

The calculation of the attitude angles is done using a quaternion representation. Such a representation is advantageous since the linearity of the

differential equations of the quaternion dynamics allows for an efficient implementation [25]. Details of the implementation of the INS mechanization can be found in [56].

Since the inertial sensor inputs $\vec{f}^{\,b}$ and $\vec{\Omega}_{ib}^{\,b}$ are affected by several error sources, when performing the integration of Equations A.1, the sensor errors are integrated as well producing an unbounded drift in the navigation parameters. The purpose of the KF described in the following section is to estimate the navigation parameter errors so that they can be fed back to the INS mechanization process and the errors subtracted from the navigation parameters.

A.1.2 Kalman filter

Given a state-space system, the associated recursive Bayesian filtering problem was presented in section 5.8.4 of this thesis and represented with Equations (5.17)-(5.21). Under the hypothesis of linear state-space system and Gaussian noise distribution, a recursive solution to the Bayesian filtering problem can be found and is represented by the Kalman filter.

The continuous linear state-space system is represented as:

$$\dot{\vec{x}} = F\vec{x} + G\vec{u} \tag{A.2}$$

$$\vec{y} = H\vec{x} + \vec{e} \tag{A.3}$$

where Equation A.2 represents the process model and Equation A.3 represents the measurement model. In addition, $\vec{x}$ represents the state of the system to be estimated, $\vec{u}$ and $\vec{e}$ the system and measurement noises and $\vec{y}$ the measurement vector.

The KF is implemented using a state vector $\vec{x} = (\delta\vec{r}^{\,n}\ \delta\vec{v}^{\,n}\ \vec{\epsilon}^{\,n}\ \delta\vec{a})^T$ of 12 components representing the INS errors:

$$\begin{aligned}
\delta\vec{r}^{\,n} &= (\delta\varphi\ \delta\lambda\ \delta h)^T = \text{latitude, longitude and altitude error states;}\\
\delta\vec{v}^{\,n} &= (\delta v_n\ \delta v_e\ \delta v_d)^T = \text{north, east and down velocity error states;}\\
\vec{\epsilon}^{\,n} &= (\delta\phi\ \delta\theta\ \delta\psi)^T = \text{pitch, roll and heading error states;}\\
\delta\vec{a} &= (\delta a_x\ \delta a_y\ \delta a_z)^T = \text{accelerometer biases.}
\end{aligned}$$

The dynamics matrix F of the process model (Equation A.2) represents the INS error dynamics expressed by the following system of differential equations:

$$\begin{aligned} \dot{\delta\vec{r}} &= -\vec{\omega}_{en} \times \delta\vec{r} + \delta\vec{v} \\ \dot{\delta\vec{v}} &= -(\vec{\omega}_{ie} + \vec{\omega}_{in}) \times \delta\vec{v} - \vec{\epsilon} \times \vec{f} + \delta\vec{a} \\ \dot{\vec{\epsilon}} &= -\vec{\omega}_{in} \times \vec{\epsilon} \\ \dot{\delta\vec{a}} &= -\beta\, \delta\vec{a} \end{aligned} \tag{A.4}$$

with $\vec{\omega}_{en}$ the rotation rate of the navigation reference system relative to the Earth reference system, $\vec{\omega}_{in}$, the rotation rate of the navigation reference system relative to the inertial reference system and $\vec{\omega}_{ie}$ the rotation rate of the Earth reference system relative to the inertial reference system. $\vec{f}$ is the acceleration vector expressed in the navigation reference system.

The KF recursion solves the Bayesian filtering problem for the following discretized state-space model:

$$\vec{x}_{k+1} = \Phi_k \vec{x}_k + \vec{w}_k \tag{A.5}$$

$$\vec{y}_k = H_k \vec{x}_k + \vec{e}_k \tag{A.6}$$

where ϕ_k is the state transition matrix which can be approximated as follows:

$$\Phi_k \approx I + F\Delta t \tag{A.7}$$

with Δt the integration time interval. The following equations represent the well known discrete KF recursion:

Prediction	*Update*
$\hat{x}_k^- = \Phi_k \hat{x}_{k-1}^+$	$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$
$P_k^- = \Phi_k P_{k-1}^+ \Phi_k^T + Q_{k-1}$	$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-)$
	$P_k^+ = P_k^- - K_k H_k P_k^-$

where P is the estimation covariance. The uncertainty in the system is injected through the matrix Q_k. An approximate expression for Q_k is :

$$Q_k \approx \Phi_k G Q G^T \Phi_k^T \Delta t \tag{A.8}$$

where G is a matrix which depends on the problem structure and was introduced in Equation A.2 and Q represents the spectral density matrix and it is defined as:

$$Q = diag(\sigma_{ax}^2\, \sigma_{ay}^2\, \sigma_{az}^2\, \sigma_{\omega x}^2\, \sigma_{\omega y}^2\, \sigma_{\omega z}^2) \tag{A.9}$$

with σ_a and σ_ω the standard deviations of accelerometers and gyroscopes respectively. R_k represents the measurement covariance matrix defined as:

$$R_k = diag(\sigma_\phi^2\ \sigma_\lambda^2\ \sigma_h^2) \tag{A.10}$$

Appendix B

The content of this appendix has been derived from [30] and summarizes the basic background theory used in section 5.6 of this thesis.

B.1 Homography estimation from point features

The homography matrix H is a projective transformation which describes the relation between corresponding 3D coplanar world points observed in two camera images. A homography relation exists also between corresponding 3D non-coplanar world points if the camera has a purely rotational motion.

The relation between coplanar points observed in two images can be expressed as $\vec{x}_2 \approx H\vec{x}_1$ where $\vec{x}_1$ and $\vec{x}_2$ are the corresponding projection of the same 3D point in two different camera views (1 and 2) expressed in homogeneous coordinates. H is the 3 x 3 homography matrix. The symbol $\approx$ indicates that the relation is valid up to a scale factor. Such points are normally expressed in homogeneous coordinates $\vec{x} = (u, v, 1)$ where u and v are the pixel coordinates in the image plane. A point is expressed in homogeneous coordinates when it is represented by equivalence classes of coordinate triples $(k\,x,\, k\,y,\, k)$ where k is a multiplicative factor.

The 3 x 3 homography matrix has 9 entries but it is defined up to a scale factor. Consequently the total number of degrees of freedom is 8.

In addition, each point $\vec{x}$ has 3 components but is represented in homogeneous form meaning that it is defined up to a scale factor (the magnitude). It follows that each point correspondence gives 2 independent equations hence a *minimal* configuration requires a set of 4 corresponding points for homography estimation. Since the point locations identified in an image are corrupted by noise or, in the worst case, wrong correspondences might occur between points in the two images, the number of point features normally used for homography estimation is higher than 4, the problem is then over-constrained. The use of minimal point configurations is in any case relevant when using the RANSAC algorithm as it will be discussed later.

There exist configurations of points called *degenerate* which must be avoided when estimating H since they do not identify a unique class of transformations. When estimating the homography transformation using 4 points, if 3 of the 4 points are collinear the configuration is degenerate, in other words the problem is not sufficiently constrained. Such a configuration must be avoided.

As stated previously, the point feature correspondences can be affected by noise or by wrong feature association (outliers). The effect produced by the feature noise in the final displacement computed by the odometry is evaluated in section 5.6.1 of this thesis. The RANSAC algorithm is used to detect and discard the outliers in the feature data set. More details about RANSAC can be found in section B.3 of this appendix.

B.2 The direct linear transformation (DLT) algorithm

The DLT is the algorithm used to compute the homography matrix H given a set of corresponding points of a planar 3D scene in two image views. The relation $\vec{x}_2 = H\vec{x}_1$ can be expressed in terms of the vector cross product:

$$\vec{x}_2^i \times H\vec{x}_1^i = 0 \quad i = 1, ..., N \tag{B.1}$$

with N the number of point correspondences and the subscripts 1 and 2 indicating the two image views. Denoting with $\vec{h}_j^T$ the j-th row of the matrix H:

$$H\vec{x}_1^{\,i} = \begin{pmatrix} \vec{h}_1^{\,T}\vec{x}_1^{\,i} \\ \vec{h}_2^{\,T}\vec{x}_1^{\,i} \\ \vec{h}_3^{\,T}\vec{x}_1^{\,i} \end{pmatrix} \tag{B.2}$$

If $\vec{x}_2^{\,i} = (x_2^i,\, y_2^i,\, z_2^i)^T$, the cross product can be written as:

$$\vec{x}_2^{\,i} \times H\vec{x}_1^{\,i} = \begin{pmatrix} y_2^i\,\vec{h}_3^{\,T}\,\vec{x}_1^{\,i} - z_2^i\,\vec{h}_2^{\,T}\,\vec{x}_1^{\,i} \\ z_2^i\,\vec{h}_1^{\,T}\,\vec{x}_1^{\,i} - x_2^i\,\vec{h}_3^{\,T}\,\vec{x}_1^{\,i} \\ x_2^i\,\vec{h}_2^{\,T}\,\vec{x}_1^{\,i} - y_2^i\,\vec{h}_1^{\,T}\,\vec{x}_1^{\,i} \end{pmatrix} \tag{B.3}$$

Equation B.1 can be finally rewritten as follows:

$$\begin{bmatrix} 0^T & -z_2^i\,\vec{x}_1^{\,i\,T} & y_2^i\,\vec{x}_1^{\,i\,T} \\ z_2^i\,\vec{x}_1^{\,i\,T} & 0^T & -x_2^i\,\vec{x}_1^{\,i\,T} \\ -y_2^i\,\vec{x}_1^{\,i\,T} & x_2^i\,\vec{x}_1^{\,i\,T} & 0^T \end{bmatrix} \begin{pmatrix} \vec{h}_1 \\ \vec{h}_2 \\ \vec{h}_3 \end{pmatrix} = 0 \tag{B.4}$$

which is a linear homogeneous system of three equations with $\vec{h}$ a 9-dimensions vector in the entries of H:

$$\vec{h}_1 = (h_{11}\, h_{12}\, h_{13})^T \tag{B.5}$$
$$\vec{h}_2 = (h_{21}\, h_{22}\, h_{23})^T \tag{B.6}$$
$$\vec{h}_3 = (h_{31}\, h_{32}\, h_{33})^T \tag{B.7}$$

In system (B.4) only two of the three equations are independent, therefore the third equation can be omitted as it is obtained up to scale. The system (B.4) can be rewritten as:

$$\begin{bmatrix} 0^T & -z_2^i\,\vec{x}_1^{\,i\,T} & y_2^i\,\vec{x}_1^{\,i\,T} \\ z_2^i\,\vec{x}_1^{\,i\,T} & 0^T & -x_2^i\,\vec{x}_1^{\,i\,T} \end{bmatrix} \begin{pmatrix} \vec{h}_1 \\ \vec{h}_2 \\ \vec{h}_3 \end{pmatrix} = 0 \tag{B.8}$$

which can be represented in the form $A_i\vec{h} = 0$ with A_i a 2 x 9 matrix.

The system of equations (B.8) hold for any homogeneous representation of the point $\vec{x} = (x, y, z)$. If the third component is taken as $z = 1$ then x and y represent the pixel coordinates of the point in the image.

If a minimal configuration of points is taken ($i = 4$), the system (B.8) becomes a 8 x 9 homogeneous system. The matrix A_i of the system has rank 8 therefore the homography matrix H can be recovered up to a scale factor. Usually the additional arbitrary constraint $\|\vec{h}\| = 1$ is used to compute a scale for the system.

If a higher number of point correspondences is used ($i > 4$) the system (B.8) is over-determined. In any case, if the point feature correspondences are exact, the matrix A_i will always have rank 8 and an exact solution for the system $A_i\vec{h} = 0$ exists. In a real case, the point feature correspondences are not exact because they are corrupted by noise. Therefore, an exact solution for $A_i\vec{h} = 0$ which is different from zero cannot be found (the matrix A_i does not have rank 8). The problem is then to find a solution which minimizes the norm $\|A_i\vec{h}\|$ with the constraint $\|\vec{h}\| = 1$ which is equivalent of minimizing $\|A_i\vec{h}\|/\|\vec{h}\|$. The solution is the unit singular vector corresponding to the smallest singular value of A_i. The algorithm which solves the problem is known as the *basic DLT algorithm*.

It is common, before applying the DLT algorithm, to perform a normalization on the point data set. The point set in both images are translated and scaled in a way that the centroid of the point configuration corresponds to the origin of the image (usually the upper left corner) and the mean distance of the points to the origin is $\sqrt{2}$. The average point will have coordinate (1, 1, 1). The two point sets in the two images are normalized independently from each other. Without entering into details, without the normalization step the matrix A_i might have a large condition number so that in the presence of noisy data the solution might diverge from the right one. After the DLT algorithm, a denormalization step is applied.

The *normalized DLT algorithm* is reported in Algorithm 2 and is taken from [30].

Algorithm 2 ***Normalized DLT algorithm.***
Given (N > 4) point correspondences $\left\{\vec{x}_1^i \leftrightarrow \vec{x}_2^i\right\}$, compute the homography matrix H such that $\vec{x}_2^i = H\vec{x}_1^i$

1. **Normalization of $\vec{x}_1^i$:** normalize the data set $\vec{x}_1^i$ by computing a similarity transformation T such that $\tilde{\vec{x}}_1^i = T\vec{x}_1^i$, where $\tilde{\vec{x}}_1^i$ is the normalized point set with centroid in the image origin (0, 0) and an average distance of the points to the origin equal to $\sqrt{2}$.

2. **Normalization of $\vec{x}_2^i$:** repeat the normalization on the data set $\vec{x}_2^i$ computing T' such that $\tilde{\vec{x}}_2^i = T'\vec{x}_2^i$.

3. **Assemble A_N:** assemble the 2N x 9 matrix A_N as in system (B.8) using the point correspondences $\left\{\tilde{\vec{x}}_1^i \leftrightarrow \tilde{\vec{x}}_2^i\right\}$.

4. **Compute $\tilde{H}$:** compute the singular value decomposition (SVD) of A_N. The solution for $\tilde{\vec{h}}$ is the unit singular vector corresponding to the smallest singular value. The SVD decompose the matrix $A_N = UDV^T$ where D is a diagonal matrix with positive entries arranged in descending order. $\tilde{\vec{h}}$ is the last column of V. The matrix $\tilde{H}$ can be assembled from $\tilde{\vec{h}}$.

5. **Denormalization:** denormalize the homography matrix $H = T^{'-1}\tilde{H}T$.

The DLT algorithm searches for the minimum of the norm of the residual vector $\|\vec{\epsilon}\| = \|A_i\vec{h}\|$ minimizing the norm $\|A_i\vec{h}\|$. The residual vector $\vec{\epsilon}$ is called *algebraic error vector* and the norm $\|\vec{\epsilon}\|$ is a scalar called *algebraic distance*:

$$\|\vec{\epsilon}_i\| = d_{alg}(\vec{x}_2^i, H\vec{x}_1^i) \tag{B.9}$$

For two vectors $\vec{x}_1$ and $\vec{x}_2$, the algebraic distance can be written as:

$$d_{alg}(\vec{x}_2, \vec{x}_1)^2 = a_1^2 + a_2^2 \;\; where \;\; \vec{a} = (a_1,\, a_2,\, a_3)^T = \vec{x}_1 \times \vec{x}_2$$

In general, the algebraic distance does not coincide with the geometric distance. Therefore the minimization of the algebraic distance might not have an intuitive meaning. One would like to achieve a global minimization of the geometric error between points as the intuition might also suggest. However, the minimization of the algebraic distance is in general preferred because it is computationally cheaper then minimizing the geometric distance. The reader interested in finding more details on the techniques used to minimize the geometric error and the relation between the algebraic and geometric error can refer to [30].

B.3 Robust outliers detection algorithm

The RANdom SAmple Consensus (RANSAC) algorithm, first presented in [26], is used to detect and discard outliers from a set of data for model estimation. In the case of odometry estimation, the RANSAC algorithm is an efficient method to determine among the set of point correspondences S the inlier subset S_i and the outlier subset S_o. The homography matrix H will be estimated from the S_i subset or *consensus set*. The goal is to estimate H from a data set containing as few outliers as possible.

A hypothesis for H is computed from a minimal subset s of point correspondences (four in a non-degenerate configuration as explained in section B.1) randomly chosen among the complete set. Then, the hypothesis H is tested on the other points of the set S and a consensus set S_i is incrementally built. New points are added to the consensus set if the residual error from the model H is lower than a predefined threshold t. If the number of points in S_i become greater than a threshold value T then S_i is taken as final consensus set and H is recomputed with the DLT algorithm using the set S_i. On the other hand, if the condition $S_i > T$ is not reached within a maximum number of iterations N, the algorithm ends and H is computed from the largest S_i. It can be observed that the computation is not performed on all possible combinations of minimal subsets of point correspondences since it would be too expensive. Statistical considerations

are applied to find the parameters T and N in order to limit the search to a statistically relevant number of points.

Algorithm 3 presents the basic steps for homography estimation using the RANSAC method [30, 26].

Algorithm 3 ***RANSAC algorithm for homography estimation.***

1. Random selection of a minimal subset of four point correspondences s in a non-degenerate configuration from the complete set S and compute H from s.
2. Determine the set of data point S_i whose residual error (for each point) is within a predefined threshold t from the model H.
3. If the dimension of the inlier set S_i is greater than a threshold T, the algorithm terminates and H can be re-estimated from the inlier set S_i.
4. If the condition $S_i > T$ is not fulfilled, the process repeats again from (1.) with a selection of a new minimal subset s.
5. N is the upper limit for the number of minimal subset s which have to be tested before terminating the algorithm. In case the condition $S_i > T$ is not satisfied, the largest S_i is taken as the largest consensus set and H estimated from it.

In Algorithm 3, three parameters have to be defined: t, T and N. In the remainder of this section, it will be explained how these parameters are computed.

- The threshold t is used in order to decide if a point correspondence belongs to the current hypothesis for H. If the point belongs to the model, it is assumed that the distance error d of the point to the

model is Gaussian with zero mean and standard deviation σ. Then, a value for the threshold t can be determined from statistical considerations. The square of the distance error d^2 is the sum of squared Gaussian variables and it follows the χ^2_m probability distribution with m degrees of freedom. In this case $m = 2$ since a 2D point has two degrees of freedom.

The probability that the value of the random variable χ^2_m is less than k^2 is represented by the cumulative chi-squared distribution defined as:

$$F_m(k^2) = \int_0^{k^2} \chi^2_m(\xi) d\xi \tag{B.10}$$

The inverse cumulative distribution $k^2 = F_m^{-1}(\alpha)$ is a function of the probability α and can be used to calculate the threshold value t. If $\alpha = 0.95$ and $m = 2$ then $k^2 = 5.99$. Finally, a point is considered to be an inlier with 95% probability if $d^2 < 5.99\sigma^2$.

The threshold value t can also be determined empirically.

- The consensus set S_i is considered to be large enough when the number of inliers is equal to the number of inliers believed to be in the data. If ϵ is the fraction of outliers in a data set of dimension n, then $T = (1 - \epsilon)n$ is the sufficient size of S_i to terminate the search.

- If the consensus set does not reach the sufficient dimension T, the search would continue until all the possible minimal subsets s are tested. This could be expensive and unnecessary since, a maximum number of iterations N could guarantee with a certain probability that at least one correct subset s has been tested during the trials.

  Suppose that $w = 1 - \epsilon$ is the fraction of inliers in the data or the probability that a point selected is an inlier. Then, the number N of selections, of 4 points each, can be computed which gives the probability p that at least one correct subset is selected. The relation can be expressed as $(1 - w^s)^N = 1 - p$ which gives:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon^4))} \tag{B.11}$$

A conservative value for the probability $p = 0.99$ is a common choice.

The problem is that in order to compute T and N, the knowledge of the fraction of outliers ϵ is required and such information is not always available. An adaptive estimation of ϵ can be applied starting from a worst case value. Then the value is updated every time a smaller outlier fraction is found during the iterations.

Bibliography

[1] http://www.vision.caltech.edu/bouguetj.

[2] NI Vision Concepts Manual, November 2005.

[3] NI Vision for LabVIEW User Manual, November 2005.

[4] O. Amidi. *An Autonomous Vision-Guided Helicopter*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

[5] D.B. Barber, J.D. Redding, McLain T.W., R.W. Beard, and C.N. Taylor. Vision-based target geo-location using a fixed-wing miniature air vehicle. *Journal of Intelligent and Robotic Systems*, 47(4):361–382, 2006.

[6] N. Bergman. *Recursive Bayesian Estimation, Navigation and Tracking Applications*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 1999.

[7] K. R. Britting. *Inertial Navigation Systems Analysis*. John Wiley & Sons, Inc., 1971.

[8] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, (24):326–376, 1992.

[9] R. S. Bucy and K. D. Senne. Digital synthesis of nonlinear filters. In *Automatica*, number 7, pages 287–298, 1971.

[10] F. Caballero, L. Merino, J. Ferruz, and A. Ollero. A visual odometer without 3d reconstruction for aerial vehicles. applications to building inspection. In *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[11] M.E. Campbell and M. Wheeler. A vision based geolocation tracking system for uavs. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, CO, 2006.

[12] G. Cao, X. Yang, and S. Chen. Robust matching area selection for terrain matching using level set method. In *ICIAR*, pages 423–430, 2005.

[13] G. Conte and P. Doherty. Vision-based unmanned aerial vehicle navigation using geo-referenced information. Accepted for publication in the EURASIP Journal of Advances in Signal Processing. 2009.

[14] G. Conte, S. Duranti, and T. Merz. Dynamic 3D Path Following for an Autonomous Helicopter. In *Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles*, Lisbon, 2004.

[15] G. Conte, M. Hempel, P. Rudol, D. Lundström, S. Duranti, M. Wzorek, and P. Doherty. High accuracy ground target geo-location using autonomous micro aerial vehicle platforms. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, Hawaii, 2008.

[16] V.N. Dobrokhodov, I.I. Kaminer, and K.D. Jones. Vision-based tracking and motion estimation for moving targets using small uavs. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, CO, 2006.

[17] P. Doherty. Advanced Research with Autonomous Unmanned Aerial Vehicles. In *Proc. of the Int. Conf. on the Principles of Knowledge Representation and Reasoning*, pages 731–732, 2004.

[18] P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman, and J. Wiklund. The WITAS unmanned aerial vehicle

project. In *Proc. of the European Conf. on Artificial Intelligence*, pages 747–755, 2000.

[19] P. Doherty, P. Haslum, F. Heintz, T. Merz, T. Persson, and B. Wingman. A Distributed Architecture for Autonomous Unmanned Aerial Vehicle Experimentation. In *Proc. of the Int. Symp. on Distributed Autonomous Robotic Systems*, pages 221–230, 2004.

[20] S. Duranti and G. Conte. In-flight identification of the augmented flight dynamics of the rmax unmanned helicopter. In *Proc. of the 17th IFAC Symposium on Automatic Control in Aerospace*, 2007.

[21] S. Duranti, G. Conte, D. Lundström, P. Rudol, M. Wzorek, and P. Doherty. Linkmav, a prototype rotary wing micro aerial vehicle. In *Proc. of 17th IFAC Symposium on Automatic Control in Aerospace*, Toulouse, France, June 2007.

[22] M. Egerstedt, X. Hu, and A. Stotsky. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control*, 46(11):1777–1782, November 2001.

[23] M. Egerstedt, T. J. Koo, F. Hoffmann, and S. Sastry. Path planning and flight controller scheduling for an autonomous helicopter. *Lecture Notes in Computer Science*, 1569:91–102, 1999.

[24] B. Etkin and L.D. Reid. *Dynamic of Flight: Stability and Control*. John Wiley and Sons, Inc., 1995.

[25] J.A. Farrell and M. Barth. *The Global Positioning System and Inertial Navigation*. McGraw-Hill, 1998.

[26] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, June 1981.

[27] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, 2001.

[28] N. Frietsch, O. Meister, C. Schlaile, J. Seibold, and G.F. Trommer. Vision based hovering and landing system for a vtol-mav with geo-localization capabilities. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008.

[29] E.L. Hall, D.L. Davies, and M.E. Casey. The selection of critical subsets for signal, image, and scene matching. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI-2, jul 1980.

[30] R. Hartley and A. Zisserman. *Multiple View Geometry, Second Edition*. Cambridge University Press, Cambridge, 2003.

[31] Object Computing Inc. Tao developers guide, version 1.1a. 2000.

[32] C. James *et. al.* Vulnerability assessment of the transportation infrastructure relying on the global positioning system. Technical report, Volpe National Transportation Systems Center, US Department of Transportation, August 2001.

[33] A. E. Johnson and J. F. Montgomery. Overview of terrain relative navigation approaches for precise lunar landing. In *Proc. of the IEEE Aerospace Conference*, Big Sky, Montana, March 2008.

[34] R. Karlsson, T. B. Schön, D. Törnqvist, G. Conte, and F. Gustafsson. Utilizing model structure for efficient simultaneous localization and mapping for a uav application. In *Proc. of the IEEE Aerospace Conference*, Big Sky, Montana, March 2008.

[35] J. Kim and S. Sukkarieh. Real-time implementation of airborne inertial-slam. *Robot. Auton. Syst.*, 55(1):62–71, 2007.

[36] T. Lemaire, C. Berger, I.K. Jung, and S. Lacroix. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364, September 2007.

[37] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[38] R. Madison, P. DeBitetto, A.R. Olean, and M. Peebles. Target geolocation from a small unmanned aircraft system. In *IEEE Aerospace Conference*, Big Sky, Montana, March 2008.

[39] P. Mantegazza *et. al.* RTAI: Real time application interface. *Linux Journal*, 72, April 2000.

[40] P.S. Maybeck. *Stochastic Models, Estimation and Control: Volume 1.* Academic Press, New York, 1979.

[41] S.S. Mehta, W.E. Dixon, D. MacArthur, and C.D. Crane. Visual servo control of an unmanned ground vehicle via a moving airborne monocular camera. In *2006 American Control Conference*, Minneapolis, USA, June 2006.

[42] T. Merz. Building a System for Autonomous Aerial Robotics Research. In *Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles*, 2004.

[43] T. Merz, S. Duranti, and G. Conte. Autonomous landing of an unmanned aerial helicopter based on vision and inertial sensing. In *Proc. of the 9th International Symposium on Experimental Robotics (ISER)*, Singapore, 2004.

[44] B. Mettler, M.B. Tischler, and T. Kanade. System identification of small-size unmanned helicopter dynamics. *American Helicopter Society 55th Annual Forum Proceedings*, May 1999.

[45] E. Michaelsen, M. Kirchhoff, and U. Stilla. Sensor pose inference from airborne videos by decomposing homography estimates. *Altan MO (ed) International Archives of Photogrammetry and Remote Sensing*, 35:1–6, 2004.

[46] M. Pachter, N. Ceccarelli, and P.R. Chandler. Vision-based target geolocation using feature tracking. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007.

[47] P-O. Pettersson. *Using Randomized Algorithms for Helicopter Path Planning.* Linköping, Sweden, 2006. Lic. Thesis Linköping University.

[48] P-O. Pettersson and P. Doherty. Probabilistic Roadmap Based Path Planning for an Autonomous Unmanned Aerial Vehicle. In *Proc. of the ICAPS-04 Workshop on Connecting Planning Theory with Practice*, 2004.

[49] W. K. Pratt. *Digital Image Processing, 2nd ed.* Wiley, New York, 1991.

[50] R.W. Prouty. *Helicopter Performance, Stability and Control.* Krieger Publishing Company, 1995.

[51] D.F. Rogers and J.A. Adams. *Mathematical Elements for Computer Graphics.* McGraw-Hill, 1990.

[52] R. M. Rogers. *Applied Mathematics in Integrated Navigation Systems.* American Institute of Aeronautics and Astronautics, Inc., 2000.

[53] R. T. Rysdyk. Uav path following for target observation in wind. *To appear in AIAA Journal of Guidance, Control, and Dynamics.*

[54] S. Saripalli, J.F. Montgomery, and G. Sukhatme. Visually-guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(2):371–380, June 2003.

[55] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, 1994.

[56] E-H. Shin. *Accuracy improvement of low cost INS/GPS for land applications.* Calgary, Alberta, 2001. Master Thesis, University of Calgary.

[57] D. G. Sim, R. H. Park, R. C. Kim, S. U. Lee, and I. C. Kim. Integrated position estimation using aerial image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):1–18, 2002.

[58] R. Skjetne and T. Fossen. Nonlinear maneuvering and control of ships. *MTS/IEEE OCEANS 2001*, 3:1808–15, 2001.

[59] M. Svensson. Aircraft Trajectory Restoration by Integration of Inertial Measurements and GPS. *Master Thesis, Linköping University*, 1999.

[60] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The MIT Press, Cambridge, MA, USA, 2005.

[61] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[62] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, J. F. Montgomery, A. Ansar, and L. H. Matthies. Coupled vision and inertial navigation for pin-point landing. In *NASA Science and Technology Conference*, 2007.

[63] M. Wzorek, G. Conte, P. Rudol, T. Merz, S. Duranti, and P. Doherty. From motion planning to control - a navigation framework for an autonomous unmanned aerial vehicle. *21th Bristol UAV Systems Conference*, April 2006.

[64] M. Wzorek and P. Doherty. Reconfigurable Path Planning for an Autonomous Unmanned Aerial Vehicle. In *Proceedings of the International Conference on Hybrid Information Technology, ICHIT*, 2006.

[65] G. Zhang and L. Shen. Rule-based expert system for selecting scene matching area. In *Intelligent Control and Automation*, volume 344, pages 546–553. Springer Berlin, 2006.

[66] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, (21):977–1000, 2003.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall:** Semantisk analys av process-beskrivningar i naturligt språk, 1977, ISBN 91-7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91-7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91-7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation,1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372-805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov***:* Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies,1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity

of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2.

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani**: Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms,1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund**: Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld**: Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén**: Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Langugaes from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis,1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and

Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi**: Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

No 613 **Man Lin:** Analysis and Synthesis of Reactive Systems: A Generic Layered Architecture Perspective, 1999, ISBN 91-7219-630-0.

No 618 **Jimmy Tjäder:** Systemimplementering i praktiken - En studie av logiker i fyra projekt, 1999, ISBN 91-7219-657-2.

No 627 **Vadim Engelson:** Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, 2000, ISBN 91-7219-709-9.

No 637 **Esa Falkenroth:** Database Technology for Control and Simulation, 2000, ISBN 91-7219-766-8.

No 639 **Per-Arne Persson:** Bringing Power and Knowledge Together: Information Systems Design for Autonomy and Control in Command Work, 2000, ISBN 91-7219-796-X.

No 660 **Erik Larsson:** An Integrated System-Level Design for Testability Methodology, 2000, ISBN 91-7219-890-7.

No 688 **Marcus Bjäreland:** Model-based Execution Monitoring, 2001, ISBN 91-7373-016-5.

No 689 **Joakim Gustafsson:** Extending Temporal Action Logic, 2001, ISBN 91-7373-017-3.

No 720 **Carl-Johan Petri:** Organizational Information Provision - Managing Mandatory and Discretionary Use of Information Technology, 2001, ISBN-91-7373-126-9.

No 724 **Paul Scerri:** Designing Agents for Systems with Adjustable Autonomy, 2001, ISBN 91 7373 207 9.

No 725 **Tim Heyer**: Semantic Inspection of Software Artifacts: From Theory to Practice, 2001, ISBN 91 7373 208 7.

No 726 **Pär Carlshamre:** A Usability Perspective on Requirements Engineering - From Methodology to Product Development, 2001, ISBN 91 7373 212 5.

No 732 **Juha Takkinen:** From Information Management to Task Management in Electronic Mail, 2002, ISBN 91 7373 258 3.

No 745 **Johan Åberg:** Live Help Systems: An Approach to Intelligent Help for Web Information Systems, 2002, ISBN 91-7373-311-3.

No 746 **Rego Granlund:** Monitoring Distributed Teamwork Training, 2002, ISBN 91-7373-312-1.

No 757 **Henrik André-Jönsson:** Indexing Strategies for Time Series Data, 2002, ISBN 917373-346-6.

No 747 **Anneli Hagdahl:** Development of IT-suppor-ted Inter-organisational Collaboration - A Case Study in the Swedish Public Sector, 2002, ISBN 91-7373-314-8.

No 749 **Sofie Pilemalm:** Information Technology for Non-Profit Organisations - Extended Participatory Design of an Information System for Trade Union Shop Stewards, 2002, ISBN 91-7373-318-0.

No 765 **Stefan Holmlid:** Adapting users: Towards a theory of use quality, 2002, ISBN 91-7373-397-0.

No 771 **Magnus Morin:** Multimedia Representations of Distributed Tactical Operations, 2002, ISBN 91-7373-421-7.

No 772 **Pawel Pietrzak:** A Type-Based Framework for Locating Errors in Constraint Logic Programs, 2002, ISBN 91-7373-422-5.

No 758 **Erik Berglund:** Library Communication Among Programmers Worldwide, 2002, ISBN 91-7373-349-0.

No 774 **Choong-ho Yi:** Modelling Object-Oriented Dynamic Systems Using a Logic-Based Framework, 2002, ISBN 91-7373-424-1.

No 779 **Mathias Broxvall:** A Study in the Computational Complexity of Temporal Reasoning, 2002, ISBN 91-7373-440-3.

No 793 **Asmus Pandikow:** A Generic Principle for Enabling Interoperability of Structured and Object-Oriented Analysis and Design Tools, 2002, ISBN 91-7373-479-9.

No 785 **Lars Hult:** Publika Informationstjänster. En studie av den Internetbaserade encyklopedins bruksegenskaper, 2003, ISBN 91-7373-461-6.

No 800 **Lars Taxén:** A Framework for the Coordination of Complex Systems´ Development, 2003, ISBN 91-7373-604-X

No 808 **Klas Gäre:** Tre perspektiv på förväntningar och förändringar i samband med införande av informa-

tionsystem, 2003, ISBN 91-7373-618-X.

No 821 **Mikael Kindborg:** Concurrent Comics - programming of social agents by children, 2003, ISBN 91-7373-651-1.

No 823 **Christina Ölvingson:** On Development of Information Systems with GIS Functionality in Public Health Informatics: A Requirements Engineering Approach, 2003, ISBN 91-7373-656-2.

No 828 **Tobias Ritzau:** Memory Efficient Hard Real-Time Garbage Collection, 2003, ISBN 91-7373-666-X.

No 833 **Paul Pop:** Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems, 2003, ISBN 91-7373-683-X.

No 852 **Johan Moe:** Observing the Dynamic Behaviour of Large Distributed Systems to Improve Development and Testing - An Emperical Study in Software Engineering, 2003, ISBN 91-7373-779-8.

No 867 **Erik Herzog:** An Approach to Systems Engineering Tool Data Representation and Exchange, 2004, ISBN 91-7373-929-4.

No 872 **Aseel Berglund:** Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV, 2004, ISBN 91-7373-940-5.

No 869 **Jo Skåmedal:** Telecommuting's Implications on Travel and Travel Patterns, 2004, ISBN 91-7373-935-9.

No 870 **Linda Askenäs:** The Roles of IT - Studies of Organising when Implementing and Using Enterprise Systems, 2004, ISBN 91-7373-936-7.

No 874 **Annika Flycht-Eriksson:** Design and Use of Ontologies in Information-Providing Dialogue Systems, 2004, ISBN 91-7373-947-2.

No 873 **Peter Bunus:** Debugging Techniques for Equation-Based Languages, 2004, ISBN 91-7373-941-3.

No 876 **Jonas Mellin:** Resource-Predictable and Efficient Monitoring of Events, 2004, ISBN 91-7373-956-1.

No 883 **Magnus Bång:** Computing at the Speed of Paper: Ubiquitous Computing Environments for Healthcare Professionals, 2004, ISBN 91-7373-971-5

No 882 **Robert Eklund:** Disfluency in Swedish human-human and human-machine travel booking dialogues, 2004. ISBN 91-7373-966-9.

No 887 **Anders Lindström:** English and other Foreign Linquistic Elements in Spoken Swedish. Studies of Productive Processes and their Modelling using Finite-State Tools, 2004, ISBN 91-7373-981-2.

No 889 **Zhiping Wang:** Capacity-Constrained Production-inventory systems - Modellling and Analysis in both a traditional and an e-business context, 2004, ISBN 91-85295-08-6.

No 893 **Pernilla Qvarfordt:** Eyes on Multimodal Interaction, 2004, ISBN 91-85295-30-2.

No 910 **Magnus Kald:** In the Borderland between Strategy and Management Control - Theoretical Framework and Empirical Evidence, 2004, ISBN 91-85295-82-5.

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN: 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005. ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8.

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

No 1120 **Thomas Gustafsson:** Management of Real-Time Data Consistency and Transient Overloads in Embedded Systems, 2007, ISBN 978-91-85831-33-3.

No 1127 **Alexandru Andrei:** Energy Efficient and Predictable Design of Real-time Embedded Systems, 2007, ISBN 978-91-85831-06-7.

No 1139 **Per Wikberg:** Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions, 2007, ISBN 978-91-85895-66-3.

No 1143 **Mehdi Amirijoo:** QoS Control of Real-Time Data Services under Uncertain Workload, 2007, ISBN 978-91-85895-49-6.

No 1150 **Sanny Syberfeldt:** Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases, 2007, ISBN 978-91-85895-27-4.

No 1155 **Beatrice Alenljung:** Envisioning a Future Decision Support System for Requirements Engineering - A Holistic and Human-centred Perspective, 2008, ISBN 978-91-85895-11-3.

No 1156 **Artur Wilk**: Types for XML with Application to Xcerpt, 2008, ISBN 978-91-85895-08-3.

No 1183 **Adrian Pop:** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages, 2008, ISBN 978-91-7393-895-2.

No 1185 **Jörgen Skågeby:** Gifting Technologies - Ethnographic Studies of End-users and Social Media Sharing, 2008, ISBN 978-91-7393-892-1.

No 1187 **Imad-Eldin Ali Abugessaisa:** Analytical tools and information-sharing methods supporting road safety organizations, 2008, ISBN 978-91-7393-887-7.

No 1204 **H. Joe Steinhauer:** A Representation Scheme for Description and Reconstruction of Object Configurations Based on Qualitative Relations, 2008, ISBN 978-91-7393-823-5.

No 1222 **Anders Larsson:** Test Optimization for Core-based System-on-Chip, 2008, ISBN 978-91-7393-768-9.

No 1238 **Andreas Borg:** Processes and Models for Capacity Requirements in Telecommunication Systems, 2009, ISBN 978-91-7393-700-9.

No 1240 **Fredrik Heintz:** DyKnow: A Stream-Based Knowledge Processing Middleware Framework, 2009, ISBN 978-91-7393-696-5.

No 1241 **Birgitta Lindström:** Testability of Dynamic Real-Time Systems, 2009, ISBN 978-91-7393-695-8.

No 1244 **Eva Blomqvist:** Semi-automatic Ontology Construction based on Patterns, 2009, ISBN 978-91-7393-683-5.

No 1249 **Rogier Woltjer:** Functional Modeling of Constraint Management in Aviation Safety and Command and Control, 2009, ISBN 978-91-7393-659-0.

No 1260 **Gianpaolo Conte:** Vision-Based Localization and Guidance for Unmanned Aerial Vehicles, 2009, ISBN 978-91-7393-603-3.

Linköping Studies in Statistics

No 9 **Davood Shahsavani:** Computer Experiments Designed to Explore and Approximate Complex Deterministic Models, 2008, ISBN 978-91-7393-976-8.

No 10 **Karl Wahlin:** Roadmap for Trend Detection and Assessment of Data Quality, 2008, ISBN: 978-91-7393-792-4

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering- att skapa samstämmighet mellan informa-tionssystemarkitektur och verksamhet, 1998. ISBN-9172-19-296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998. ISBN-9172-19-299-2.

No 3 **Anders Avdic:** Användare och utvecklare - om anveckling med kalkylprogram, 1999. ISBN-91-7219-606-8.

No 4 **Owen Eriksson:** Kommunikationskvalitet hos informationssystem och affärsprocesser, 2000. ISBN 91-7219-811-7.

No 5 **Mikael Lind:** Från system till process - kriterier för processbestämning vid verksamhetsanalys, 2001, ISBN 91-7373-067-X

No 6 **Ulf Melin:** Koordination och informationssystem i företag och nätverk, 2002, ISBN 91-7373-278-8.

No 7 **Pär J. Ågerfalk:** Information Systems Actability - Understanding Information Technology as a Tool for Business Action and Communication, 2003, ISBN 91-7373-628-7.

No 8 **Ulf Seigerroth:** Att förstå och förändra systemutvecklingsverksamheter - en taxonomi för metautveckling, 2003, ISBN91-7373-736-4.

No 9 **Karin Hedström:** Spår av datoriseringens värden - Effekter av IT i äldreomsorg, 2004, ISBN 91-7373-963-4.

No 10 **Ewa Braf:** Knowledge Demanded for Action - Studies on Knowledge Mediation in Organisations, 2004, ISBN 91-85295-47-7.

No 11 **Fredrik Karlsson:** Method Configuration - method and computerized tool support, 2005, ISBN 91-85297-48-8.

No 12 **Malin Nordström:** Styrbar systemförvaltning - Att organisera systemförvaltningsverksamhet med hjälp av effektiva förvaltningsobjekt, 2005, ISBN 91-85297-60-7.

No 13 **Stefan Holgersson:** Yrke: POLIS - Yrkeskunskap, motivation, IT-system och andra förutsättningar för polisarbete, 2005, ISBN 91-85299-43-X.

No 14 **Benneth Christiansson, Marie-Therese Christiansson:** Mötet mellan process och komponent - mot ett ramverk för en verksamhetsnära kravspecifikation vid anskaffning av komponentbaserade informationssystem, 2006, ISBN 91-85643-22-X.