Linköping Studies in Science and Technology

Thesis No. 1307

Navigation Functionalities for an Autonomous UAV Helicopter

by

Gianpaolo Conte

Submitted to Linköping Institute of Technology at Linköping University in partial fulfillment of the requirements for degree of Licentiate of Engineering

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

Linköping 2007

Navigation Functionalities for an Autonomous UAV Helicopter

by

Gianpaolo Conte

March 2007
ISBN 978-91-85715-35-0
Linköping Studies in Science and Technology
Thesis No. 1307
ISSN 0280-7971
LiU-Tek-Lic-2007:16

ABSTRACT

This thesis was written during the WITAS UAV Project where one of the goals has been the development of a software/hardware architecture for an unmanned autonomous helicopter, in addition to autonomous functionalities required for complex mission scenarios. The algorithms developed here have been tested on an unmanned helicopter platform developed by Yamaha Motor Company called the RMAX.

The character of the thesis is primarily experimental and it should be viewed as developing navigational functionality to support autonomous flight during complex real-world mission scenarios. This task is multidisciplinary since it requires competence in aeronautics, computer science and electronics.

The focus of the thesis has been on the development of a control method to enable the helicopter to follow 3D paths. Additionally, a helicopter simulation tool has been developed in order to test the control system before flight-tests. The thesis also presents an implementation and experimental evaluation of a sensor fusion technique based on a Kalman filter applied to a vision based autonomous landing problem. Extensive experimental flight-test results are presented.

The work in this thesis is supported in part by grants from the Wallenberg Foundation, the SSF MOVIII strategic center and an NFFP04-031 "Autonomous flight control and decision making capabilities for Mini-UAVs" project grant.

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

Acknowledgements

This work would have not been possible without the support and help of my family, colleagues and friends. I would like to thanks all of them here.

Especially I would like to thanks:

My supervisor Patrick Doherty who has believed in me supporting my research and helping me in the time of darkness.

Simone Duranti for too many things: for the exciting and productive discussions about (but not only) UAVs, for his intuitions and many inputs to my research and for sharing a common point of view on the definition of ”good food”.

My colleagues Piotr Rudol, Fredrik Heintz, Per-Magnus Olsson, Simone Duranti and Per Nyblom to have spent part of their time to go through my thesis and giving me vital feedback.

Piotr and Mariusz for their effort, time and especially extra time put into the project, for making the robot-lab never boring and for keeping the robot-lab fridge never empty on friday.

All the AIICS members for being available at anytime and for making the working environment always friendly.

Felix and Dave with whom I share the hard task of surviving the Swedish winter, I have to say they make it so much easier.

I also thanks the swedish authority for all the presents left under the windshield wipers of my car, especially for not enforcing to pay them.

My family and Roberta for being there.

"To fly is my religion."
Richard Bach

To Antonio.

Preface

The work presented in this thesis was done as part of the requirement for a Licentiate degree at the Artificial Intelligence and Integrated Computer System (AIICS) division at Linköping University. The focus of the thesis has been on development of navigation functionalities for an unmanned helicopter. A flight control mode which enables an unmanned helicopter to follow 3D paths has been developed (Paper I, Paper II). Additionally, a sensor fusion technique has been applied to a vision based autonomous landing problem (Paper III).

The original refereed and published papers upon which this thesis is based are included as an appendix to this thesis:

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

Paper II *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 III *T. Merz, S. Duranti, and G. Conte.* Autonomous landing of an unmanned aerial helicopter based on vision and inertial sensing. *Proc. of the 9th International Symposium on Experimental Robotics, 2004.*

Linköping, January 2007

Gianpaolo Conte

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 grown in the direction of civil applications as a consequence of the cost reduction of this technology.

Aircraft navigation can be accomplished by safely solving four tasks: 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. In order to replace the pilot completely a number of problems have to be solved. For example, it is difficult to replace the skills of a pilot in perceiving and avoiding obstacles. This is one of the reasons why the introduction of UAVs in non-segregated airspace still represents a challenge.

The work presented in this thesis was initiated as part of the WITAS UAV Project [17, 2], where the main goal was to develop technologies and functionalities necessary for the successful deployment of a fully au-

tonomous 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.

Many universities [26, 28, 1, 19, 10, 27, 29] have been and continue to do research with autonomous helicopter systems. Most of the research has focused on low-level control of such systems with less emphasis on high-autonomy as in the WITAS UAV Project.

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 [3, 15].

This thesis focuses on two aspects of the navigation task: *flight control* and *state estimation*. The main results of this thesis have been published in Paper I and Paper II (see Appendix) which are related to flight control issues, and Paper III (see Appendix) related to sensor fusion and state estimation applied to the problem of a vision based autonomous landing. The algorithms presented are implemented and tested on a UAV helicopter and currently in use in the context of a number of autonomous UAV missions.

The thesis also presents a simulation tool developed for control system validation. The simulator is based on system identification work described in [4]. The helicopter simulator has been an invaluable tool for development of flight control modes. It is implemented in the C-language and coupled to the complete software architecture. Simulation tests can be done in real-time and with actual helicopter hardware-in-the-loop. A complete flight mission can be tested in the field with the helicopter hardware in-the-loop before the actual flight test.

The first contribution of this thesis (Paper I) is the development of a Path Following Control Mode (PFCM). This control mode enables the 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. As stated before, in order to achieve this task safely the helicopter must perceive and than avoid eventual obstacles on its way. Additionally, it must stay within the allowed flight control 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 are in [18]. The PFCM presented in this thesis, together with the path execution mechanism, gives the right balance between safety and flexibility and can also be used for dynamic path planning. Recently some experiments have been made to exploit the possibility of avoiding *no-fly-zones* which are added dynamically from a ground control station or other sources during flight. These experiments are intended to provide more dynamic path planning. Details of these experiments are available in [30]. A natural extension of this feature is the replacement of the manual selection of no-fly-zones with an automatic selection. Provided the UAV has the capability to detect obstacles in the environment, these can be treated as no-fly-zones and avoided by using the existing dynamic path planning capability.

Paper II contains an extended description of the WITAS UAV navigation framework and it shows how the PFCM is integrated in the software architecture and used in complex missions.

The second contribution of the thesis is the integration of a Kalman Filter (KF) into the control system architecture. The KF is used to fuse the information coming from different sensors available on-board the UAV in order to estimate the helicopter state. The estimated state is used by the control system to control the helicopter. The reason why several sensors are fused together is because each sensor has different properties which should be taken advantage of. In this thesis the KF is used for fusing inertial sensors (INS) and a vision sensor for the vision based autonomous landing problem described in Paper III. The same KF is implemented on the UAV helicopter for fusing GPS together with inertial sensors. The reuse of software for different applications is highly desirable in such a complex system such as an autonomous UAV.

Vision based autonomous landing is a challenging problem. Landing is the most dangerous phase of a flight, so the state estimation of a UAV must be accurate and continuous in time. The autonomous landing problem is solved by using high accuracy position data from a vision system which uses a single camera and inertial data taken from an Inertial Measurement Unit (IMU). What makes this application special is the fact that the GPS is not used at all during autonomous landing. This makes the UAV independent

of external position signals.

The UAV platform used for the experimentation is an 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 (Fig.1.1). The area is used mainly for fire fighter training and includes a number of building structures and road network. 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.

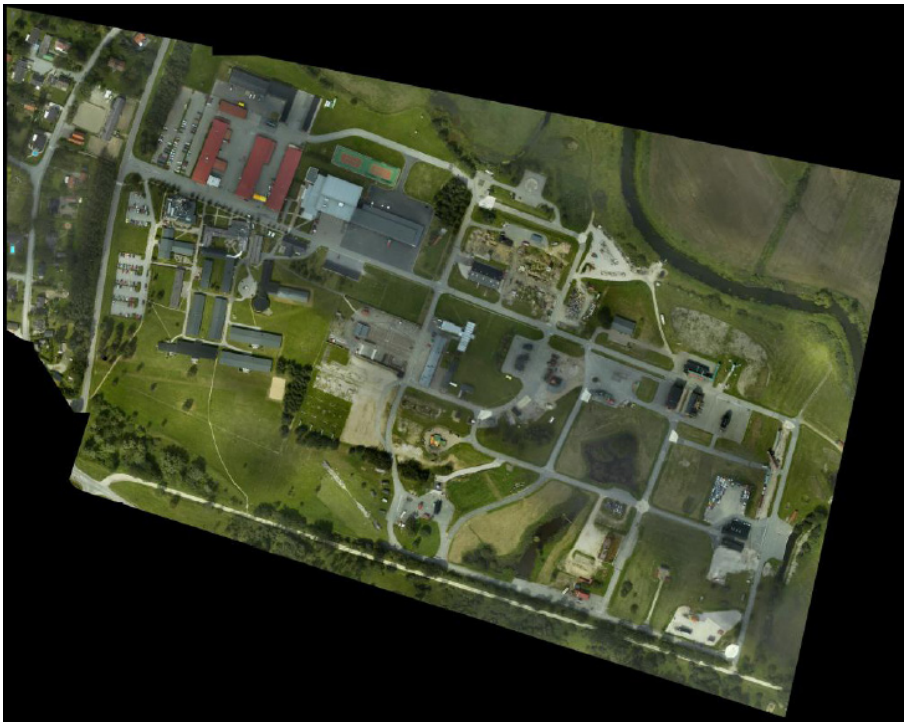


Figure 1.1: Orthophoto of the flight-test area in Revinge (south of Sweden).

Chapter 2

Overview

This chapter presents an overview of the UAV architecture developed during the WITAS UAV Project. The components which have been developed and are part of this thesis will be considered in this context.

A description of the UAV helicopter platform used for the experimental tests will also be presented. Details of the platform which are relevant to this thesis will also be considered in order to gain a better understanding of the components developed.

2.1 UAV software architecture

The software architecture developed during the WITAS UAV Project is a complex distributed architecture for high level autonomous missions. The architecture enables the UAV to perform so-called *push button mission*. This is intended to mean that a UAV is capable of planning and executing a 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 photogrammetry scenario. In such a 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. The details of this mission are described in Paper II. To accomplish this kind of mission, an architecture

which integrates many different functionalities is required.

Fig. 2.1 provides a schematic of the software architecture that has been developed. The modules focused on in this thesis have been emphasized in the schematic. The boxes emphasized with bold lines and text have been developed and made operational and will be described in detail in this thesis.

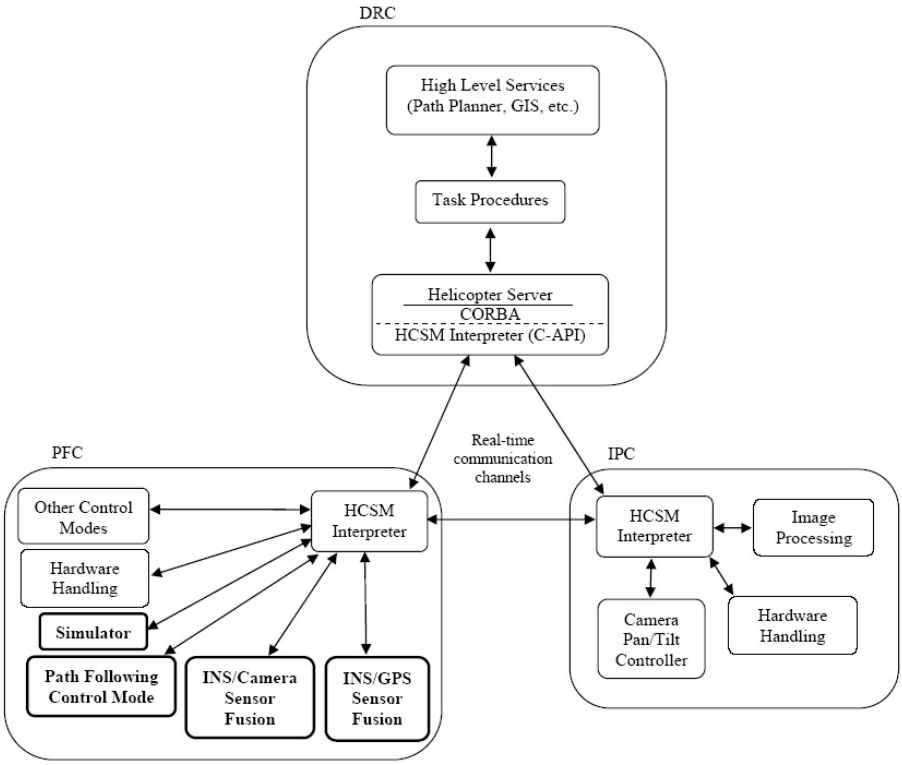


Figure 2.1: UAV software architecture schematic. The modules emphasized using bold lines are the focus of this thesis.

The software architecture has been implemented using three computers

which will be described shortly.

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

The PFC executes predominantly hard real-time tasks such as the flight control modes or the sensor fusion algorithms. This part of the system uses a Real-Time Application Interface (RTAI) [14] 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 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 [11] is in use. More details about the complete software architecture can be found in Paper III and in [3, 15].

As can be observed from the boxes emphasized in Fig. 2.1, this thesis deals with a number of functionalities which are contained in the PFC system. A brief introduction as to what these functionalities actually do will now be described.

The *simulator* mentioned in the introduction implements the helicopter dynamics. Moreover it emulates the helicopter sensor outputs so that during the simulation the complete software architecture can be tested in a closed loop. The simulator can also run on the on board hardware so that during the simulation it is possible to see the helicopter actuators moving as they would in actual flight. Such simulation using hardware-in-the-loop

provides a powerful way to quickly test the correct functioning of all the components on the field, including the interface to the helicopter.

The *path following control mode* (PFCM) enables the helicopter to follow a 3D path. It receives a geometric description of a path via a number of parameters from the path planner. From these parameters it generates a continuous description of the path, then it adds velocity and acceleration constraints in order to calculate the inputs to be sent to the helicopter. An algorithm which generates the proper control inputs at each control cycle has been developed. The PFCM also generates a number of events and other outputs which are necessary for monitoring purposes in flight.

The two *sensor fusion* algorithms implemented are based on a Kalman filter and produce estimates of the helicopter state in terms of position, velocity and attitude. They are used to fuse the inertial sensors with the GPS, and the inertial sensors with the video camera during autonomous landing. Since they are based on the same dynamic model, only the results of the integration between inertial sensors and video camera will be presented. It will be shown how the integration of the video camera with inertial sensors can effectively improve the robustness of the helicopter state estimate compared to a solution obtained from image processing alone.

2.2 The UAV helicopter platform

The Yamaha RMAX helicopter used for the experimentation (Fig. 2.2) 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 Fig.2.3). 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 2.2: 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 reduced mechanical complexity in the system.

Figure 2.3: 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 3.3 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 process which will be described in chapter 5.

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 as an attitude stabilization system.

The hardware platform developed during the WITAS UAV Project is integrated with the Yamaha platform as shown in Fig. 2.4. It is based on three PC104 embedded computers.

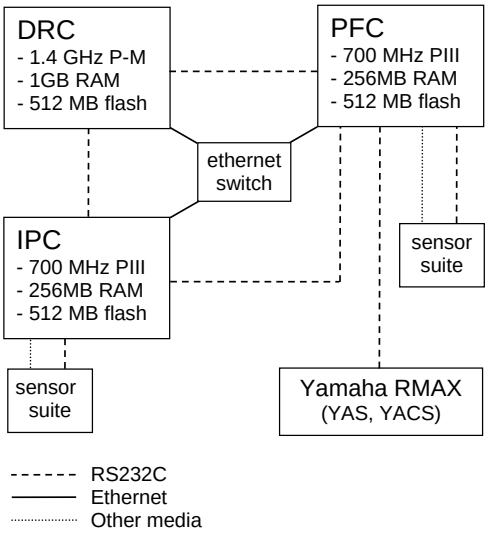


Figure 2.4: On-board hardware schematic.

The PFC is implemented on a 700Mhz Pentium III and includes a wireless Ethernet bridge, a GPS receiver and a barometric altitude sensor. An earlier version of the system included a compass as a heading sensor source. The compass has been removed from the latest version since the Kalman filter which fuses the inertial sensors and GPS provides sufficiently stable heading information. The PFC communicates with the helicopter through a serial line RS232C, where the inertial sensor data from the YAS is passed to the PFC. The PFC can also send control inputs to the YACS for helicopter control. The IPC runs on a second PC104 embedded computer (PIII 700MHz), and 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 all high-end autonomous functionality. 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.

Chapter 3

Simulation

3.1 Introduction

This chapter describes the simulation tool which has been used to develop and test the control system for 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 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 Hardware-in-the-loop simulation

Two versions of the simulator have been developed. A non real-time version exists which is 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 also exists which is used to test the complete UAV architecture, where simulations are performed with the helicopter *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 20ms when the system is in simulation mode. Fig. 3.1 shows the hardware/software components involved in the 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.

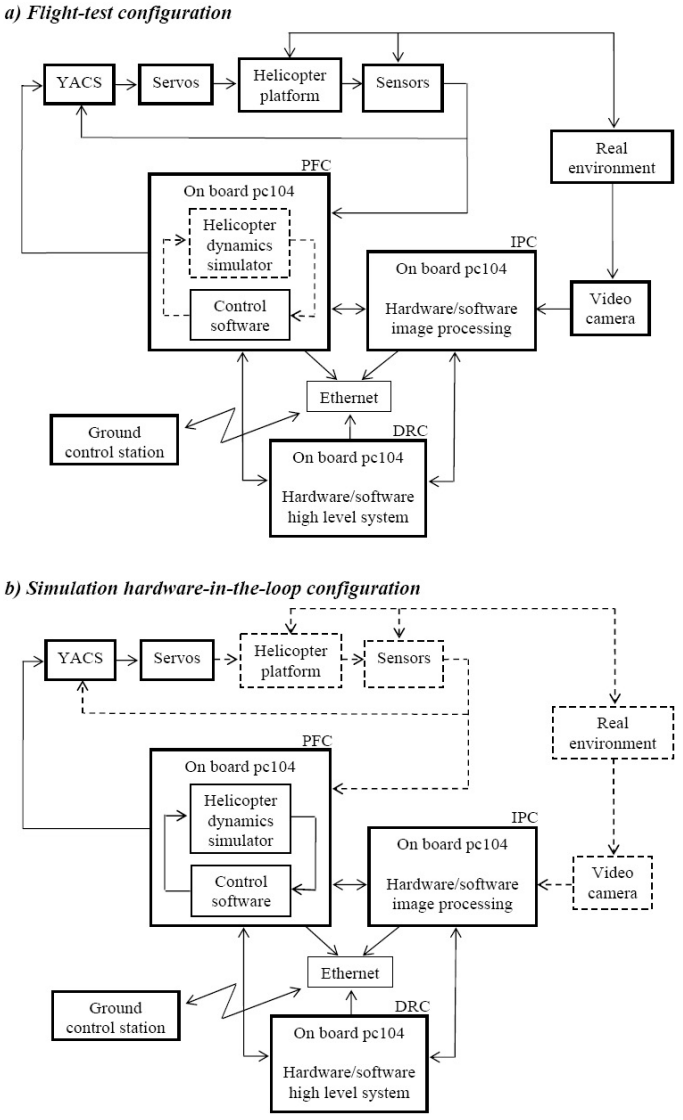


Figure 3.1: Figure a) depicts the hardware/software architecture in flight-test configuration. Figure b) depicts the architecture during the hardware-in-the-loop simulation.

Currently, the video camera is not used in the simulation loop but the system can still control the camera pan/tilt. A virtual environment (Fig. 3.2), reproducing the flight-test area described in the introduction (Fig.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.

Figure 3.2: Virtual environment used for flight mission simulation.

3.3 Reference frames

This section provides an overview of the different reference frames used in this thesis.

The *Earth frame* (Fig. 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 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 frame* (Fig. 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.

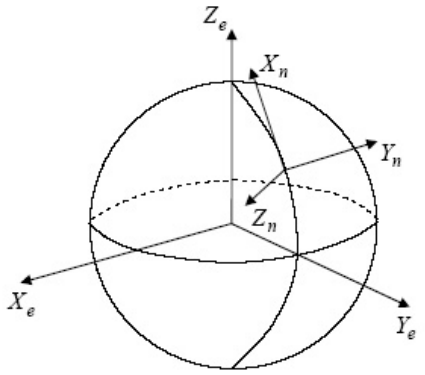


Figure 3.3: Earth and navigation frames.

The *body frame* (Fig. 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 nose, the y-axis orthogonal to the Y_b axis and pointing to the right side of the helicopter body, and the Z_b axis pointing down so that

it's 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.

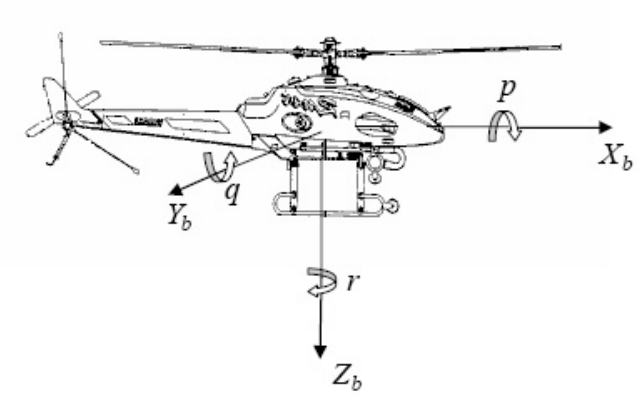


Figure 3.4: Body frame.

In order to transform a vector from the body frame to the navigation frame a rotation matrix has to be applied:

$$C_b^n = \begin{bmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\phi cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \quad (3.1)$$

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

$$C_n^b = (C_b^n)^T \quad (3.2)$$

The transformation matrix 3.1 and 3.2 will be used later in the thesis.

3.4 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 [4]. 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.

3.4.1 Augmented helicopter attitude dynamics

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

The equations 3.3 represent the four input/output transfer functions in the Laplace domain:

$$\begin{aligned}
\Delta\Phi &= \frac{2.3(s^2+3.87s+53.3)}{(s^2+6.29s+16.2)(s^2+8.97s+168)}\Delta AIL \\
\Delta\Theta &= \frac{0.5(s^2+9.76s+75.5)}{(s^2+3s+5.55)(s^2+2.06s+123.5)}\Delta ELE \\
\Delta R &= \frac{9.7(s+12.25)}{(s+4.17)(s^2+3.5s+213.4)}\Delta RUD \\
\Delta A_Z &= \frac{0.0828s(s+3.37)}{(s+0.95)(s^2+13.1s+214.1)}\Delta THR
\end{aligned} \tag{3.3}$$

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 unit 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.

3.4.2 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 [7]. The first set represents the translational dynamics along the three body axes:

$$\begin{aligned} X &= m(\dot{u} + qw - rv) + mgsin\theta \\ Y &= m(\dot{v} + ru - pw) - mgcos\theta sin\phi \\ Z &= m(\dot{w} + pv - qu) - mgcos\theta cos\phi \end{aligned} \tag{3.4}$$

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:

$$\begin{aligned} L &= I_x\dot{p} - (I_y - I_z)qr \\ M &= I_y\dot{q} - (I_z - I_x)rp \\ N &= I_z\dot{r} - (I_x - I_y)pq \end{aligned} \tag{3.5}$$

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:

$$\begin{aligned} \dot{\phi} &= p + qsin\phi tan\theta + rcos\phi tan\theta \\ \dot{\theta} &= qcos\phi - rsin\phi \\ \dot{\psi} &= qsin\phi sec\theta + rcos\phi sec\theta \end{aligned} \tag{3.6}$$

These three sets of nonlinear equations are valid for a generic aircraft. The transfer functions in 3.3 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 3.3 we derive $\phi(t)$, $\theta(t)$ and $r(t)$ which can be used in 3.6 in order to find the other parameters $p(t)$, $q(t)$, $\psi(t)$.

The equations in 3.5 will not be used in the model because the dynamics represented by these equations is contained in the first three transfer functions in 3.3. The motion equations in 3.4 can be rewritten as follows:

$$\begin{aligned} \dot{u} &= F_x - qw + rv - gsin\theta \\ \dot{v} &= F_y - ru + pw + gcos\theta sin\phi \\ \dot{w} &= F_z - pv + qu + gcos\theta cos\phi \end{aligned} \tag{3.7}$$

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 sideway. 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 3.3. 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

substantially 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 3.7 are than rewritten in find form as follows:

$$\begin{aligned} \dot{u} &= X_u u - qw + rv - gsin\theta \\ \dot{v} &= Y_v v - ru + pw + gcos\theta sin\phi \\ \dot{w} &= Z_w w + T - pv + qu + gcos\theta cos\phi \end{aligned} \tag{3.8}$$

where X_u, Y_v and Z_w are the aerodynamic derivatives (accounting for the aerodynamic drag). The value 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:

$$T = -g - \Delta a_Z \tag{3.9}$$

where Δa_Z is given by the fourth transfer function in 3.3.

Comparing the equations in 3.8 with other works in the literature as for example in [16] it can be noticed that the rotor flapping terms are missing (the rotor flapping represents the possibility of the helicopter rotor disk to tilt relatively to the helicopter fuselage). On the other hand the flapping dynamics is contained in the transfer functions in 3.3. It was not possible to model the rotor flapping explicitly in the equations 3.8 because it is not observable from the black-box model identification approach used. The fact that the flapping terms are not included in equations 3.8 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. The model anyway is good enough for position and velocity control loop design. In [20] the mathematical formulation for rotor flapping can be found.

3.5 Simulation results

The validation of the RMAX helicopter model can be found in [4]. 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 Fig. 3.5 and Fig. 3.6 the simulation (dashed line) and flight-test (continuous line) are overlapped on the same graph. The upper-left plot of Fig. 3.5 represents the helicopter path 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 Fig. 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 Fig. 3.7 and Fig. 3.8 the same path segment is tested but the helicopter accelerates until 10 m/s 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 due to the fact that the aerodynamics of the rotor is not modeled. Taking into account such effects is necessary for flight conditions far from hovering. It is interesting to notice that the simulation can predict quite accurately the saturation of the roll input (Fig. 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.

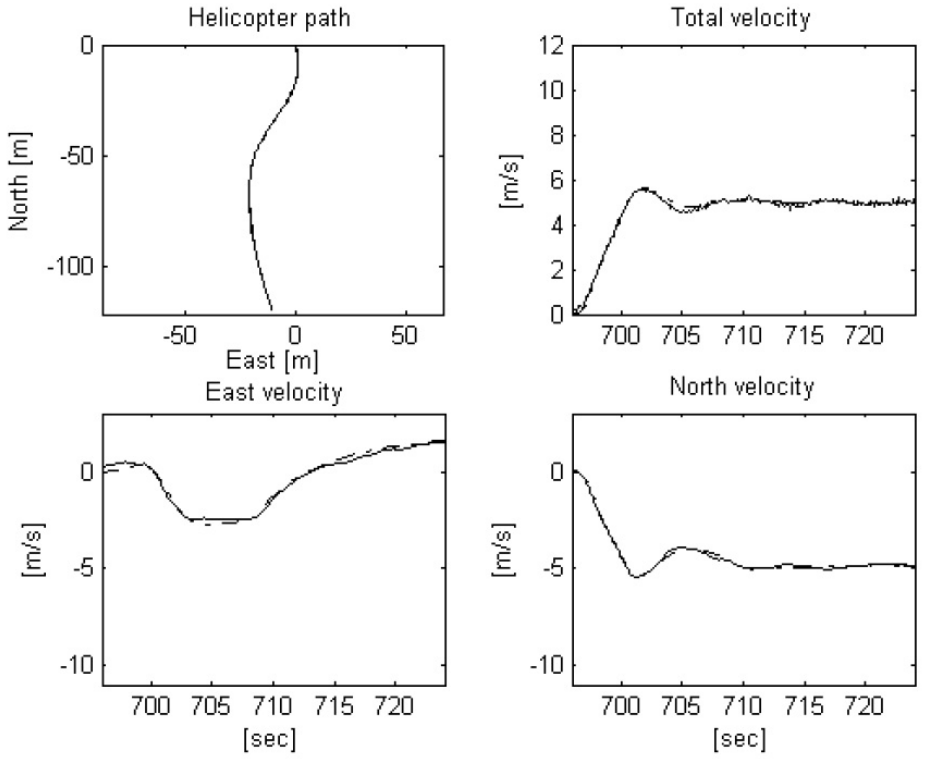


Figure 3.5: **Results 1st flight.** Comparison between flight-test position and velocity data (solid) with simulation data (dash-dot). The helicopter accelerates to 5 m/s target velocity.

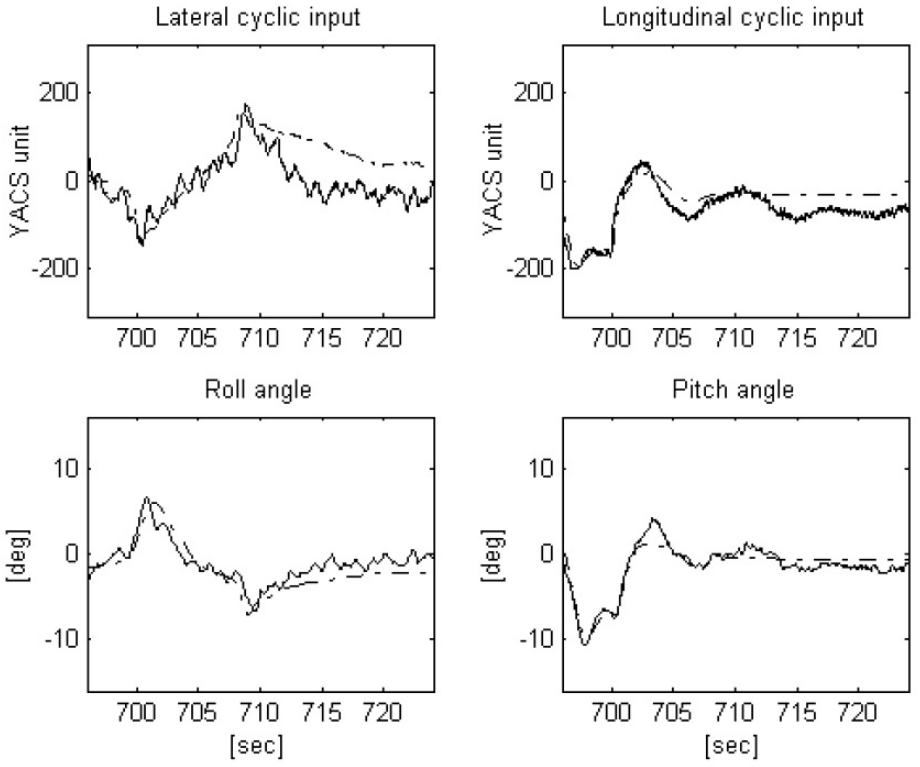


Figure 3.6: **Results 1st flight.** Comparison between flight-test helicopter inputs and attitude data (solid) with simulation data (dash-dot). The helicopter accelerates to 5 m/s target velocity.

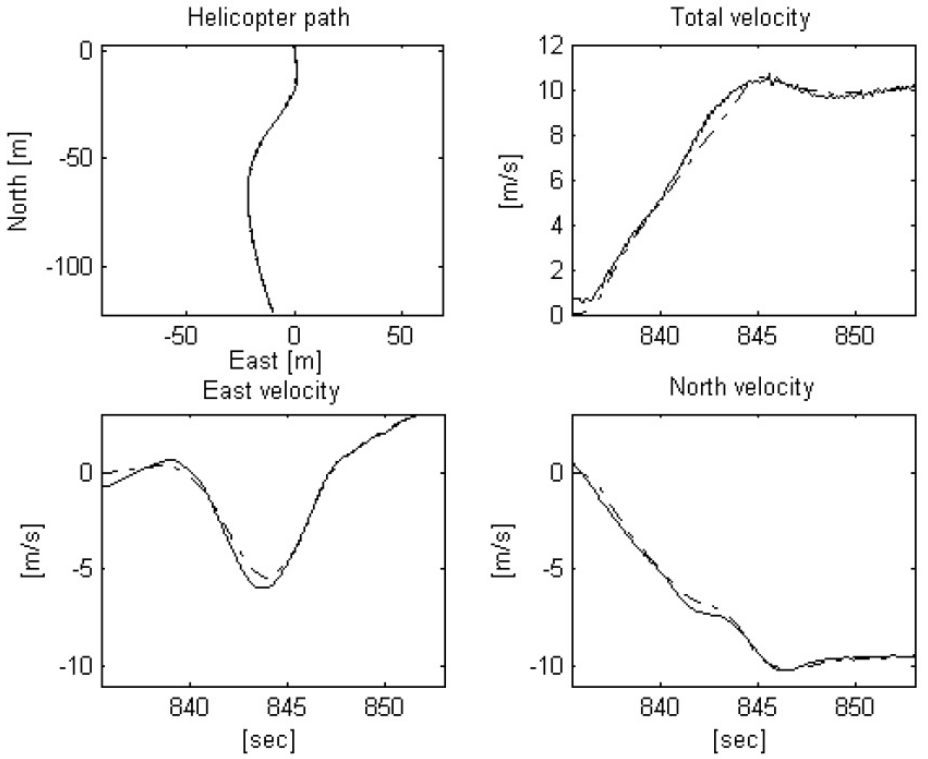


Figure 3.7: **Results 2nd flight.** Comparison between flight-test position and velocity data (solid) with simulation data (dash-dot). The helicopter accelerates to 10 m/s target velocity.

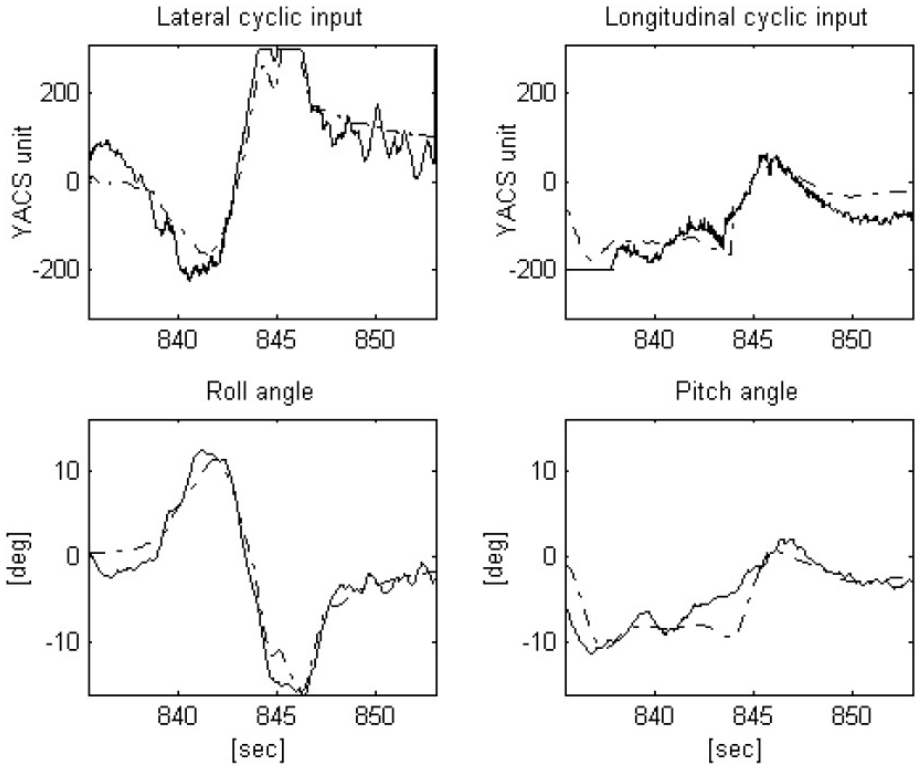


Figure 3.8: **Results 2nd flight.** Comparison between flight-test helicopter inputs and attitude data (solid) with simulation data (dash-dot). The helicopter accelerates to 10 m/s target velocity.

3.6 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 on the field. In addition, the fact that the helicopter servos are in the simulation loop provides a rapid verification that the right 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 PFCM described in this chapter and presented in Paper I, has been designed to navigate an autonomous helicopter in an area cluttered with obstacles, such as an urban environment. In this thesis, the path planning problem is not addressed although it is in [18]. It is assumed that a path planning functionality 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 geometry of the path that the helicopter has to follow. A geometric path segment, represented by a set of parameters, is then input to the PFCM.

Before starting with the description of the PFCM, some basic terminology will be provided.

The *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.

Several methods have been proposed to solve the problem of generation and execution of a state-space trajectory for an autonomous helicopter [6, 9]. 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 [18] 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 [24, 22]. 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 [24].

The approach used in this thesis does not use the Serret-Frenet formulas but a different guidance algorithm similar to the one described in [5] 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 (Fig. 4.1) is basically a point lying on the path where the helicopter ideally should be.

The guidance algorithm developed uses information from the model in 3.8 described in section 3.4.2 in order to improve the path tracking

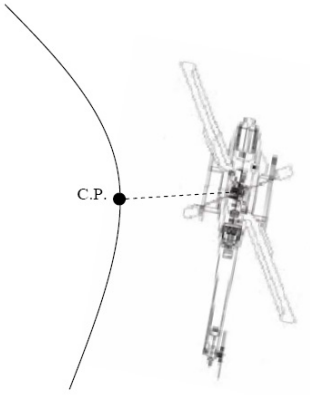


Figure 4.1: Control point on the reference path.

error while maintaining a reasonable flight speed. The experimental results presented show the validity of the control approach.

Fig. 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.

4.2 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 helicopter model described in chapter 3.

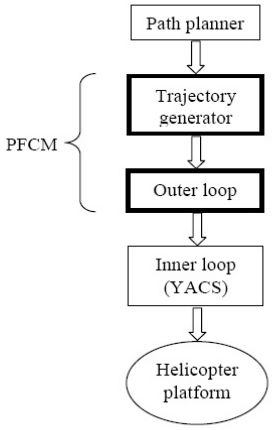


Figure 4.2: The PFCM includes two modules: trajectory generator and outer loop.

4.2.1 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:

$$\boldsymbol{P}^n(s) = \boldsymbol{A}s^3 + \boldsymbol{B}s^2 + \boldsymbol{C}s + \boldsymbol{D} \tag{4.1}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are 3D vectors calculated from the boundary conditions of the segment with s as the segment parameter.

Fig. 4.3 depicts a path segment with the relative boundary conditions. The segment is defined by the starting point coordinates $\boldsymbol{P}(0)$, the end point coordinates $\boldsymbol{P}(1)$ and two vectors which represent the direction of the segment tangent at the starting point $\boldsymbol{P'}(0)$ and at the end point $\boldsymbol{P'}(1)$ so that the 3D path segment is defined by 12 parameters. The path planner [18] calculates the 12 parameters ensuring the continuity of the path and of the first order derivative at the segment joints.

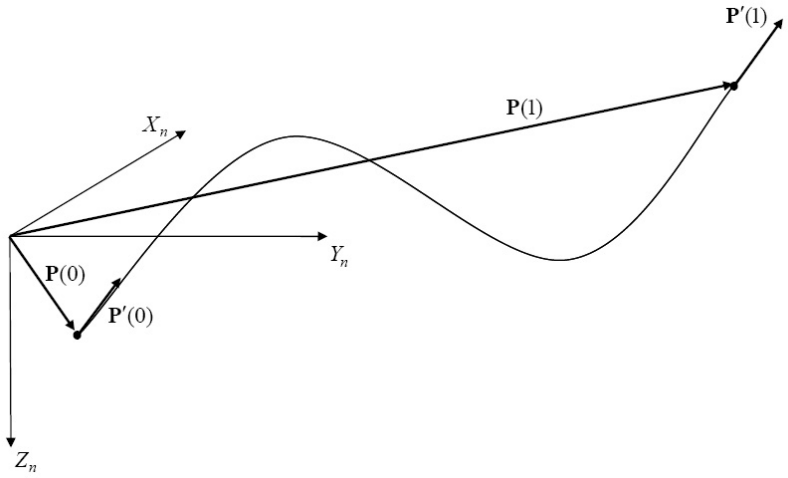


Figure 4.3: Boundary conditions for a path segment.

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 $\boldsymbol{P'}(0)$ and $\boldsymbol{P'}(1)$.

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 [21]. 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 [18].

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 $\boldsymbol{P}(0)$ to $s = 1$ which corresponds to the end point $\boldsymbol{P}(1)$ of the same segment. When the helicopter enters the next segment the parameter is reset to zero.

$$\boldsymbol{P}^n(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}(0) \\ \boldsymbol{P}(1) \\ \boldsymbol{P'}(0) \\ \boldsymbol{P'}(1) \end{bmatrix} \tag{4.2}$$

For control purposes the tangent and the curvature need to be calculated. The path tangent $\boldsymbol{T}^n$ is:

$$\boldsymbol{T}^n(s) = \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}(0) \\ \boldsymbol{P}(1) \\ \boldsymbol{P'}(0) \\ \boldsymbol{P'}(1) \end{bmatrix} \tag{4.3}$$

The path curvature $\boldsymbol{K}^n$ is:

$$\boldsymbol{Q}^n(s) = \begin{bmatrix} 6s & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}(0) \\ \boldsymbol{P}(1) \\ \boldsymbol{P'}(0) \\ \boldsymbol{P'}(1) \end{bmatrix}$$

$$\boldsymbol{K}^n(s) = \frac{\boldsymbol{T}^n(s) \times \boldsymbol{Q}^n(s) \times \boldsymbol{T}^n(s)}{|\boldsymbol{T}^n(s)|^4} \tag{4.4}$$

In the guidance law the curvature radius which is $\boldsymbol{R}^n = 1/\boldsymbol{K}^n$ will be used. Since $\boldsymbol{T}^n$ and $\boldsymbol{R}^n$ 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 3.3.

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.

4.2.2 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 [18]).

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 [18].

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:

$$\boldsymbol{E} \bullet \boldsymbol{T} = 0 \tag{4.5}$$

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

$$e_f = \boldsymbol{E} \bullet \boldsymbol{T}/|\boldsymbol{T}| \tag{4.6}$$

that is the magnitude of the error vector projected on the tangent $\boldsymbol{T}$. The vector $\boldsymbol{E}$ is calculated as follows:

$$\boldsymbol{E} = \boldsymbol{p}_{cp,n-1} - \boldsymbol{p}_{heli} \tag{4.7}$$

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

$$d\boldsymbol{p} = \boldsymbol{p}' \cdot ds \tag{4.8}$$

Equation 4.8 is applied in the discretized form:

$$s_n = s_{n-1} + \frac{e_f}{|\frac{d\boldsymbol{p}_{cp}}{ds}|_{n-1}} \tag{4.9}$$

where s_n is the new value of the parameter. Equations 4.6 and 4.9 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.

4.2.3 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 3.8 will be used to derive the guidance law which enables the helicopter to follow a 3D path.

The *sin* and *cos* can be linearized around $\theta = 0$ and $\phi = 0$ since in our flight condition, the pitch and roll angles are between the interval $\sim \pm 20$ deg. This means that we can approximate the *sin* of the angle to the angle itself (in radians) and the *cos* of the angle to 1. By doing this from the first and second equation of 3.6 it is possible to calculate the body angular rate p and q:

$$\begin{aligned} q &= \dot{\theta} + r\phi \\ p &= \dot{\phi} - r\theta \end{aligned} \tag{4.10}$$

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 substituting 4.10 it is possible to rewrite the system in 3.8 in the following form:

$$\begin{aligned} \dot{u} &= X_u u - (\dot{\theta} + r\phi)w + rv - g\theta \\ \dot{v} &= Y_v v - ru + (\dot{\phi} - r\theta)w + g\phi \\ \dot{w} &= Z_w w + T - (\dot{\phi} - r\theta)v + (\dot{\theta} + r\phi)u + g \end{aligned} \tag{4.11}$$

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:

$$\begin{aligned} r &= \frac{u}{R_y^b} \\ \dot{\theta} &= \frac{u}{R_z^b} \\ v &= \dot{v} = 0 \\ w &= \dot{w} = 0 \end{aligned} \tag{4.12}$$

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 4.11 can finally be rewritten as:

$$\begin{aligned} r &= \frac{u}{R_y^b} \\ \theta &= \frac{X_u u - \dot{u}}{g} \\ \phi &= \frac{u^2}{g R_y^b} \\ T &= -\frac{u^2}{R_z^b} - \frac{u^4}{g(R_y^b)^2} - g \end{aligned} \tag{4.13}$$

In the right hand side of 4.13 we have the four inputs that can be given as a reference signal to an inner control loop (like 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 4.13 which does not involve the dynamic constraints is represented by a stationary turn on the horizontal plane with constant

radius (picture (a) of Fig. 4.4). The solution for this flight condition is given by 4.13 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 4.13:

$$\begin{aligned} u_1 &= |R_y^b r_{max}| \\ u_2 &= |\frac{g\theta_{max}}{X_u}| \\ u_3 &= \sqrt{|\phi_{max} g R_y^b|} \\ u_4 &= (|-g(T_{max}+g)(R_y^b)^2|)^{\frac{1}{4}} \end{aligned} \tag{4.14}$$

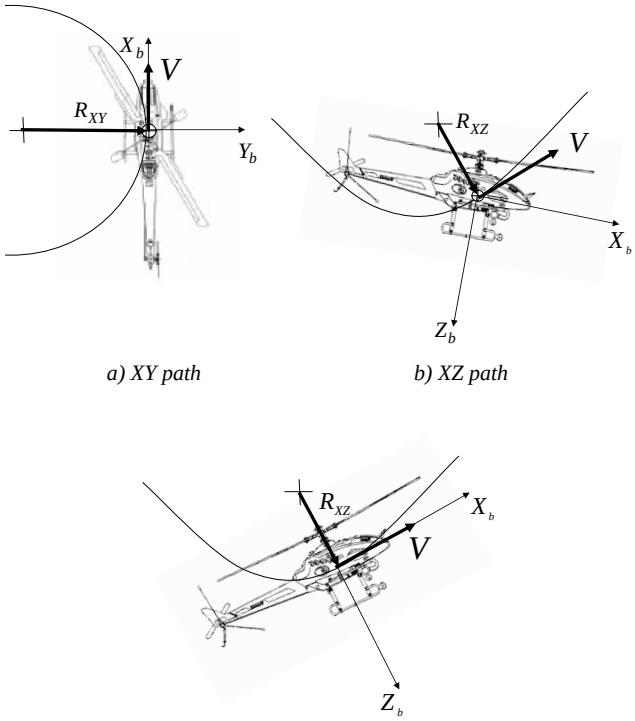
The minimum of these four velocities can be taken as the maximum speed for the path:

$$u_{max} = min(u_1, u_2, u_3, u_4) \tag{4.15}$$

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

Let's now examine a maneuver in the vertical plane ($R_y^b = \infty$, $R_z^b = constant$). From the second equation in 4.12 and the second equation in 4.13 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 constraint $\dot{\theta} = u/R_z^b$ 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 Fig. 4.4 (c). The helicopter instead will follow the path as it is shown in Fig. 4.4 (b).



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.

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 $\delta\boldsymbol{p}$.

   The position error vector is the difference between the control point position $\boldsymbol{p}^n_{cp}$ and the helicopter position given by the INS/GPS system $\boldsymbol{p}^n_{heli}$. 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 $\boldsymbol{C}^b_n$:

   $$\delta\boldsymbol{p}^b \;=\; \boldsymbol{C}^b_n(\boldsymbol{p}^n_{cp} - \boldsymbol{p}^n_{heli})$$

   As explained in section 4.2.2, 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 $\delta\boldsymbol{p}^b$ can be calculated together with the local path tangent T and curvature K.

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

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

$$\boldsymbol{\tau}^n = \frac{\boldsymbol{T}^n}{|\boldsymbol{T}|} \tag{4.16}$$

In order to be used in the control equations in 4.24, the velocity error vector must be expressed in the body frame in the same way as the position error vector:

$$\delta\boldsymbol{v}^b = \boldsymbol{C}_n^b(v_{targ} \cdot \boldsymbol{\tau}^n - \boldsymbol{v}_{heli}^n) \tag{4.17}$$

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 Fig. 4.5.

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

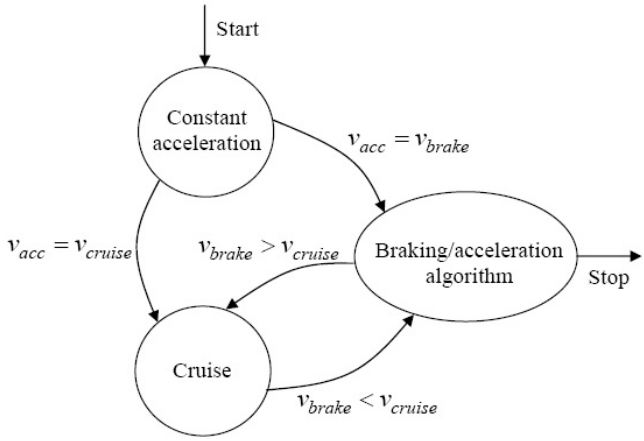


Figure 4.5: State-machine representing the calculation of the velocity v_{targ1}.

braking condition is activated when the following condition becomes true:

$$v_{brake} < v_{cruise} \tag{4.18}$$

with

$$v_{brake} = \sqrt{|(2 \cdot l_{end} \cdot Acc + v_{end}^2)|} \tag{4.19}$$

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 4.18 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 Paper I.

The calculation of v_{targ2} takes into account the kinematic constraints described by the equations in 4.14. For safety reasons the flight envelope has been limited to: $r_{max} = 40$ rad/sec maximum yaw rate, ϕ_{max} = 15 deg maximum roll angle, θ_{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 Nz_{max} = 1.1 means increasing the helicopter weight 10 percent. The maximum takeoff weight of the RMAX is 94kg and the RMAX weight used in this experimental test is around 80kg, so a load factor of 1.1 ensures enough safety. The second equation in 4.14 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 4.15. Finally we can calculate v_{targ2} as:

$$v_{targ2} = \frac{u_{max}}{\tau_x^b} \tag{4.20}$$

where τ_x^b is the projection of the vector in 4.16 on the helicopter X_b body axis.

Fig. 4.6 shows the three velocities u_1, u_3 and u_4, depending on the path curvature radius, calculated from equations in 4.14 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 80kg, in this case Nz_{max} could become the limiting factor.

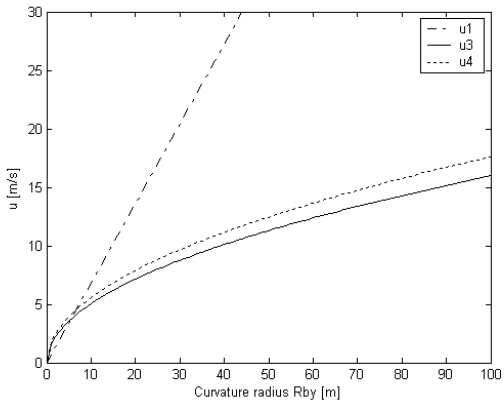


Figure 4.6: Velocity constraint due to the maximum yaw rate, roll angle and load factor.

It is important to mention that the velocity u_{max} is calculated using equation 4.14 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 Fig. 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 [20].

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

$$V_i = 60\sqrt{\frac{DL}{2\rho}} \tag{4.21}$$

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 4.21 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 m. From the diagram, the vertical speed when the light turbulence first occurs is

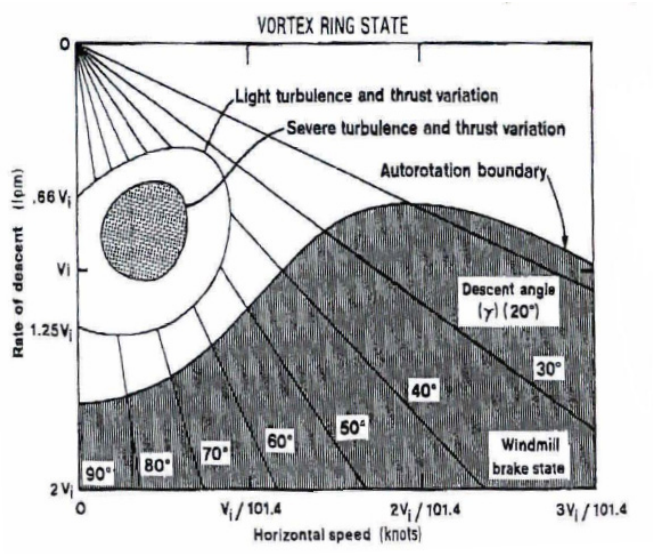


Figure 4.7: Flow states in descending forward flight.

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 Fig. 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.

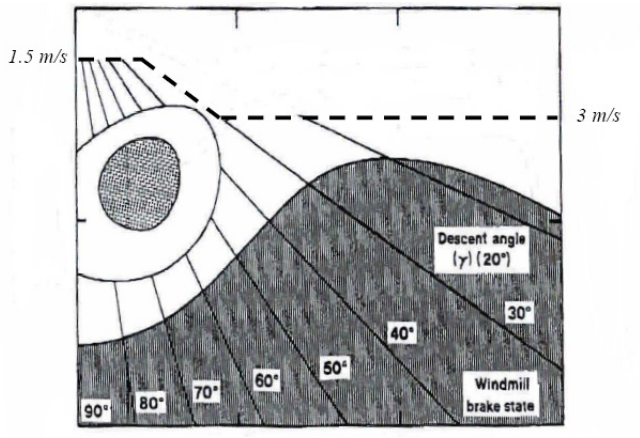


Figure 4.8: Maximum descent velocity used in the PFCM for the RMAX helicopter.

The calculation of v_{targ3} is then:

$$\begin{aligned} \gamma &= atan(\frac{\tau_z^n}{\tau_x^n}) \\ w_{MAXdescent} &= 1.5 \quad 90^\circ > \gamma \geq 30^\circ \\ w_{MAXdescent} &= 3 \quad 30^\circ > \gamma > 0^\circ \\ v_{targ3} &= \frac{w_{MAXdescent}}{\tau_z^n} \end{aligned} \tag{4.22}$$

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 4.13 where the component of the curvature radius R_y^b is calculated as follows:

$$\begin{aligned} \boldsymbol{K}^b &= \boldsymbol{C}_b^n \boldsymbol{K}^n \\ R_y^b &= \frac{1}{K_y^b} \end{aligned} \tag{4.23}$$

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

4.3 Outer loop control equations

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

$$\begin{aligned} YAW_{yacs} &= r_{ff} + K_1^y \delta\psi \\ PITCH_{yacs} &= K_1^p \delta p_x + K_2^p \delta v_x + K_3^p \frac{d}{dt}\delta v_x + K_4^p \int \delta v_x dt \\ ROLL_{yacs} &= \phi_{ff} + K_1^r \delta p_y + K_2^r \delta v_y \\ THR_{yacs} &= K_1^t \delta p_z + K_2^t \delta v_z + K_3^t \int \delta v_z dt \end{aligned} \tag{4.24}$$

where the K's are the control gains, r_{ff} and ϕ_{ff} are the feed-forward control terms resulting from the model in 4.13. 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.

4.4 Experimental results

This section presents experimental results of the PFCM implemented on the RMAX helicopter. In Fig. 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 Fig. 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 decreasing 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. Results of several trials of the same path flown with different wind conditions are shown in Paper I.

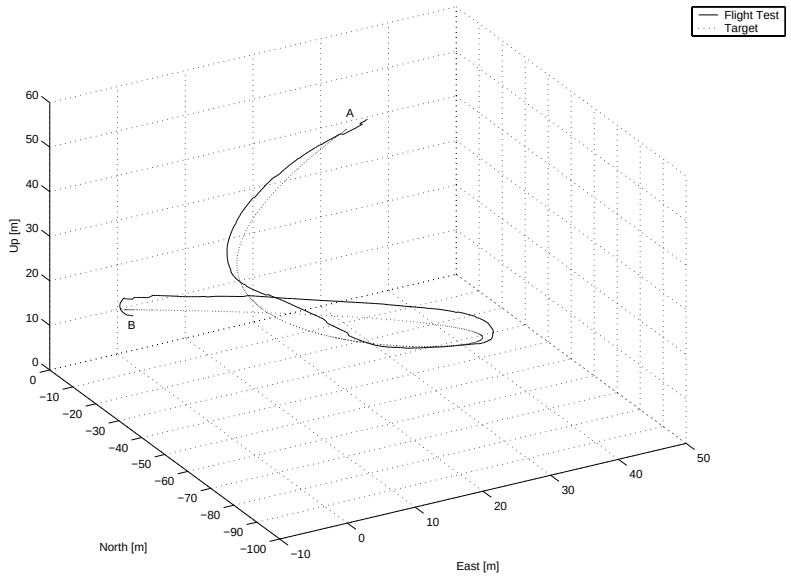


Figure 4.9: Target and actual 3D helicopter path.

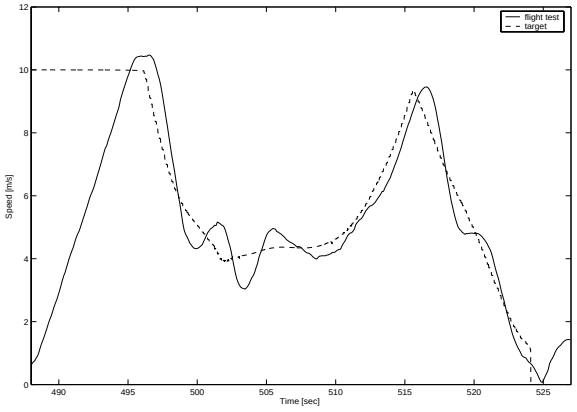


Figure 4.10: Target and actual helicopter speed.

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 Fig. 4.11 b). From the diagram one can compare the difference between the previous control scheme, Fig. 4.11 a), and the new one Fig. 4.11 b).

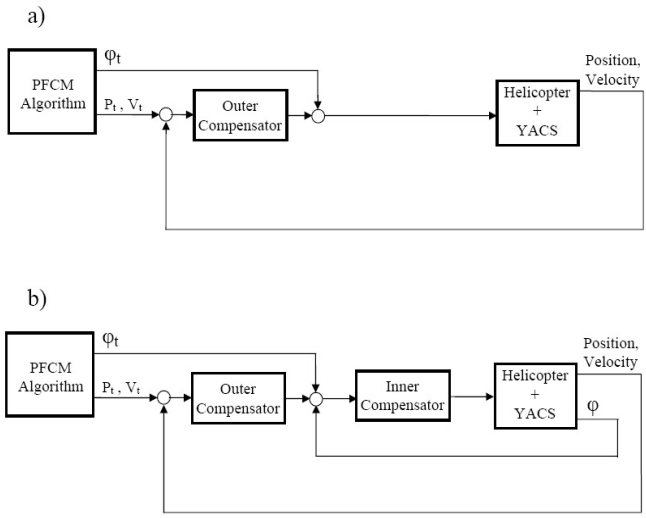


Figure 4.11: a)Previous lateral control. b)Modified lateral control loop using a lead compensator.

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

$$C(s) = K(\alpha\frac{1+s}{1+\alpha s}) + K_I\frac{1}{s} \tag{4.25}$$

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 Fig. 4.12. The target velocity in both cases was set to 10 m/s. Fig. 4.13 depicts the target velocity and the actual helicopter velocity relative to the path on the right side of Fig. 4.12. The diagram on the right side in Fig. 4.12 depicts the path flown with the basic PFCM controller (Fig. 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.

The diagram on the left in Fig. 4.12 depicts a test of the same path flown with the modified roll control loop (Fig. 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.

4.5 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 the PFCM in order to handle communication failures with the path planner (this can happen since the path planner is implemented on a separate computer). More details on this topic can be found in Paper I. Moreover, since the path planner was developed before the PFCM, a number of constraints

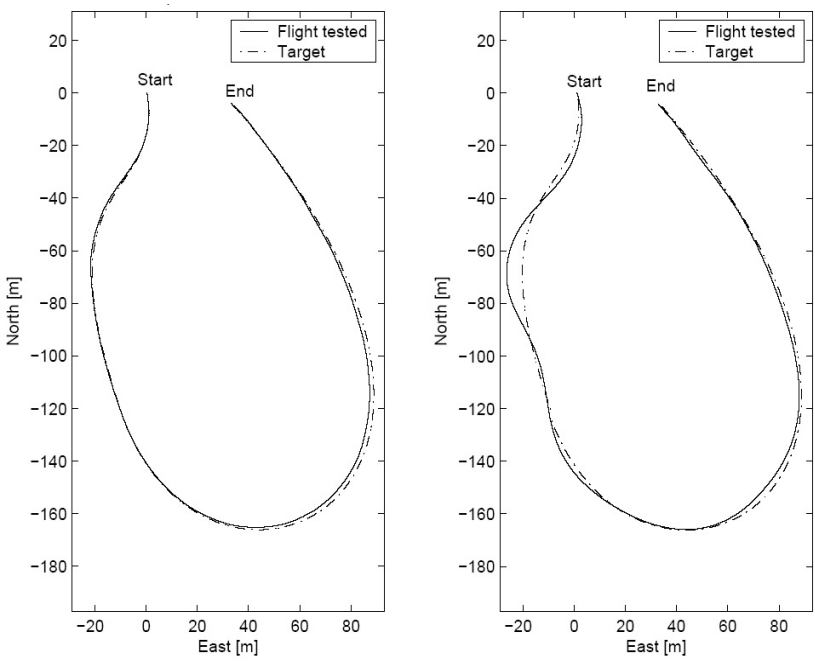


Figure 4.12: Comparison of path tracking performances using two different roll control strategy. On the right side is depicted the flight-test of the modified roll control loop with the lead compensator added. On the left the same test is done using the old roll control configuration. The flight-tests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

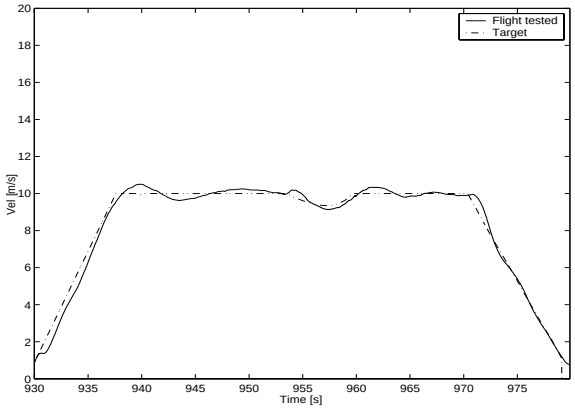


Figure 4.13: Target velocity and actual helicopter velocity.

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.

Chapter 5

Sensor fusion for vision based landing

This chapter describes the sensor fusion approach applied to a vision based landing capability described in Paper III.

The vision based autonomous landing mode developed during the WITAS Project has been tested on the 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 and it is placed on the ground during the landing maneuver. 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 and the pattern. This position is then used in the sensor fusion filter described in this chapter in order to provide reliable helicopter state for the autonomous landing. 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 loosing the pattern from the camera view due to accidental, abrupt helicopter movements.

Vision based autonomous landing is a complex problem and it requires

competences in several disciplines such as image processing, sensor fusion and control. Paper III describes the approach and solution to the complete problem. This chapter focuses on the sensor fusion problem involved in the vision based autonomous landing mode. Details of the image processing and control strategy are not described here. The reader interested in the details of these problems should read Paper III in the appendix of this thesis.

The motivations for the development of a vision based landing mode are of two categories: scientific and technical. The scientific motivation is that a helicopter which does not rely on external sources of information (like GPS) contributes to the scientific goal of a self-sufficient autonomous system. The technical motivation is that GPS technology is generally not robust while operating close to obstacles. For example, in an urban environment the GPS signal can be obscured by buildings or corrupted by multi path reflections or nearby radio frequency transmitters. The landing approach proposed in Paper III is completely independent of a GPS, so it can be used for landing the helicopter in proximity of obstacles found in urban environments.

In order to stabilize and control a UAV helicopter, an accurate and reliable state estimation is required. The standard strategy to solve this problem is to use several sensors with different characteristics, such as inertial sensors and GPS, and fuse them together using a Kalman filter. The integration between inertial sensors and GPS is a common practice and an extensive literature on this topic is available. Several approaches to this problem can be found in [13, 25, 23].

The method used here to fuse vision data with inertial sensors is similar to that used for GPS and inertial sensor integration with a number of differences in the implementation. The great experience gained in many successful experimental landings with our RMAX platform provides strong confirmation that the same sensor integration technique used for GPS and inertial sensors can be used when the GPS is replaced with a suitable image processing system. The vision based landing problem for an unmanned helicopter has been addressed by other research groups, some related work on this problem can be found in Paper III.

As already mentioned, the landing problem is solved by using a single camera mounted on a pan-tilt unit and an inertial measurement unit (IMU)

composed of three accelerometers and three gyros. An artificial pattern is placed on the ground and it is used by the image processing system to calculate the pose of the helicopter. A picture of the pattern is shown in Fig. 5.1.

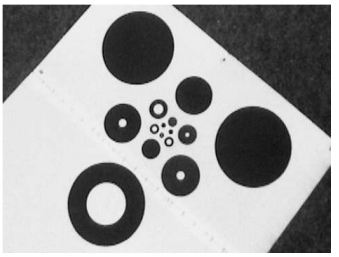


Figure 5.1: Pattern used for the vision based autonomous landing.

Vision and inertial sensors are combined together because of their complementarity. Vision provides drift-free position data, while inertial sensors provide position, velocity and attitude information at higher frequency but affected by drift. In fact position, velocity and attitude angles derived from inertial data alone experience unbounded error growth in time.

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 aircrafts or military submarines are equipped with such sensors. Small UAVs like 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 which have different characteristics. Fig. 5.2 shows a classification of inertial sensor performance. The data is taken from [23].

On the other hand, a vision based landing approach which relies only on a vision system data suffers of several problems. The vision system is sensitive to illumination conditions such as sun reflection or shadows which

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 5.2: Inertial sensor performance classification. The data are taken from [23].

can partially cover the pattern. In these situations the vision system is blind and does not deliver any position data. A clever landing control is to choose the best approach in order to avoid these situations using the knowledge of the sun position as explained in Paper III, but this is not always compatible with the wind direction which is also a factor when determining the landing approach. Also the pattern view can be lost before touching down when the helicopter is very close to the pattern. In fact, 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 filter can still deliver useful information to continue the landing maneuver. This capability will be shown in the experimental results.

By fusing inertial and vision data together using the Kalman filter technique, it is possible to obtain a reliable 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 angles information than the one given by vision only as will be shown by the experimental results.

5.1 Filter architecture

The filter developed is composed of two main functions: the INS mechanization function and the Kalman filter function. The INS mechanization function performs the time integration of the inertial sensors while the Kalman filter function estimates the errors of the INS mechanization. The errors estimated by the Kalman filter are then used to correct the final solution and are fed back into the mechanization process. The feedback architecture is shown in Paper III.

5.1.1 Filter initialization

The filter turns on automatically as soon as the first valid data from the vision system is available, then the first step is the calculation of the heading of the helicopter relative to the pattern. The heading is given directly by the vision system but a median filter over two seconds period is applied on the raw vision heading. The results from the median filter are taken as initial heading for the filter initialization. The assumption is that during the two seconds after the filter has turned on, the helicopter does not make any large yaw maneuver. Otherwise the median value might differ too much from the initial heading. The raw vision heading is not used directly in order to avoid the risk of initializing the filter with an outlier. The initialization of the heading is done carefully because the convergence of the filter to the correct heading is very slow when the helicopter is near a hovering condition, which is true in the case of a landing maneuver.

Once the heading is available, the filter can be initialized. The raw relative position coming from the vision system is taken as initial position. The vertical and horizontal velocities are initialized to zero since the landing approach starts from a hovering condition. Even if the initial position and velocity have a relatively small error, the filter converges very fast to the right value since the covariance of the vision measurement is quite low. The initial pitch and roll angles are taken directly from the Yamaha Attitude

Sensor (YAS). The YAS sensor has been introduced in section 2.2. These angles are only used to initialize the filter and are not used again during the landing. In this way the filter initialization does not rely on GPS at all and so the vision based positioning system is completely independent from it.

After the filter has been initialized the system waits ten seconds to make sure that all the filter parameters have converged before the landing procedure can begin. If the vision capability is lost during this time, the initialization procedure starts again from scratch. During the landing procedure, consistency checks are made in order to detect and reject measurement outliers from the vision system. In Paper III, details of the monitoring scheme implemented are given.

5.1.2 INS mechanization

The INS mechanization function implements and solves the navigation differential equations given in 5.1.

$$\begin{aligned} \dot{\boldsymbol{r}}^n &= \boldsymbol{v}^n \\ \dot{\boldsymbol{v}}^n &= \boldsymbol{C}_b^n \boldsymbol{f}^b - (2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n) \times \boldsymbol{v}^n + \boldsymbol{g}^n \\ \dot{\boldsymbol{C}}_b^n &= \boldsymbol{C}_b^n (\boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{in}^b) \end{aligned} \tag{5.1}$$

where $\boldsymbol{r}^n$ is the position , $\boldsymbol{v}^n$ the velocity and $\boldsymbol{C}_b^n$ the direction cosine matrix of the attitude angles calculated relative to a local navigation frame. In addition $\boldsymbol{f}^b$ and $\boldsymbol{\Omega}_{ib}^b$ are the accelerometer and gyro outputs, $\boldsymbol{g}^n$ is the gravity vector and $\boldsymbol{\omega}_{ie}^n$ the Earth rotation rate.

The calculation of the attitude angles is made using a quaternion representation because the linearity of the differential equations allows for an efficient implementation [8]. Details of the implementation of the INS mechanization can be found in [23].

5.1.3 Kalman filter

The Kalman filter implementation utilizes a 12-state linear INS error model of which nine navigation error states (latitude, longitude and altitude error;

north, east and down velocity error; roll, pitch and heading error), and three accelerometer biases modeled as first order Markov processes. The Kalman filter algorithm estimates the errors of the INS mechanization by fusing the estimate of these errors provided by an internal linear model and the measurements from the vision system. The theory behind the Kalman filter can be found in [12].

The linear model used for the INS errors is represented by the system of differential equations in 5.2:

$$\begin{aligned} \delta\dot{\boldsymbol{r}} &= -\boldsymbol{\omega}_{en}\times\delta\boldsymbol{r}+\delta\boldsymbol{v} \\ \delta\dot{\boldsymbol{v}} &= -(\boldsymbol{\omega}_{ie}+\boldsymbol{\omega}_{in})\times\delta\boldsymbol{v}-\boldsymbol{\psi}\times f+\delta\boldsymbol{a} \\ \dot{\boldsymbol{\psi}} &= -\boldsymbol{\omega}_{in}\times\boldsymbol{\psi} \\ \delta\dot{\boldsymbol{a}} &= -\beta\delta\boldsymbol{a} \end{aligned} \tag{5.2}$$

where $\delta\boldsymbol{r}$ is the position error, $\delta\boldsymbol{v}$ is the velocity error and $\delta\boldsymbol{\psi}$ is the attitude error. $\delta\boldsymbol{a}$ represents the three accelerometer biases. In addition $\boldsymbol{\omega}_{en}$ is the rotation rate vector of the navigation reference system relative to the Earth reference system, $\boldsymbol{\omega}_{in}$ is the rotation rate vector of the navigation reference system relative to the inertial reference system and $\boldsymbol{\omega}_{ie}$ is the rotation rate vector of the Earth reference system relative to the inertial reference system. The update of the filter is performed when a new measurement from the image processing system is available. The raw measurement delivered by the image processing is the 3D helicopter position relative to the pattern. The image processing measurement covariance used in the filter is scheduled with the distance to the pattern. The relation between the uncertainty of the vision measurement and the distance to the pattern is found in Paper III.

The Kalman filter is implemented in discretized form and the recursive implementation of the discretized Kalman filter equations can be found easily in the literature. A detailed implementation of a nine state Kalman filter can be found in [23].

5.2 Experimental results

The results presented in this section show the benefits resulting from the sensor fusion technique used in the vision based landing system.

The 12-state Kalman filter has been implemented in the C-language in the software architecture and runs at 50Hz. The measurement update from the vision system is 20Hz. When a new measurement from the vision system is available, the filter performs an update step. The accelerometers and gyros data are sampled at 50Hz although their output is at higher frequency. In Fig. 5.3 the available sensor characteristics are listed.

Sensor	Output Rate	Accuracy	Range	Resolution	Bias
Accelerometers	66 Hz	-	+- 1 G	1 mG	13mg
Gyros	200 Hz	-	+- 256 deg/s	0.1 deg/s	< 0.1deg/s
Vision	20 Hz	Paper III	20 m	384x288 pixels	-

Figure 5.3: Available characteristics of the sensors involved in the sensor fusion techniques used in the vision based landing system.

The sensors used to validate the filter results are a GPS system (centimeter accuracy) and the Yamaha Attitude System (YAS) for the attitude angles (around 2 deg accuracy).

The plots from Fig. 5.4 to Fig. 5.9 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 5.4 shows the comparison between the filter and the raw vision measurements.

Figures 5.5, 5.6 and 5.7 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 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 is shown in Paper III, 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

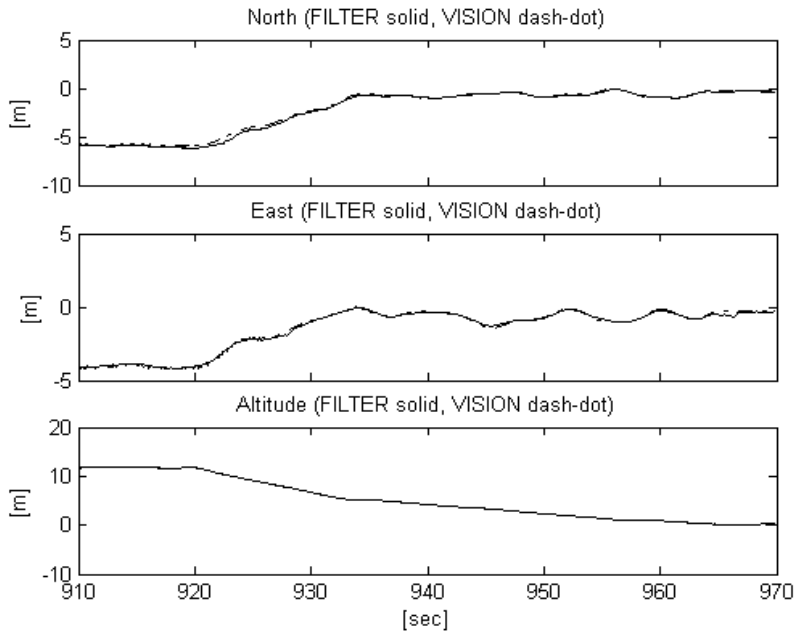


Figure 5.4: Comparison between vision and filtered position data during autonomous landing.

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.

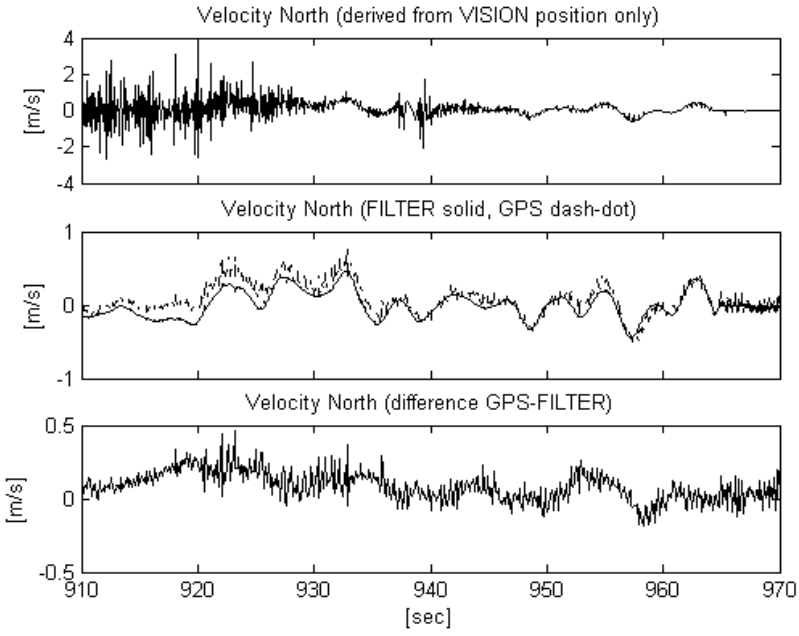


Figure 5.5: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the North component.

Figures 5.8 and 5.9, show a comparison between the attitude angles provided by the vision system alone (see Paper III for details as to how the vision system calculates the attitude angles), the attitude angles calculated

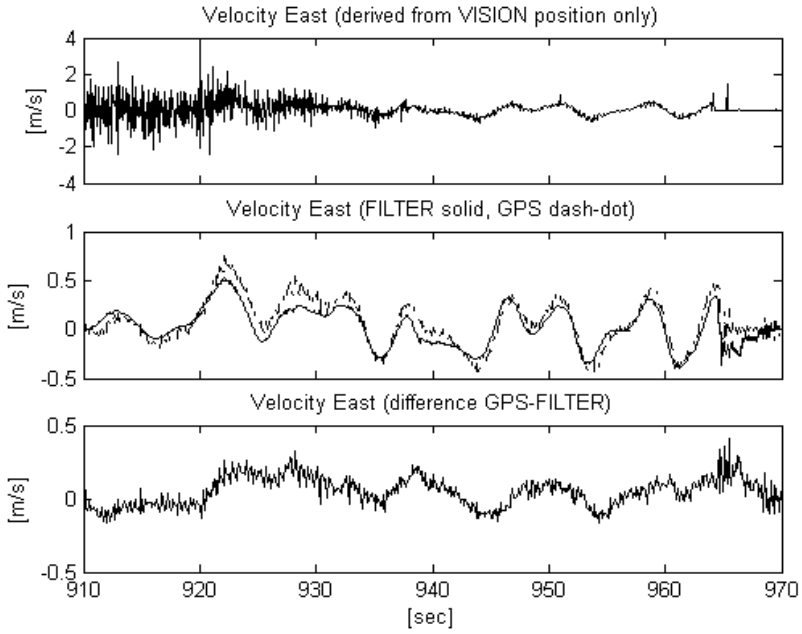


Figure 5.6: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the East component.

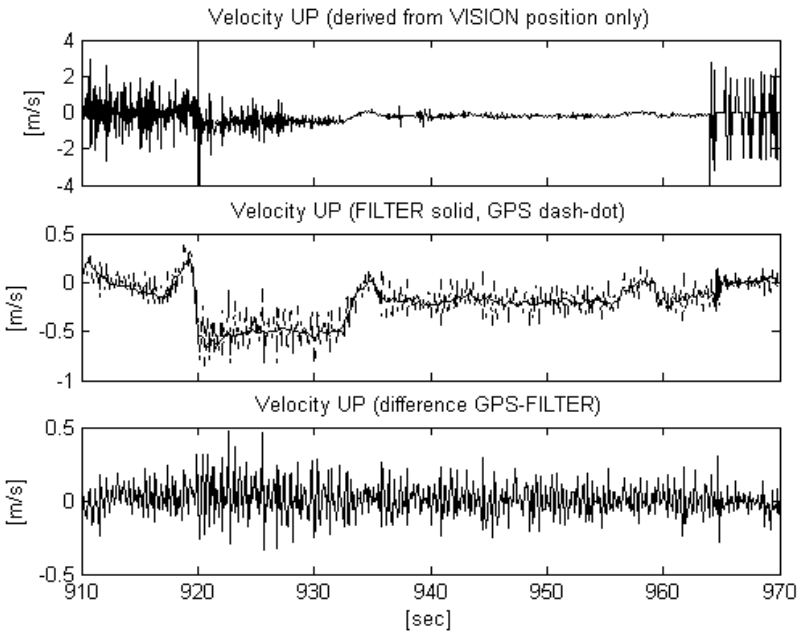


Figure 5.7: Comparison between velocity derived from raw vision position, sensor fusion and GPS for the vertical component.

by the Kalman filter described in this chapter and the attitude given by the YAS (attitude sensor built-in the RMAX helicopter). The attitude calculated by the vision system alone suffers from bias errors and noise 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. In fact the camera is mounted on a suspended platform mounted on springs in order to damp helicopter vibrations and this produces alignment error. The angles given by the filter do not suffer from this problem and are in good agreement with the YAS measurements.

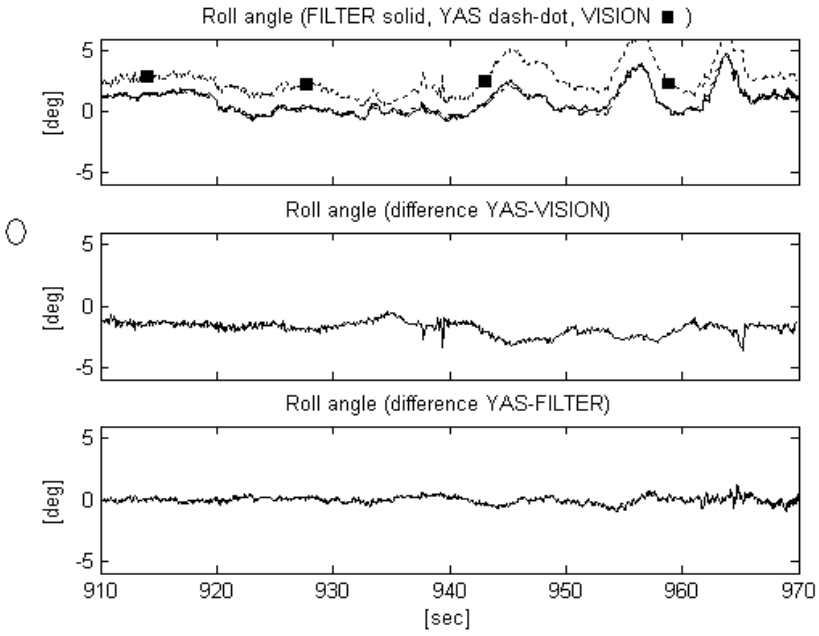


Figure 5.8: Comparison between roll angle calculated by the vision system, sensor fusion and YAS.

Fig. 5.10 shows an altitude plot from a different landing test. One can observe that when the helicopter was about 0.6 meters above the ground the

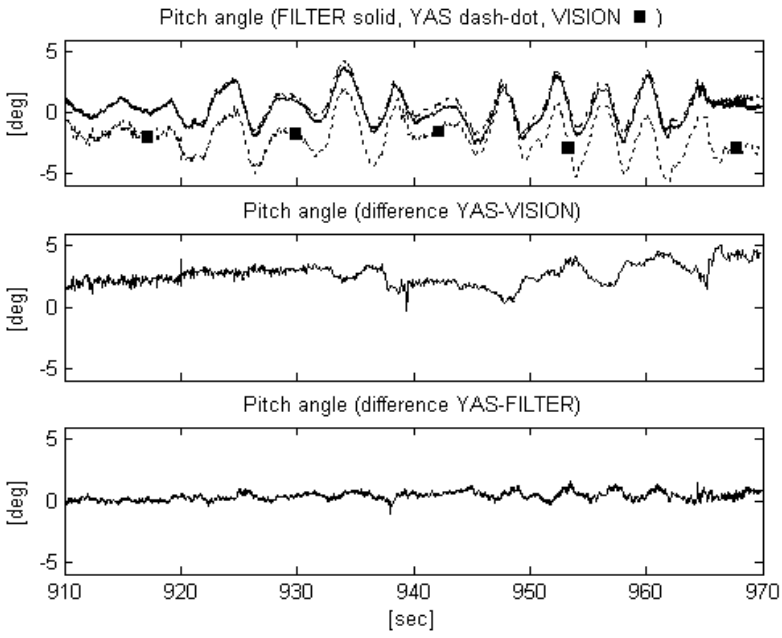


Figure 5.9: Comparison between pitch angle calculated by the vision system, sensor fusion and YAS.

vision system stopped delivering valid data because the pattern disappeared accidentally out of the camera's field of view. The filter continued to deliver position information using its dead-reckoning capability until the landing was terminated safely. This shows how sensor fusion techniques, enhance the overall robustness of the landing procedure.

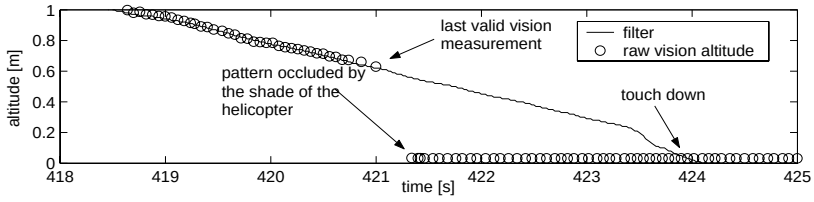


Figure 5.10: Altitude plot of an autonomous landing completed with the vision lost before touch down.

5.3 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 have 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 sensor with 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.

Bibliography

[1] MARVIN: TU Berlin. http://pdv.cs.tu-berlin.de/MARVIN/.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] Bernard Etkin and Lloid Duff Reid. *Dynamic of Flight: Stability and Control*. John Wiley and Sons, Inc., 1995.

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

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

[10] MIT/Draper Autonomous Helicopter Project. http://web.mit.edu/whall/www/heli/.

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

[12] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering, 82 (Series D): 35-45*, 1960.

[13] Jay Hyoun Kwon. Airborne Vector Gravimetry Using GPS/INS. *Master Thesis, The Ohio State University*, 2000.

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

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

[16] 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.

[17] Doherty P., Granlund G., Kuchcinski K., Sandewall E., Nordberg K., Skarman E., and Wiklund J. The WITAS unmanned aerial vehicle project. In *Proc. of the European Conf. on Artificial Intelligence*, pages 747–755, 2000.

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

[19] AVATAR: USC Autonomous Flying Vehicle Project. http://www-robotics.usc.edu/∼avatar.

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

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

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

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

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

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

[26] Berkeley Aerorobot Team. http://robotics.eecs.berkeley.edu/bear/.

[27] Georgia Tech UAV. http://www.ae.gatech.edu/labs/controls/uavrf/.

[28] Hummingbird: Stanford University. http://sun-valley.stanford.edu/users/heli/.

[29] CMU Autonomous Helicopter Project. www.cs.cmu.edu/afs/cs/project/chopper/www.

[30] 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.

Appendix A

A.1 Paper I

DYNAMIC 3D PATH FOLLOWING FOR AN AUTONOMOUS HELICOPTER

Gianpaolo Conte, [1] **Simone Duranti,** [1]
Torsten Merz [1]

Department of Computer and Information Science,
Linköping University, SE-58183 Linköping, Sweden

Abstract: A hybrid control system for dynamic path following for an autonomous helicopter is described. The hierarchically structured system combines continuous control law execution with event-driven state machines. Trajectories are defined by a sequence of 3D path segments and velocity profiles, where each path segment is described as a parametric curve. The method can be used in combination with a path planner for flying collision-free in a known environment. Experimental flight test results are shown.

Keywords: aerospace engineering, architectures, autonomous vehicles, finite state machines, helicopter control, splines, trajectories

1. INTRODUCTION

This work is part of the WITAS Unmanned Aerial Vehicle (UAV) Project (Doherty, 2004), a long-term basic research project with the goal of developing information technology systems for UAVs and core functionalities necessary for the execution of complex missions. The main objective is the development of an integrated hardware/software UAV for fully autonomous missions in an urban environment. A research prototype has been developed using a Yamaha RMAX helicopter as a flying platform. A number of interesting missions have been successfully demonstrated in a small uninhabited urban area in the south of Sweden called Revinge, which is used as an emergency services training area.

In order to navigate in an area cluttered by obstacles, such as an urban environment, path planning, path following (PF) and path switching mechanisms are needed. Several methods have been proposed to solve this type of navigation problem (Egerstedt *et al.*, 1999; Frazzoli, 2001).

Our major achievements in this paper are the development and flight-testing of an algorithm to follow a 3D path with a given velocity profile and a switching mechanism to enable the integration with a path planner in the deliberative part of the UAV architecture (Pettersson and Doherty, 2004). The method developed for PF is weakly model dependent and computationally efficient. The strategy used to follow a desired path is a velocity control tangent to the path and a position control orthogonal to it: the helicopter has to fly close to the geometric path with a specified forward speed. This approach is better known as dynamic PF. In the trajectory tracking problem the system is designed to follow a trajectory in the state-space domain where the state is parameterized in time: the path is not prioritized. In PF methods the path is always prioritized and this is a requirement for robots that for example have to follow roads and avoid collisions with buildings. A theoretical approach to dynamical PF is given in (Sarkar *et al.*, 1994). The path we want to follow

[1] Supported by the Wallenberg Foundation, Sweden

is a three-dimensional parameterized space curve. The motion of the reference point on the curve is governed by a differential equation containing error feedback. Similar methods have also been investigated in (Egerstedt *et al.*, 2001).

In (Harbick *et al.*, 2001) a technique for following planar spline trajectories using a behavior-based control architecture is implemented and tested in flight. The method developed in this paper differs from (Harbick *et al.*, 2001) in that it allows dynamic modification of the trajectory during execution and provides a mechanism that coordinates and monitors the processes to achieve proper control. Furthermore, our implementation allows 3D path tracking and information about the curvature of the path is fed forward in the control loops for enhanced tracking accuracy during manouvred flight at higher speeds.

2. SYSTEM OVERVIEW

The WITAS UAV system consists of a slightly modified Yamaha RMAX helicopter and the WITAS on-board system (Fig. 1). In this paper we focus on system components which are relevant for dynamic 3D path following. Our aerial robot has many more skills. A description of the full hard- and software system can be found in (Doherty *et al.*, 2004; Merz, 2004).

Fig. 1. The WITAS helicopter

The helicopter has a total length of 3.6 m (incl. main rotor), a max. take-off weight of 95 kg, and is powered by a 21 hp two-stroke engine. Yamaha equipped the remote-controlled RMAX with an attitude sensor (YAS) and an attitude control system (YACS). For the experiments described here the following components of the WITAS on-board system were used: an integrated INS/GPS with DGPS correction, a barometric altitude sensor, a PC104 embedded computer (700 MHz Pentium PIII), and a wireless Ethernet bridge.

The PC104 computer reads all sensors, runs the control software, and sends commands to the YACS. Sensor measurements and control outputs are logged in this computer and sent simultaneously to a ground station for on-line analysis. Different control modes and task procedures can be selected by a ground operator during flight.

Paths are decomposed into path segments, which are requested by the dynamic path following controller during execution. This method is chosen, as it allows to model almost any space curve and makes path modification easy. If a segment is not available in time, the system switches into a safety mode.

The structure of the hybrid control system for dynamic path following is shown in Fig. 2. Each block represents a *functional unit*. All functional units can be executed concurrently and asynchronously. At the highest level a *task procedure* provides control with path segment data. A task procedure is a computational mechanism that achieves a certain behavior of the WITAS UAV system (Doherty *et al.*, 2004). It is coupled to a state machine which coordinates data transfer, reports errors to the task procedure, and switches control modes. It uses statements derived from sensor measurements as conditions for state transitions. A set-point generator computes a number of set-points from path segment data and sensor measurements and passes it to an outer loop control. The inner loop is the Yamaha Attitude Control System (YACS) that stabilizes the attitude angles, the yaw and the vertical dynamics.

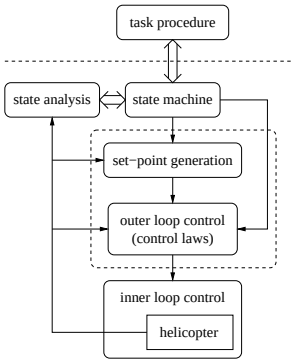


Fig. 2. Structure of the hybrid control system

3. TASK PROCEDURE AND STATE MACHINE

The interaction between task procedures and low level control is handled by an event-driven state machine (hybrid control). In the system considered here, a hierarchical concurrent state machine is implemented (HCSM). It is represented as a set of state transition diagrams similar to Harel's statecharts (Harel, 1987).

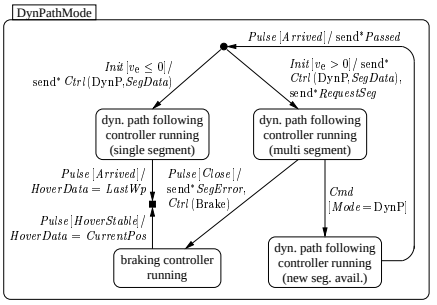


Fig. 3. State machine for dynamic path following

In the following, the state machine for dynamic path following is explained. The state transition diagram [2] is shown in Fig. 3. For a path with one segment (end velocity $v_e \leq 0$), the segment parameters *SegData* are passed to the set-point generator and the dynamic path following controller is started. A segment is defined by start and end points, start and end directions, target velocity, and end velocity. In the case of several segments (end velocity $v_e > 0$), the state machine passes the segment parameters, starts the same controller and sends a *RequestSeg* event to the task procedure. The next segment has to be provided before the helicopter reaches a point from where it is impossible to stop at the end point of the current segment (*Close* becomes true). This is a safety mechanism which prevents the helicopter from leaving the current path in case no new segment is available. In this case, or if the helicopter is not able to slow down to the desired velocity at the end point, a *SegError* event is sent and a braking controller is started which will brake the helicopter with maximum deceleration. When the helicopter passes the end point of a path segment (*Arrived* becomes true) a *Passed* event is sent to the task procedure. The state machine exits, when the helicopter hovers.

Fig. 4 shows an example of a state machine for a path with two segments [3]. The upper part models a task procedure (user state machine) and the lower part a flight mode switching mechanism. Both machines run concurrently. When the autonomous mode is engaged (*AutoSwitch* becomes true) the hovering controller is started. As soon as the helicopter hovers stably, the first segment is flown. The hovering controller is started again when the helicopter arrives at the final waypoint.

[2] *Pulse* is an event sent periodically, *Init* triggers a transition from an entry state (circular node) when condition holds, *Exit* is sent to the superstate when entering an exit state (square node).

[3] Rectangular boxes within state nodes denote nested state machines. Superstate transitions are executed prior to substate transitions.

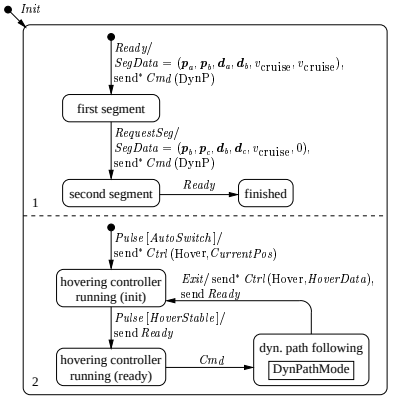


Fig. 4. Example of a state machine for a path with two segments

In the real system, the state machine for mode switching handles more flight modes and is separated from the user state machine (Merz, 2004).

4. SET-POINT GENERATION

The PF algorithm provides the set-points for the outer loop control. The inputs are provided by the event handler and the position sensor (INS/DGPS).

The analytical description of the 3D path is a cubic spline that has second-order continuity (C^2) at the joints, this is a requirement which avoids discontinuity in the helicopter's acceleration. A global reference frame is associated with each segment where the X-axis points north, the Y-axis points east and the Z-axis points down. The analytical form of the curve is:

$$\boldsymbol{P} = \boldsymbol{A}s^3 + \boldsymbol{B}s^2 + \boldsymbol{C}s + \boldsymbol{D} \tag{1}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are 3D vectors defined by the boundary conditions and s is the linear coordinate of the curve.

For each value of s the path generator provides the path parameters: position, tangent and curvature. The curvature is used to compute the centripetal acceleration needed to follow the path (feed-forward term in the lateral control law), while the tangent $\boldsymbol{T}$ is used to align the helicopter body to the path. The curvature $\boldsymbol{K}$ is a 3D vector and is calculated in the global frame as follows:

$$\boldsymbol{K} = \boldsymbol{T} \times \boldsymbol{Q} \times \boldsymbol{T} / |\boldsymbol{T}|^4 \tag{2}$$

$$\boldsymbol{T} = 3\boldsymbol{A}s^2 + 2\boldsymbol{B}s + \boldsymbol{C} \tag{3}$$

$$\boldsymbol{Q} = 6\,\boldsymbol{A}s + 2\boldsymbol{B} \tag{4}$$

where $\mathbf{Q}$ is the second order derivative.

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:

$$\boldsymbol{E} \bullet \boldsymbol{T} = 0 \tag{5}$$

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

$$e_f = \boldsymbol{E} \bullet \boldsymbol{T}/|\boldsymbol{T}| \tag{6}$$

that is the magnitude of the error vector projected on the tangent $\boldsymbol{T}$. The control point is updated using the differential relation:

$$d\boldsymbol{P} = \boldsymbol{P}' \cdot ds \tag{7}$$

Equation 7 is applied in the discretized form:

$$s(n) = s(n-1) + \frac{e_f}{|\frac{d\boldsymbol{P}}{ds}(n-1)|} \tag{8}$$

where $s(n)$ is the new value of the parameter. Once the new value of s is known, all the path parameters can be calculated.

The PF algorithm receives as inputs the target velocity v_t and the final velocity v_e that the helicopter must have at the end of the segment. The path planner assigns the target velocity which is related to the mission specification only. This means that the path planner doesn't have to take into account any dynamic limitation of the helicopter itself. The control law tries to keep the target velocity, but when it is not compatible with the local curvature of the path and the helicopter performance limitations, the algorithm provides an automatic limit on velocity. Velocity limitations can be activated for two reasons: due to the turn bank or the yaw rate limit of the helicopter. In order to make a coordinated turn at constant altitude, the flight mechanics provides the relation between the velocity, the roll angle and the curvature radius of the turn. Mechanical limits exist on the maximum achievable swash plate angles, and furthermore the helicopter envelope has currently been opened up to $\phi_{\max} \pm 8$ deg for the roll angle (relative to the hovering bank angle that is about 4.5 deg), and $\omega_{\max} \pm 26$ deg/sec for the yaw rate. Under the described limitations, it is possible to calculate the maximum speed:

$$V_{max1} = \sqrt{Rg\phi_{max}} \tag{9}$$

$$V_{max2} = \omega_{max}^2 R \tag{10}$$

where R is the local curvature radius and g is the gravity acceleration. The target speed assigned to the path is compared with these two limits and the lower speed is taken as target.

The braking algorithm continually checks the distance between the helicopter and the end point of the path. If the required acceleration to reach the final target velocity exceeds a given value (currently set to 1 m/s^2), the current target velocity is limited in order to maintain a constant deceleration. In order to know the distance between the helicopter and the end of the path, an estimate of the final arc length of the curve has to be calculated. The arc length of the spline between the control point and the end point of the path is:

$$l_{end} = \int_S^{S_{end}} \sqrt{[x'(s)]^2 + [y'(s)]^2 + [z'(s)]^2} ds \tag{11}$$

If an analytical solution of the integral cannot be found, a numerical method is used (rectangular integration with for example 20 integration steps).

To gain computational time, the increments of the flown path l_n are subtracted from l_{tot} (total length of the path, i.e. l_{end} at first iteration) to get a good estimate of l_{end} at each control cycle:

$$l_n = \sqrt{\begin{matrix}[x_n - x_{n-1}]^2 + \\ [y_n - y_{n-1}]^2 + \\ [z_n - z_{n-1}]^2\end{matrix}} \tag{12}$$

A path segment is considered finished when l_{end} is small enough. It should be emphasized that l_{end} is the arc length between the control point on the curve and the end point of the path and not between the helicopter and the end of the path. This makes the system more robust with regard to position error of the helicopter on the path.

5. CONTROL LAWS

The outer loop control (velocity and position control) provides inputs for the YACS in order to follow the path with the desired velocity. The inner loop deals with the coupling dynamics of the helicopter, so that the outer loop can handle the four degrees of freedom as decoupled (i.e. yaw rate, vertical velocity, pitch and roll angles). The position and velocity error and centripetal acceleration vectors are computed in the global frame and then transformed into control inputs after rotation in the helicopter's body frame. The acceleration vector is used as feed forward input in the control law to improve the tracking in the presence of path curvature. As regards the acceleration vector, only the component in the horizontal plane orthogonal to the path is used. PD and PI compensators are used respectively for position and velocity control. The algorithm described in the previous section makes sure that the position error vector is orthogonal to the path and the velocity error vector is tangent to the path. Given that the two error vectors are orthogonal the velocity control doesn't interfere with the position control. The control equations for the four channels are the following:

$$\begin{aligned}\theta_C &= K_{px}\delta X + K_{dx}\delta\dot{X} + K_{pvx}\delta V_X + \\ &+ K_{ivx}\delta V_{Xsum} + K_{fx}A_X \\ \Delta\phi_C &= K_{py}\delta Y + K_{dy}\delta\dot{Y} + K_{pvy}\delta V_Y + \\ &+ K_{ivy}\delta V_{Ysum} + K_{fy}A_Y \\ V_{ZC} &= K_{pz}\delta Z + K_{dz}\delta\dot{Z} + K_{pvz}\delta V_Z + \\ &+ K_{ivz}\delta V_{Zsum} + K_{fz}A_Z \\ \omega_C &= K_{pw}\delta\psi \end{aligned} \quad (13)$$

where the subscripted K's are control gains, the δ's are control errors, the pedices *sum* indicate the integral terms and the A's the components of the centripetal acceleration vector. θ_C is the target pitch angle, $\Delta\phi_C$ is the desired roll angle relative to the hovering roll angle, ω_C is the target yaw rate and V_{ZC} is the target vertical velocity.

6. EXPERIMENTAL RESULTS

The PF mode has been tested first in simulation and then in flight. The flight dynamics mathematical model of the augmented RMAX has been developed within the WITAS project and implemented in C. Simulations are done using hardware in the loop.
Only results from the flights are reported in the following.

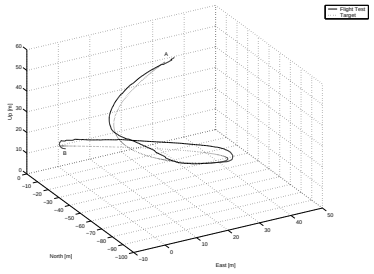


Fig. 5. Target and actual 3D helicopter path

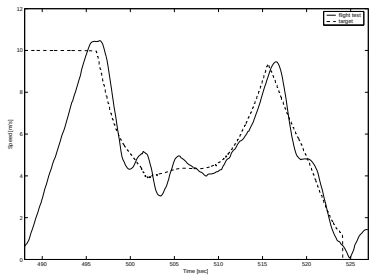


Fig. 6. Target and actual speed of the helicopter

Fig. 5 and 6 show a 3D segment and the velocity profile during one of the flight-tests. The helicopter hovers at point A at 40 meters altitude, starts the descending spiral, brakes and hovers at point B at 10 meters altitude. The maximum speed for the flight was set to 10 m/s, and the controller limited the target speed according to the local curvature and the braking algorithm. The maximum vertical speed component was around 3 m/s.

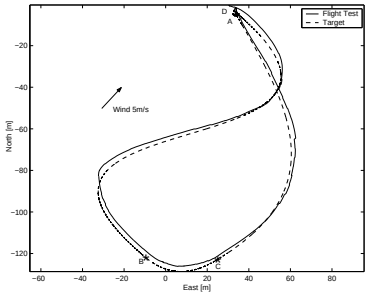


Fig. 7. Multisegment 2D path

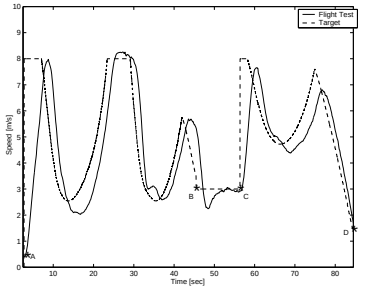


Fig. 8. Speed profile of a multisegment path

Fig. 7 and 8 show a trajectory consisting of 3 path segments at constant altitude. The mission starts with autonomous hovering in point A, then the helicopter flies the first path segment with maximum speed of 8 m/s; at point B the first segment is finished and a path switching leads the helicopter to the second segment with a maximum speed of 3 m/s; in point C the switch to the third path segment with maximum speed of 8 m/s takes place. Finally the helicopter brakes and hovers in point D where the mission ends. The wind was blowing constantly at 5 m/s. The tracking error depends on the angle between the path and the wind direction. In this case the maximum error is about 3 meters.

Table 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,

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 1. Experimental data

representative paths of typical flight manoeuvres 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. Fig. 5 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 column are 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). Because of the occurence of sudden jumps of the INS/GPS position signal, the maximum errors shown in the table are not always imputable to control errors; to evaluate the performance of the PF, the average error gives more reliable information.

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.

7. CONCLUSIONS AND FUTURE WORK

The results of the experimentation show a satisfactory tracking behavior. The position control error is well within the accuracy of the available position measurement. The algorithm has also been successfully tested in relatively severe weather conditions, with wind levels up to 15 m/s. In the presence of the strongest wind levels, the algorithm could be improved in order to reduce the lateral error; an integrative compensator could be added for this purpose but the tuning would not be straight forward in presence of high curvature. The PF mode is now implemented in the software architecture of the WITAS helicopter and is being used as a core functionality in complex mission tasks.

REFERENCES

Doherty *et al.* (2004). A distributed architecture for autonomous unmanned aerial vehicle experimentation. In: *Proc. of the 7th International Symposium on Distributed Autonomous Robotic Systems*. pp. 221–230.

Doherty, P. (2004). Advanced research with autonomous unmanned aerial vehicles. In: *Proc. of the 9th International Conference on the Principles of Knowledge Representation and Reasoning*. pp. 731–732.

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

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

Frazzoli, E. (2001). Robust Hybrid Control for Autonomous Vehicle Motion Planning. PhD thesis. Massachusetts Institute of Technology.

Harbick, K., J. Montgomery and G. Sukhatme (2001). Planar spline trajectory following for an autonomous helicopter. In: *Proc. of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. pp. 408–413.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming* **8**(3), 231–274.

Merz, T. (2004). Building a system for autonomous aerial robotics research. In: *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*.

Pettersson, P-O and P. Doherty (2004). Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle. In: *ICAPS-04 Workshop on Connecting Planning Theory with Practice*.

Sarkar, N., X. Yun and V. Kumar (1994). Control of mechanical systems with rolling constraints: Application to dynamic control of mobile robots. *International Journal of Robotics Research (MIT Press)* **13**(1), 55–69.

A.2 Paper II

From Motion Planning to Control - A Navigation Framework for an Autonomous Unmanned Aerial Vehicle

From Motion Planning to Control - A Navigation Framework for an Autonomous Unmanned Aerial Vehicle

Mariusz Wzorek, Gianpaolo Conte, Piotr Rudol
Torsten Merz, Simone Duranti, Patrick Doherty
Department of Computer and Information Science
Linköping University, SE-58183 Linköping, Sweden
{marwz,giaco,pioru,g-torme,simdu,patdo}@ida.liu.se

ABSTRACT

The use of Unmanned Aerial Vehicles (UAVs) which can operate autonomously in dynamic and complex operational environments is becoming increasingly more common. While the application domains in which they are currently used are still predominantly military in nature, in the future we can expect widespread usage in the civil and commercial sectors. In order to insert such vehicles into commercial airspace, it is inherently important that these vehicles can generate collision-free motion plans and also be able to modify such plans during their execution in order to deal with contingencies which arise during the course of operation. In this paper, we present a fully deployed autonomous unmanned aerial vehicle, based on a Yamaha RMAX helicopter, which is capable of navigation in urban environments. We describe a motion planning framework which integrates two sample-based motion planning techniques, Probabilistic Roadmaps and Rapidly Exploring Random Trees together with a path following controller that is used during path execution. Integrating deliberative services, such as planners, seamlessly with control components in autonomous architectures is currently one of the major open problems in robotics research. We show how the integration between the motion planning framework and the control kernel is done in our system.

Additionally, we incorporate a dynamic path reconfigurability scheme. It offers a surprisingly efficient method for dynamic replanning of a motion plan based on unforeseen contingencies which may arise during the execution of a plan. Those contingencies can be inserted via ground operator/UAV interaction to dynamically change UAV flight paths on the fly. The system has been verified through simulation and in actual flight. We present empirical results of the performance of the framework and the path following controller.

BIOGRAPHY

Patrick Doherty is a Professor at the Department of Computer and Information Science (IDA), Linköping University (LiU), Sweden. He is director of the Artificial Intelligence and Integrated Computer Systems Division at IDA and his research interests are in the area of knowledge representation, automated planning, autonomous systems, approximate reasoning and UAV technologies.
Mariusz Wzorek (MSc) is a graduate student at LiU. Research focus: automated planning techniques, autonomous unmanned systems and robotics.
Gianpaolo Conte (MSc) is a graduate student at LiU. Research focus: control and sensor fusion related problems.
Piotr Rudol (MSc) is a graduate student at LiU. Research focus: non-GPS navigation, mapping and obstacle avoidance for UAVs.
Torsten Merz (PhD) has been involved in the WITAS project at LiU for many years. Since 2006, he is working at the Autonomous Systems Lab at CSIRO (Australia) leading the development of an unmanned aerial vehicle for power line inspection.
Simone Duranti (MSc) is working at Saab Aerosystems and LiU. He has been involved in the WITAS project at LiU for many years.

21th Bristol UAV Systems Conference — April 2006

1 Introduction

Navigating in environments cluttered with obstacles in the vicinity of building structures requires path planning algorithms which deliver collision-free paths, accurate controllers able to execute such paths even in the presence of inhospitable weather conditions (e.g. wind gusts) and a reliable mechanism that coordinates the two.

This paper describes an approach to combining path planning techniques with a path execution mechanism (including a robust 3D path following control mode) in a distributed software architecture used in a fully deployed rotor-based Unmanned Aerial Vehicle (UAV). Details of many of the software components used in the distributed architecture are provided. An emphasis is placed on the components responsible for path execution. The approach allows for interaction of a path planning algorithm with a path following control mode and copes with their different timing characteristics and distributed communication. It also includes a safety mechanism which is necessary for operating UAVs in urban environments. For the experiments we present in this paper, we assume a predominantly static environment which is described by a 3D model. An onboard geographic information system (GIS) is used to supply information about building structures, vegetation, etc. Certain types of dynamic changes in the environment are handled by the use of no-fly zones or pop-up zones which can be added or removed on the fly during the course of a mission.

Our hardware/software framework incorporates software distribution technologies for a number of reasons. Firstly, existing commercial off-the-shelf (COTS) hardware suitable enough for airborne systems, does not yet have sufficient computational power and storage space to encompass all the necessary software components needed to achieve sophisticated mission scenarios autonomously. Additionally, in order to use third-party software without compromising flight safety, it is necessary to separate software components that can crash the operating system from software that is crucial for the UAV flight operation. Another reason for using a distributed solution is to take advantage of additional resources which may be found on the Internet.

One of the long term goals which has guided our research is the idea of *push-button* missions where the ground operator supplies mission tasks to a UAV at a very high-level of abstraction and the UAV system does most of the work ranging from planning to actual execution of the mission.

The Autonomous UAV Technologies Laboratory [1] at Linköping University, Sweden, has been developing fully autonomous rotor-based UAV systems in the mini- and micro-UAV class. Our current system design is the result of an evolutionary process based on many years of developing, testing and maintaining sophisticated UAV systems. In particular, we have used the Yamaha RMAX helicopter platform and developed a number of micro air vehicles from scratch.

Much effort has also gone into the development of useful ground control station interfaces which encourage the idea of push-button missions, letting the system itself plan and execute complex missions with as little effort as possible required from the ground operator other than stating mission goals at a high-level of abstraction and monitoring the execution of the ensuing mission. The mission scenarios we use are generic in nature and may be instantiated relative to different applications. For example, the functionality required for the monitoring/surveillance mission described below can be modified slightly and used in mission scenarios such as power line inspection.

An example of such a push-button mission that has been used as an application scenario in our research is a combined monitoring/surveillance and photogrammetry mission out in the field in an urban area with the goal of investigating facades of building structures and gathering both video sequences and photographs of building facades. For this experiment, we have used a Yamaha RMAX helicopter system as a platform. Let us assume the operational environment is in an urban area with a complex configuration of building and road structures. A number of these physical structures are of interest since one has previously observed suspicious behavior and suspects the possibility of terrorist activity. The goal of the mission is to investigate a number of these buildings and acquire video and photos from each of the building's facades. It is assumed the UAV has a 3D model of the area and a GIS with building and road structure information on-line.

The ground operator would simply mark building structures of interest on a map display and press a button to generate a complete multi-segment mission that flies to each building, moves to waypoints to view each facade, positions the camera accordingly and begins to relay video and/or photographs. The motion plans generated are also guaranteed to be collision-free from

[1] `www.ida.liu.se/~patdo/auttek/`

static obstacles. If the ground operator is satisfied with the generated mission, he or she simply clicks a confirm button and the mission begins. During the mission, the ground operator has the possibility of suspending the mission to take a closer look at interesting facades of buildings, perhaps taking a closer look into windows or openings and then continuing the mission. This mission has been successfully executed robustly and repeatedly from take-off to landing using the RMAX. The plan generation and execution mechanism described in this paper is an integral part of such a complex mission.

1.1 Related Work

Many universities (20; 22; 1; 19; 9; 21; 23) have been and continue to do research with autonomous helicopter systems. Most of the research has focussed on low-level control of such systems with less emphasis on high-autonomy as in our case. The research area itself is highly multidisciplinary and requires competences in many areas such as control system design, computer science, artificial intelligence, avionics and electronics. We briefly mention a number of interesting university research projects representative of the type of research being pursued.

The Autonomous Helicopter Project at Carnegie Mellon University has done a great deal of research on helicopter modeling (16) and helicopter control (2), developing and testing robust flight control for full-envelope flight.

The School of Aerospace Engineering at Georgia Tech has developed an adaptive control system using neural networks (10) and has demonstrated the ability for high speed flight and operation with aggressive flight maneuvers using the Yamaha RMAX helicopter.

The motion planning problem for helicopters has been investigated by (6). Here the problem of the operation of an autonomous vehicle in a dynamic environment has been an issue of research and a method to reduce the computational complexity of the problem has been proposed based on the quantization of the system dynamics leading to a control architecture based on a hybrid automaton. The proposed approach has been tested in simulation.

1.2 Paper Outline

The paper is structured as follows. In section 2, we describe the hardware architecture developed and used on a Yamaha RMAX helicopter and on-board hardware components. In section 3, we describe the distributed CORBA-based software architecture and a number of software modules used in an integrated manner with the robotic system. Much of this work was completed in the WITAS [2] UAV project. In section 4, we describe the planning techniques used in the UAV system. In section 5, the path following control mode is described in detail and in section 6 we descibe the path execution mechanism. Dynamic path replanning capability is described in section 7 and experimental results of two example missions are presented in section 8. In section 9, we conclude and discuss future work.

2 The Hardware Platform

Figure 1: The WITAS RMAX Helicopter in an urban environment

The WITAS UAV platform (4) is a slightly modified Yamaha RMAX helicopter (Fig. 1). It has a total length of 3.6 m (including main rotor) and is powered by a 21 hp two-stroke engine with a maximum takeoff weight of 95 kg. The helicopter has a built-in attitude sensor (YAS) and an attitude control system (YACS). The hardware platform developed during the WITAS UAV project is integrated with the Yamaha platform as shown in Fig. 2. It contains three PC104 embedded computers. The primary flight control (PFC) system runs on a PIII (700Mhz), and includes a wireless Ethernet bridge, a GPS receiver, and several additional sensors including a barometric altitude sensor. The PFC is connected to the YAS and YACS, an image processing computer and a computer for deliberative capabilities. The image processing (IPC)

[2]WITAS is an acronym for the Wallenberg Information Technology and Autonomous Systems Lab which hosted a long term UAV research project (1997-2004).

system runs on the second PC104 embedded computer (PIII 700MHz), and includes a color CCD camera mounted on a pan/tilt unit, a video transmitter and a recorder (miniDV). The deliberative/reactive (DRC) system runs on the third PC104 embedded computer (Pentium-M 1.4GHz) and executes all high-end autonomous functionality. Network communication between computers is physically realized with serial line RS232C and Ethernet. Ethernet is mainly used for CORBA applications (see below), remote login and file transfer, while serial lines are used for hard real-time networking.

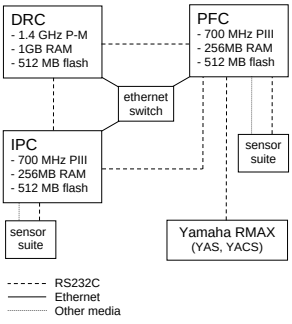


Figure 2: On-board hardware schematic

3 The Software Architecture

A hybrid deliberative/reactive software architecture has been developed for the UAV. Conceptually, it is a layered, hierarchical system with deliberative, reactive and control components, although the system can easily support both vertical and horizontal data and control flow. Fig. 3 presents the functional layer structure of the architecture and emphasizes its reactive-concentric nature. Fig. 4 depicts the navigation subsystem and main software components. With respect to timing characteristics, the architecture can be divided into two layers: (a) the hard real-time part, which mostly deals with hardware and control laws (also referred to as the Control Kernel) and (b) the non real-time part, which includes deliberative services of the system (also referred to as the High-level system) [3]. All three

[3]Note that distinction between the Control Kernel and the High-level system is done based mainly on the timing characterisitcs and it does not exclude, for example, placing some deliberative

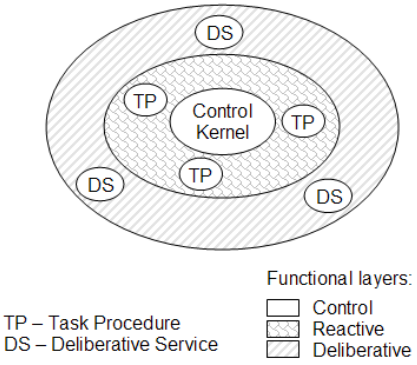


Figure 3: Funcional structure of the architecture

computers in our UAV platform (i.e. PFC, IPC and DRC) have both hard and soft real-time components but the processor time is assigned to them in different proportions. On one extreme, the PFC runs mostly hard real-time tasks with only minimum user space applications (e.g. SSH daemon for remote login). On the other extreme, the DRC uses the real-time part only for device drivers and real-time communication. The majority of processor time is spent on running the deliberative services. Among others, the most important ones from the perspective of this paper are the Path Planner, the Task Procedure Execution Module and the Helicopter Server which encapsulates the Control Kernel (CK) of the UAV system.

The CK is a distributed real-time runtime environment and is used for accessing the hardware, implementing continuous control laws, and control mode switching. Moreover, the CK coordinates the real-time communication between all three on-board computers as well as between CKs of other robotic systems. In our case, we perform multi-platform missions with two identical RMAX helicopter platforms developed in the WITAS UAV project. The CK is implemented using C language. This part of the system uses the Real-Time Application Interface (RTAI) (13) which provides industrial-grade real-time operating system functionality. RTAI is a hard real-time extension to a standard Linux kernel (Debian in our case) and has been developed at the Department of the Aerospace Engineering of Politecnico di Milano.

The real-time performance is achieved by inserting

services (e.g. prediction) in the Control Kernel.

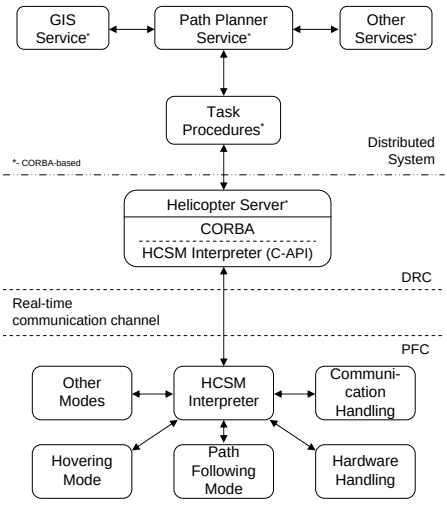


Figure 4: Navigation subsystem and main software components

a module into the Linux kernel space. Since the module takes full control over the processor it is necessary to suspend it in order to let the user space applications run. The standard Linux distribution is a task with lower priority, it runs preemptively and can be interrupted at any time. For that reason a locking mechanism is used when both user- and kernel-space processes communicate though shared memory. It is also important to mention that the CK is self-contained and only the part running on the PFC computer is necessary for maintaining flight capabilities. Such separation enhances safety of the operation of the UAV platform which is especially important in urban environments.

The Control Kernel has a hybrid flavor. Its components contain continuous control laws and mode switching is realized using event-driven hierarchical concurrent state machines (HCSMs). They can be represented as state transition diagrams similar to those of statecharts (8). In our system, tables describing transitions derived from such diagrams are passed to the system in the form of text files and are interpreted by a HCSM Interpreter at run-time in each of the on-board computers. Thanks to its compact and efficient implementation, the interpreter runs in the real-time part of the system as a task with high execution rate. It allows coordinating all *functional units* of the control system from the lowest level hardware components (e.g. device drivers) through control laws (e.g. hovering, path following) and communication to the interface used by the Helicopter Server.

The use of HCSMs also allows implementing complex behaviors consisting of other lower level ones. For instance, landing mode includes control laws steering the helicopter and coordinates camera system/image processing functionalities. When the landing behavior is activated, the CK takes care of searching for a pre-defined pattern with the camera system, feeding the Kalman filter with image processing results which fuses them with the helicopter's inertial measurements. The CK sends appropriate feedback when the landing procedure is finished or it has been aborted. For details see (15).

For achieving best performance, a single non-preemptive real-time task is used which follows a predefined static schedule to run all functional units. Similarly, the real-time communication physically realized using serial lines is statically scheduled with respect to packet sizes and rates of sending. For a detailed description see (14).

The high-level part of the system has reduced timing requirements and is responsible for coordinating the execution of reactive Task Procedures (TPs). This part of the system uses Common Object Request Broker Architecture (CORBA) as its distribution backbone. A TP is a high-level procedural execution component which provides a computational mechanism for achieving different robotic behaviors by using both deliberative and control components in a highly distributed and concurrent manner. The control and sensing components of the system are accessible for TPs through the Helicopter Server which in turn uses an interface provided by the Control Kernel. A TP can initiate one of the autonomous control flight modes available in the UAV (i.e. take off, vision-based landing, hovering, dynamic path following (see section 5) or reactive flight modes for interception and tracking). Additionally, TPs can control the payload of the UAV platform which currently consists of the video camera mounted on a pan-tilt unit. TPs receive data delivered by the PFC and IPC computers i.e. helicopter state and camera system state (including image processing results), respectively. The Helicopter Server on one side uses CORBA to be accessible by TPs or other components of the system, on the other side it communicates through shared memory with the HCSM based interface running in the real-time part of the DRC

software.

The software architecture described is used to achieve missions which require deliberative services such as path planners and control laws such as path following described in detail in sections 4 and 5, respectively. Details of the interaction between the TPs, path planners and the Control Kernel are presented in section 6.

4 The Path Planning Algorithms

In this section, we provide a brief overview of the sample-based path planning techniques used in our framework. More detailed descriptions of the two path planners can be found in (18; 17).

The problem of finding optimal paths between two configurations in a high-dimensional configuration space such as a helicopter is intractable in general. Sample-based approaches such as probabilistic roadmaps (PRM) or rapidly exploring random trees (RRT) often make the path planning problem solvable in practice by sacrificing completeness and optimality.

4.1 Probabilistic Roadmaps

The standard probabilistic roadmap (PRM) algorithm (11) works in two phases, one off-line and the other on-line. In the off-line phase a roadmap is generated using a 3D world model. Configurations are randomly generated and checked for collisions with the model. A local path planner is then used to connect collision-free configurations taking into account kinematic and dynamic constraints of the helicopter. Paths between two configurations are also checked for collisions. In the on-line or querying phase, initial and goal configurations are provided and an attempt is made to connect each configuration to the previously generated roadmap using the local path planner. A graph search algorithm such as A* is then used to find a path from the initial to the goal configuration in the augmented roadmap.

Fig. 5 provides a schema of the PRM path planner used in the WITAS UAV system. The planner uses an OBBTree-algorithm (7) for collision checking and an A* algorithm for graph search. Here one can optimize for shortest path, minimal fuel usage, etc. The following extensions have been made with respect to the standard version of the PRM algorithm in order to adapt the approach to our UAV platform.

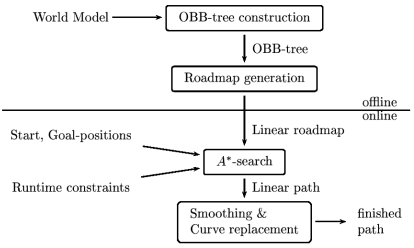


Figure 5: PRM path plan generation

- *Multi-level roadmap planning*

  The standard probabilistic roadmap algorithm is formulated for fully controllable systems only. This assumption is true for a helicopter flying at low speed with the capability to stop and hover at each waypoint. However, when the speed is increased the helicopter is no longer able to negotiate turns of a smaller radius, which imposes demands on the planner similar to non-holonomic constraints for car-like robots. In this case, linear paths are first used to connect configurations in the graph and at a later stage these are replaced with cubic curves when possible. These are required for smooth high speed flight. If it is not possible to replace a linear path segment with a cubic curve then the helicopter has to slow down and switch to hovering mode at the connecting waypoint before continuing. From our experience, this rarely happens.

- *Runtime constraint handling*

  Our motion planner has been extended to deal with different types of constraints at runtime not available during roadmap construction. Such constraints can be introduced at the time of a query for a path plan. Some examples of runtime constraints currently implemented include maximum and minimum altitude, adding forbidden regions (no-fly zones) and placing limits on the ascent-/descent-rate. Such constraints are dealt with during the A* search phase.

The mean planning time in the current implementation (for our operational environments) is below 1000 ms and the use of runtime constraints do not noticeably influence the mean.

4.2 Rapidly Exploring Random Trees

The use of rapidly exploring random trees (RRT) provides an efficient motion planning algorithm that constructs a roadmap online rather than offline. The algorithm (12) generates two trees rooted in the start and end configurations by exploring the configuration space randomly in both directions. While the trees are being generated, an attempt is made at specific intervals to connect them to create one roadmap. After the roadmap is created, the remaining steps in the algorithm are the same as with PRMs. In comparison with the PRM planner, the mean planning time with RRT is also below 1000 ms, but in this case, the success rate is much lower and the generated plans are not optimal which may sometimes cause anomalous detours (17).

Results of the path planning algorithms are used by the path following controller described in the following section.

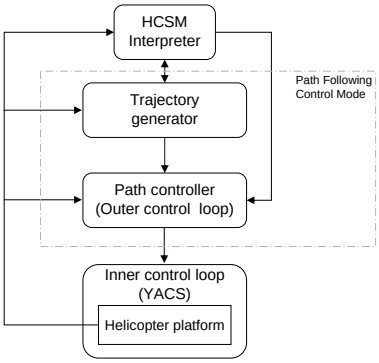


Figure 6: Interaction between the Path Following Control Mode and other components of the Control Kernel

5 Path Following Control Mode

In this section, we provide a detailed description of a path following control mode (Fig. 6, the bottom part of Fig. 4) which executes paths provided by the planner introduced in section 4. As described in section 3, the HCSM-based Control Kernel coordinates execution of different control modes available in the UAV system, including the path following control mode.

In the classical approach to the trajectory following problem, a trajectory is generated directly taking into account the dynamic constraints of the system. In our approach, however, we split the problem into two parts. First, the path planner generates a geometrical description of a path and the dynamic constraints are handled later by the path following mode by generating an appropriate velocity profile. In order to navigate safely in urban environments, the path following mode has been designed to minimize the tracking error during a path execution. Because paths generated by the planner are collision-free (relative to the static obstacles present in the model), staying closer to the geometric path assures safer navigation in the environment.

A path is composed of one or more segments (each described by a 3D cubic curve) which are passed sequentially to the control mode. The mode is implemented as a function which takes as input the segment geometry, the desired cruise and final velocities. It is called by the Control Kernel with a frequency of 50Hz and its output consists of four control signals (pitch, roll, yaw and the vertical channel). Additionally, the function returns a set of status flags which are used by the HCSM in control to generate appropriate events, i.e. path segment switching mechanism and safety braking procedure.

When the helicopter reaches the end point of the current segment, the controlling HCSM is informed and an $Arrived$ event is generated. The next call to the path following function is made using parameters describing the next segment to be flown. The process continues until all segments of the path are executed.

The safety braking procedure is activated in case the next segment is not provided by the High-level system (e.g. due to communication failure) in a specific time. This time point is calculated as the minimum distance necessary to stop the helicopter at the end of the current segment without overshooting.

The path tracking error is also available to the controlling HCSM and it can be used to take appropriate actions in case it becomes too large to guarantee safe operation. Such a situation can arise if the wind is too strong for the platform to keep it on the desired path.

The path following control mode is conceptually divided into two parts: (a) the *Trajectory Generator* (TG) which calculates the reference trajectory used by (b) the *Path Controller* (PC) to calculate the control signals. We consider each part in the next two subsections.

5.1 Trajectory Generator

A trajectory represents the evolution of a dynamic system in the state-space domain, where the state variables are parameterized in time. In our approach the generated reference trajectory (based on the planner output) depends not on the time but on the position of the helicopter relative to the path. In order to calculate the reference trajectory, a control point on the path which is the closest to the helicopter position must be found. Once this is done, the trajectory parameters can be calculated and used by the path controller (described in the next subsection) as set-points. A feedback method is used to find the control point, similar approach is used in (5).

The reference trajectory in this work is a vector represented by 9 components: $x(s)$, $y(s)$ and $z(s)$ represent the respective East, North positions and altitude of the helicopter relative to an initial point; $v_x(s)$, $v_y(s)$ and $v_z(s)$ the North, East and vertical velocity components; $\phi(s)$, $\theta(s)$ and $\psi(s)$ the roll, pitch and yaw angles; s is the path segment parameter. Details of the calculation of the parameter s through feedback can be found in (3).

The geometric path is composed of several segments represented by a 3D cubic polynomial. The motivation for using this type of curve is given in (17). A 3D geometric segment is represented by the following equation in a vector form:

$$\boldsymbol{P}(s) = \boldsymbol{A}s^3 + \boldsymbol{B}s^2 + \boldsymbol{C}s + \boldsymbol{D}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are 3D vectors defined by the boundary conditions calculated by the path planner and passed to the control mode, s=[0,1] is the parameter of the curve and $\boldsymbol{P} = [x, y, z]$. 0nce the parameter s is found the position coordinates x, y, z can be calculated.

In order to achieve the necessary tracking performance, the path curvature is fed forward in the roll control law. The target roll angle value is calculated according to the following formula:

$$\phi(s) = \frac{V^2}{R_{xy}(s)g}$$

where V is the helicopter speed, g is the gravity acceleration and $R_{xy}(s)$ is the local curvature radius of the path projected on the horizontal plane. The curvature radius of the path is a 3D vector and it is calculated analytically as explained in (3).

The same approach could be applied in order to calculate the target pitch angle $\theta(s)$. However, for the helicopter flight envelope we are interested in, the dynamics of the path curvature in the vertical direction is not very fast. Therefore, the feed forward term is not used for the pitch channel.

The yaw angle $\psi(s)$ is calculated from the tangent vector of the path. The tangent is projected on the horizontal (East-North) plane and used as reference signal for the yaw control.

The target path velocity ($V_{tar}(s)$) is derived from two input parameters: the cruise velocity V_c (desired velocity for the segment) and the final velocity V_f (velocity that the helicopter must have at the end of the segment). Both velocities are given by the path planners.

The calculation of $V_{tar}(s)$ along the path segment is divided into three phases: *acceleration*, *cruise* and *braking*. The acceleration phase is active only during execution of the first segment of the path. During this phase the velocity increases with a constant rate until the cruise velocity V_c is reached. Note that in this case V_{tar} depends on time rather than on the path parameter (s) since it is not important at which position of the path the acceleration phase is terminated.

The braking phase is active when the following condition is satisfied: the remaining path length $d_{end}(s)$ is equal to the distance required to brake the helicopter from V_c to V_f for given deceleration a_{brake}:

$$d_{end}(s) = \frac{|V_c^2 - V_f^2|}{2a_{brake}}$$

The target velocity in the braking phase is a function of $d_{end}(s)$:

$$V_{tar}(s) = \sqrt{|2d_{end}(s)a_{brake} + V_f^2|}$$

This guarantees achieving the desired velocity at the end of the path segment. In case $V_f > V_c$, the helicopter accelerates in order to reach V_f at the end of the path segment. The path planner assures the continuity of the velocity profile between segments. In order to make a coordinated turn a consistency check must be done with respect to the generated $V_{tar}(s)$. For such a maneuver, the helicopter must compensate the centripetal acceleration with a certain amount of roll angle. Because of safety reasons the maximum roll angle (ϕ_{max}) and maximum yaw rate (ω_{max}) are specified. Therefore, the maximum velocity during a turn maneuver is also restricted. The two velocity limits

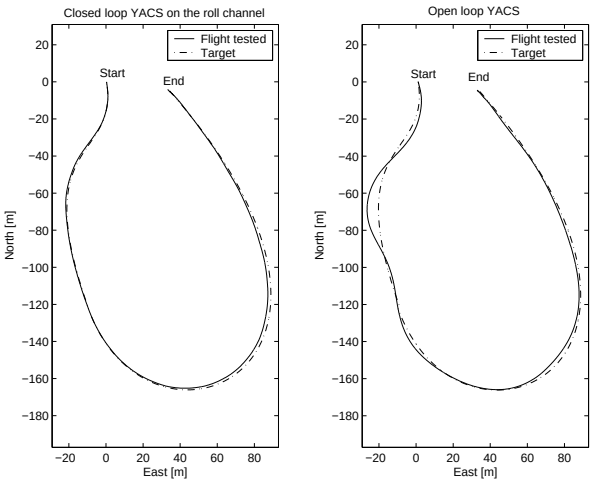


Figure 7: Comparison of path tracking performances using two different roll controller. The flight-test where performed at 36 km/h constant velocity for both paths.

are calculated as follows:

$$V_{max1}(s) = \sqrt{R_{xy}(s) g \phi_{max}}$$
$$V_{max2}(s) = \omega_{max}^2 R_{xy}(s)$$

The minimum between $V_{max1}(s)$, $V_{max2}(s)$ and $V_{tar}(s)$ is taken as target velocity by the path controller described in the next subsection. Thus, the calculated velocity is compatible with the curvature radius of the path.

5.2 Path Controller

Usually the control system for a helicopter consists of an *inner loop*, which is responsible for stabilizing the attitude, and the *outer loop*, which controls the position and velocity.

In our system we use the Yamaha Attitude Control System (YACS) provided with the RMAX helicopter for the attitude control. For the outer loop we use four decoupled PID controllers. Their outputs are used as input to the YACS system. In (3) the control approach for the RMAX has been described and experimental results provided.

In this section, we present results of a modified controller, which uses an additional feedback loop on the roll channel. Several experiments were done closing the roll angle loop around the YACS in order to test if it is feasible to improve path tracking precision without a complete redesign of the attitude controller.

The modified controller has been flight-tested on the RMAX helicopter at a constant velocity of 36km/h on a path with changing curvature. Such a path is typically used to navigate in areas with obstacles. The results are shown in Fig. 7 where the same path was tested both with the roll loop closed around the YACS, and without. Note, that at the beginning of the path when the dynamic response of the controller is more important because of the changing curvature, the control with additional feedback loop over the roll channel performs much better than the other one. The error for the closed loop controller in this part is below 1 meter, while for the other is around 6 meters. The results obtained during flight-tests show that the loop on the roll channel reduces the path tracking error, what makes the controller more suitable for obstacle-cluttered environments.

The following section describes interaction between the path following control mode with a Task Procedure responsible for executing a planned path.

6 Path Execution Mechanism

The key element of the framework is the path execution mechanism. It allows for seemless integration of hard real-time componets (e.g. the path following controller) with inherently non real-time deliberative services (e.g. the path planners).

The execution of the path provided by the path planner is divided into two parts, namely the Task Procedure and the Path Following controller. The standard path execution scheme in our architecture for static operational environments is depicted in Fig. 8 (key functional componets involved in navigation are drawn in black). A UAV mission is specified via a Task

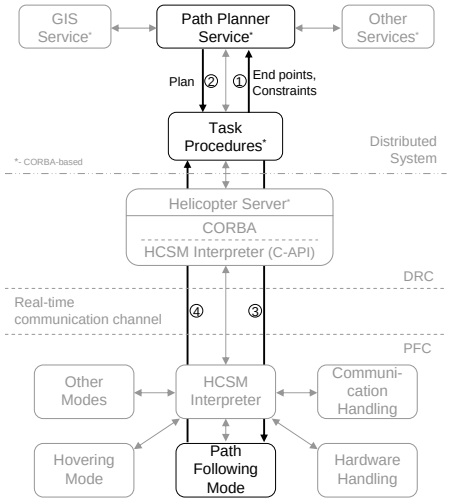


Figure 8: Path execution mechanism

Procedure in the reactive layer of our architecture, (perhaps after calling a task-based planner). For the purpose of this paper, a TP can be viewed as an augmented state machine.

For the case of flying to a waypoint, an instance of a navigation TP is created. First it calls the path planner service (step 1) with the following parameters: initial position, goal position, desired velocity and additional constraints.

If successful, the path planner (step 2) generates a segmented path which is represented as a set of cubic polynomial curves. Each segment is defined by start and end points, start and end directions, target velocity and end velocity. The TP sends the first segment (step 3) of the path via the control system interface and waits for the *Request Segment* event. It is generated by the HCSM responsible for the path execution as soon as the path following (PF) controller output is fed with a path segment.

When a *Request Segment* event arrives (step 4) the TP sends the next segment data to the HCSM which coordinates the path execution. This procedure is repeated (step 3-4) until the last segment is executed. However, because the high-level system is not implemented in hard real-time it may happen that the next segment does not arrive at the Control Kernel on time. In this case, the controller has a timeout limit after which it goes into safety braking mode in order to stop and hover at the end of the current segment. The timeout is determined by a velocity profile, current position and current velocity.

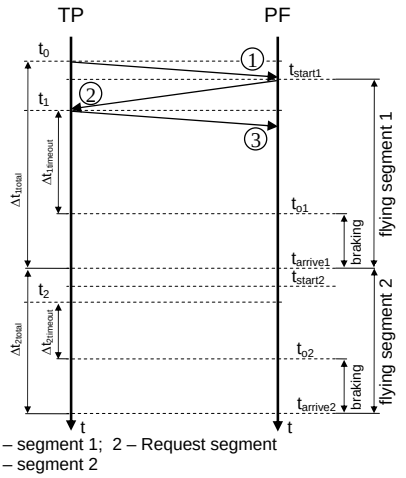


Figure 9: Execution timeline for trajectory consisting of 2 segments

Fig. 9 depicts a timeline plot of the execution of a trajectory (2 segments). At time t_0, a TP sends the first segment of the path to the PF controller and waits for a *Request segment* event which arrives immediately (t_1) after the helicopter starts to fly (t_{start1}). Typical time values for receiving a *Request segment* event ($t_1 - t_0$) are well below $200ms$. Time t_{o1} is the timeout

for the first segment which means that the TP has a $\Delta_{t_1 timeout}$ time window to send the next segment to the PF controller before it initiates a safety braking procedure. If the segment is sent after t_{o1}, the helicopter will start braking. In the current implementation, segments are not allowed to be sent after a timeout. This will be changed in a future implementation. In practice, the $\Delta_{t_1 timeout}$ time window is large enough to replan the path using the standard path planner. The updated segments are then sent to the PF controller transparently.

The described path execution mechanism allows for dynamic replacement of path segments if necessary. The following section describes the process of dynamic path replanning.

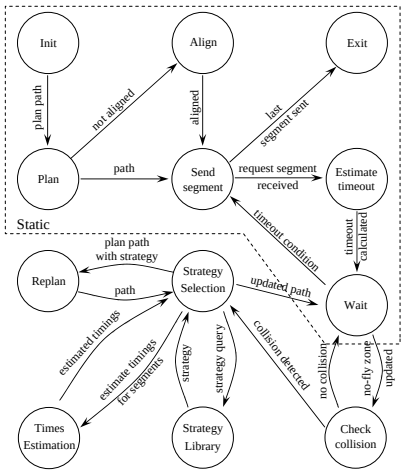


Figure 10: The dynamic path replanning automaton

7 Dynamic Replanning of the Path

The design of the path execution mechanism provides a method for feeding path segments to the PF controller iteratively. This is a particularly interesting part of the design because it gives the deliberative or decision-making layer of the architecture the opportunity to anticipate problems at longer temporal horizons and then to modify one or more segments in the original mission path plan based on any contingencies it discovers.

There are several services that are used during the path replanning stage. They are called when changes in the environment are detected and an update event is generated in the system. The augmented state machine associated with the TP used for the dynamic replanning of a path is depicted in Fig. 10. The TP takes a start and an end point and a target velocity as input. The TP then calls a path planning service (*Plan* state) which returns an initial path.

If the helicopter is not aligned with the direction of the flight, a command to align is sent to the controller (*Align* state).The TP then sends the first segment of the generated path to the PF controller (*Send segment* state) and calls the Prediction service to estimate a timeout for the current segment (*Estimate timeout* state). Based on the segment timeout and system latency, a condition is calculated for sending the next segment. If there is no change in the environment the TP waits (*Wait* state) until a timeout condition is true and then sends the next segment to the PF controller.

In case new information about newly added or deleted forbidden regions (no-fly zone updated) arrives, the TP checks if the current path is in collision with the updated world model (*Check Collision* state). If a collision is detected in one or more segments the TP calls a Strategy Selector service (*Strategy Selection* state) to determine which replanning strategy is the most appropriate to use at the time. The Strategy Selector service uses the Prediction service for path timings estimation (*Times Estimation* state) to get estimated timeouts, total travel times etc. It also uses the Strategy Library service (*Strategy Library* state) to get available replanning strategies that will be used to replan when calling the path planner (*Replan* state). The TP terminates when the last segment is sent.

All time estimations that have to do with paths or parts of paths are handled by the Prediction service. It uses the velocity profile of a vehicle and path parameters to calculate timeouts, total times, and combinations of those. For instance, in the case of flying a two-segment trajectory (see execution timeline in Fig. 9) it can estimate timeouts ($\Delta_{t_1 timeout}$, $\Delta_{t_2 timeout}$), total travel times ($\Delta_{t_1 total}$, $\Delta_{t_2 total}$) as well as a combined timeout for the first and the second segment (t_{o2}-t_1).

When part of a path is not valid anymore, the path planner service can be called in order to repair an existing plan or to create a new one. There are many strategies that can be used at that step which can give different results depending on the situation. The Strategy Library stores different replanning strategies including information about the replanning algorithm to

be used, the estimated execution time and the priority. Example strategies are shown in Fig. 11.

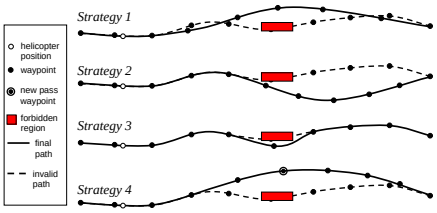


Figure 11: Examples of replanning strategies.

Strategy 1

Replanning is done from the next waypoint (start point of the next segment) to the final end point. This implies longer planning times and eventual replacement of collision-free segments that could be reused. The distance to the obstacle in this case is usually large so the generated path should be smoother and can possibly result in a shorter flight time.

Strategy 2

Segments up to the colliding one are left intact and replanning is done from the last collision-free waypoint to the final end point. In this case, planning times are cut down and some parts of the old plan will be reused. But since the distance to the obstacle is shorter than in the previous case, it might be necessary for the vehicle to slow down at the joint point of two plans, this can result in a longer flight time.

Strategy 3

Replanning is done only for colliding segments. The helicopter will stay as close to the initial path as possible.

Strategy 4

There can be many other strategies that take into account additional information that can make the result of the replanning better from a global perspective. An example is a strategy that allows new pass waypoints that should be included in the repaired plan.

Note that each of these strategies progressively re-uses more of the plan that was originally generated, thus cutting down on planning times but maybe producing less optimal plans. The decision as to which strategy to use is made by the Strategy Selector service. In the current implementation of the framework it uses a simple algoritm for chosing strategies based on user-predefined priorities.

More details on dynamic replanning are presented in (24). One example of using the dynamic replanning technique is presented in the following section.

8 Experimental results

In this section, we provide a description of two generic missions, instances of which were flown at the Swedish Rescue Services Agency facilities in Revinge in southern Sweden. The site, which is usually used by firefighters and other emergency rescue forces for training purposes, consists of a number of building structures, a road network and different types of terrain and vegetation (trees, bushes, etc.). A 3D map of the area is provided by an onboard geographical information system (GIS) which is used by the path planner service to generate collision-free paths according to the framework described in the previous sections.

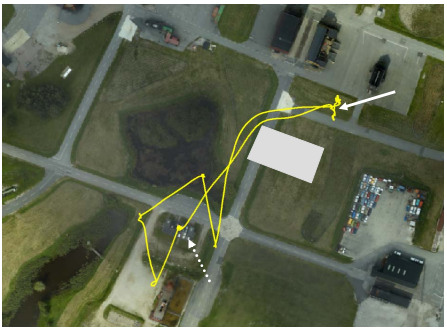


Figure 12: Mission 1. White solid arrow designates take off and landing position. White dotted arrow points to the target building. Gray polygonal area marks the no-fly zone created above the ground station vehicle. Solid line represents the actual flight path.

In the first mission, the UAV took off autonomously and hovered. It then flew towards a building previously designated by the ground operator. Upon arrival at the building, it gathered video footage of all the facades. It then flew back to home base and landed autonomously. The operator's task was to select a building of interest using a ground station user interface. The information was sent to the UAV and the mission plan was generated

Figure 13: Mission 1. Images captured during the mission. Clockwise, starting from the upper left corner of the figure: the North, West, South and Top view of the building.

on-board. As the helicopter was reaching successive hovering positions in front of each facade and over the building roof, the camera was controlled autonomously to keep the object of interest in the center of the image. This kind of mission was performed several times with different buildings chosen as observation targets. The logged flight-test data of one of the missions is plotted on the map in Fig. 12. Fig. 13 presents several frames taken from video footage from the mission demonstrations.

The second mission demonstrates the use of the dynamic replanning capability of the framework. The flight started with autonomous take off, and the helicopter began executing the planned path towards the designated waypoints. After arriving at the first one, the direction of flight changed to south and the ground operator added a no-fly zone intersecting the flight path. The information was sent to the helicopter and the on-board system activated the replanning mechanism. A new path was planned, and the flight continued avoiding the no-fly zone. After the helicopter arrived at the last waypoint, it was commanded to return to home base and land. Fig. 14 shows the logged flight-test data superimposed on the map of the area.

9 Conclusions and Future Work

A distributed hardware/software architecture has been described which includes a framework for integrating path planning techniques, a path following control mode, and a path execution mechanism which allow for UAV operation in obstacle-cluttered environments in addition to dynamic replanning of flight paths. The

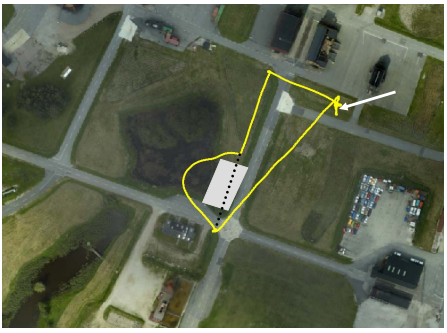


Figure 14: Mission 2. White solid arrow designates take off and landing position. Solid line represents the actual flight path executed counter-clockwise. Gray polygonal area marks the no-fly zone added during the flight by the ground operator. Black dotted line shows invalidated part of the path.

path planning algorithms are based on the use of sample-based probabilistic techniques which sacrifice completeness in plan generation for tractability. Details of the path following controller are provided in addition to experimental results using the framework. These results show that the controller keeps the UAV within one meter of the desired path during flights with a velocity of 10 m/s.

The proposed method for 3D trajectory execution views flight paths as sequences of segments where each segment is incrementally fed to the path following controller. This scheme supports dynamic replanning and replacement of flight path segments (e.g. because of adding no-fly zones). It also enhances the safety of UAV operations. The self-contained PFC computer's role is to request subsequent segments as a mission flight path is being flown. In cases where the new segments do not arrive on time, the PFC automatically switches to a hover control mode. This approach would help to avoid situations where a longer path which is passed to the controller becomes unacceptable as time progresses, but due to communication failure with the rest of the system, is not made aware of the problem. This could lead to potentially catastrophic situations.

The systems and techniques described here have been implemented and fully tested on our WITAS UAV systems. The framework proposed allows for the seamless coexistence of the hard real-time Control Kernel and the soft real-time high-level deliberative system by taking advantage of timing and other characteristics of both.

The Control Kernel is in charge of flight mode switching of the hybrid control system and coordinating the real-time communication among other things. It uses HCSMs to do this. The control kernel is easily extendible and allows for the implementation of additional functionality requiring a rigorous timing regime.

The use of CORBA provides a straightforward means of transparently distributing deliberative services such as task- and motion planners across different processors in a single platform, onto other airborne platforms or onto different ground control stations. Task procedures are used in the implementation of reactive behaviors which provide the software and conceptual *glue* between the lower control layer and the upper deliberative layer in the architecture. Because CORBA is being used, both reactive behaviors and deliberative functionalities can be implemented in many different languages as long as they have supporting IDL mappings.

Future work includes extending two of the services used in the dynamic replanning technique, namely, the Strategy Selector and the Strategy Library services. On the hardware side, inclusion of additional sensors necessary for perceiving the environment reactively are planned. In order to enhance the autonomy of the platform, the need for a 3D elevation map requirement must be loosened. This can be achieved by adding sensors enabling mapping and obstacle avoidance in unknown and dynamic environments. A new and enhanced version of the Control Kernel is also under evaluation. Among other features, it introduces data flow support into the state machine concept.

10 Acknowledgements

This work is funded by the Wallenberg Project under the WITAS UAV Project. Many members of the WITAS UAV project helped to make this work possible.

References

1. MARVIN: TU Berlin. http://pdv.cs.tu-berlin.de/MARVIN/.

2. M. La Civita. *Integrated Modeling and Robust Control for Full-Envelope Flight of Robotic*

Helicopters. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2003.

3. 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*, 2004.

4. 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.

5. 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.

6. E. Frazzoli, M. Dahleh, and E. Feron. Robust Hybrid Control for Autonomous Vehicles Motion Planning. In *Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 1999. Technical report LIDS-P-2468.*, 1999.

7. S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

8. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

9. MIT/Draper Autonomous Helicopter Project. http://web.mit.edu/whall/www/heli/.

10. E. N. Johnson and S.K. Kannan. Adaptive flight control for an autonomous unmanned helicopter. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002.

11. L. E. Kavraki, P. Švestka, J.C. Latombe, and M. H. Overmars. Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. *Proc. of the IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

12. J. J. Kuffner and S. M. LaValle. RRT-connect: An Efficient Approach to Single-Query Path Planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 995–1001, 2000.

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

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

15. 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*, 2004.

16. B. Mettler, M.B. Tischler, and T. Kanade. System identification modeling of a small-scale unmanned helicopter. *Journal of the American Helicopter Society*, October 2001.

17. P-O Pettersson. Using Randomized Algorithms for Helicopter Path Planning. *Lic. Thesis Linköping University.*, 2006.

18. 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.

19. AVATAR: USC Autonomous Flying Vehicle Project. http://www-robotics.usc.edu/∼avatar.

20. BEAR: Berkeley Aerorobot Team. http://robotics.eecs.berkeley.edu/bear/.

21. Georgia Tech UAV. http://www.ae.gatech.edu/labs/controls/uavrf/.

22. Hummingbird: Stanford University. http://sun-valley.stanford.edu/users/heli/.

23. CMU Autonomous Helicopter Project. www.cs.cmu.edu/afs/cs/project/chopper/www.

24. M. Wzorek and P. Doherty. Preliminary Report: Reconfigurable Path Planning for an Autonomous Unmanned Aerial Vehicle. In *Proceedings of the 24th Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG-05)*, 2005.

A.3 Paper III

Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing

Torsten Merz, Simone Duranti, and Gianpaolo Conte

Department of Computer and Information Science
Linköping University, SE-58183 Linköping, Sweden

Abstract. In this paper, we propose an autonomous precision landing method for an unmanned helicopter based on an on-board visual navigation system consisting of a single pan-tilting camera, off-the-shelf computer hardware and inertial sensors. Compared to existing methods, the system doesn't depend on additional sensors (in particular not on GPS), offers a wide envelope of starting points for the autonomous approach, and is robust to different weather conditions. Helicopter position and attitude is estimated from images of a specially designed landing pad. We provide results from both simulations and flight tests, showing the performance of the vision system and the overall quality of the landing.

1 Introduction

Many autonomous landing systems for Unmanned Aerial Vehicles (UAVs) are based on GPS and a dedicated close range sensor for accurate altitude measurement (radar altimeter, sonar, infrared or theodolites). However, in urban environments buildings and other obstacles disturb the GPS signal and can even cause loss of signal (multi-path effects, EM noise due to active emitters). Once the GPS signal is lost, the dead reckoning capability of affordable on-board inertial navigation systems does not allow precision navigation for more than few seconds, before diverging. Hence the need of a robust observation of the position: a vision system is self-contained, not jammable, and provides in the proposed implementation a position measurement one order of magnitude more accurate than standard GPS (cm accuracy or better). Estimating velocity from vision is difficult due to limited image frame rate and sensor resolution. In the proposed method velocity is estimated accurately and robustly by fusing vision position with the measurements of inertial sensors that usually belong to the standard instrumentation of an UAV for stabilization. The problem is to develop: (a) a vision system with a sufficient operating range to allow robust pose estimation from a reasonable distance at a sufficient rate with low latency using a landing pad of minimal size; (b) a method to fuse these data with inertial measurements; (c) a suitable flight controller. In an autonomous landing system all components have to match each other. For instance, for calculating vision estimates a

Fig. 1. The WITAS helicopter descending to the landing pad.

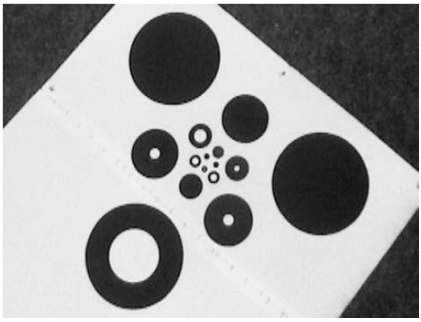


Fig. 2. Landing pad with reference pattern seen from the on-board camera.

proper trade-off between accuracy, range, latency, and rate has to be found optimizing the overall performance of the system.

Our method requires a special landing pad (Fig. 2). As unmanned helicopters usually operate from a designated home base this is not a real constraint. A precise and fast pan/tilt camera is used to extend the range of the vision system and decouple the helicopter attitude from the vision field. We developed a single camera solution, as multi-camera systems make the system more complex and expensive and don't offer significant advantages when using known landmarks. For the experimentation we used a helicopter platform (Fig. 1) which had been developed in the WITAS project [2,1].

Vision-based control of small size autonomous helicopters is an active area of research. A good overview of the state-of-the-art can be found in [7]. Our contribution to the landing problem consists of: (a) many demonstrated landings with a system only based on images from a single camera and inertial data using off-the-shelf computer hardware; (b) a wide envelope of starting points for the autonomous approach; (c) robustness to different weather conditions (wind, ambient light); (d) a quantitative evaluation of the vision system and the landing performance.

2 Vision System

The vision system consists of a camera mounted on a pan/tilt unit (PTU), a computer for image processing, and a landing pad (a foldable plate) with a reference pattern on its surface. In this section, we explain the design of the reference pattern, describe the image formation, and present the image processing algorithm.

The reference pattern is designed to fulfill the following criteria: fast recognition, accurate pose estimation for close and distant range, minimum size, and minimal asymmetry. We have chosen black circles on white background as they are fast to detect and provide accurate image features (Fig. 2). From

the projection of three circles lying on the corner points of an equilateral triangle the pose of an object is uniquely determined, assuming all intrinsic camera parameters are known. Circles are projected as ellipses, described by the center point $\boldsymbol{u}_e$, the semi-major axis l_a, the semi-minor axis l_b, and the semi-major axis angle θ_e The pose of the landing pad with respect to the camera coordinate system is estimated by minimizing the reprojection error of the extracted center points and semi-axes of the three ellipses. We use five circle triplets of different size (radius 2 to 32 cm, distance 8 to 128 cm) with common center point to achieve a wide range of possible camera positions. Each triplet is uniquely determined by a combination of differently sized inner circles.

A point ${}^{p}\tilde{\boldsymbol{x}}$ in the landing pad frame is projected on the image plane as follows:

$$\tilde{\boldsymbol{u}} = \boldsymbol{P}\,{}^{p}\tilde{\boldsymbol{x}} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} {}^{c}_{p}\boldsymbol{R} & {}^{c}\boldsymbol{t}_p \\ \boldsymbol{0}_3^T & 1 \end{pmatrix} {}^{p}\tilde{\boldsymbol{x}} \quad \tilde{\boldsymbol{u}} \in \mathcal{P}^2 \quad {}^{p}\tilde{\boldsymbol{x}} \in \mathcal{P}^3 \tag{1}$$

The extrinsic camera parameters are given by the three Euler angles of the rotation matrix ${}^{c}_{p}\boldsymbol{R}$ and the three components of the translation vector ${}^{c}\boldsymbol{t}_p$. We use a camera model with the following intrinsic parameters: "focal lengths" α_u and α_v in pixels, principal point (u_0, v_0), and four lens distortion coefficients . All intrinsic parameters are calibrated using Bouguet's calibration toolbox [3]. A conic in $\mathcal{P}^2$ is the locus of all points $\tilde{\boldsymbol{u}}$ satisfying the homogeneous quadratic equation $\tilde{\boldsymbol{u}}^T \boldsymbol{C}\,\tilde{\boldsymbol{u}} = 0$. The transformation of a circle $\boldsymbol{C}_p$ on the landing pad into an ellipse $\boldsymbol{C}_i$ in the image plane is given by[4]:

$$\boldsymbol{C}_i = (\boldsymbol{H}^{-1})^T \boldsymbol{C}_p\, \boldsymbol{H}^{-1} \tag{2}$$

The homography matrix $\boldsymbol{H}$ is the projection matrix $\boldsymbol{P}$ without third column ($z = 0$). We calculate the ellipse center and axes from $\boldsymbol{C}_i$ and represent the parameters in a common feature vector $\boldsymbol{c}$.

Fig. 3 shows a data flow diagram of the vision system. Round-edged boxes represent image processing functions, sharp-edged boxes indicate independent processes, and dashed lines show trigger connections. Closed contours are extracted from gray-level images using a fast contour following algorithm with two parameters: edge strength and binarization threshold. The latter is calculated from the intensity distribution of the reference pattern. In the contour list we search for the three biggest ellipses belonging to a circle triplet. Ellipse parameters are estimated by minimizing the algebraic distance of undistorted contour points to the conic using SVD [4,6]. After having found three ellipses, the corresponding contours are resampled with sub-pixel accuracy. A coarse pose is estimated based on the ratio of semi-major axes

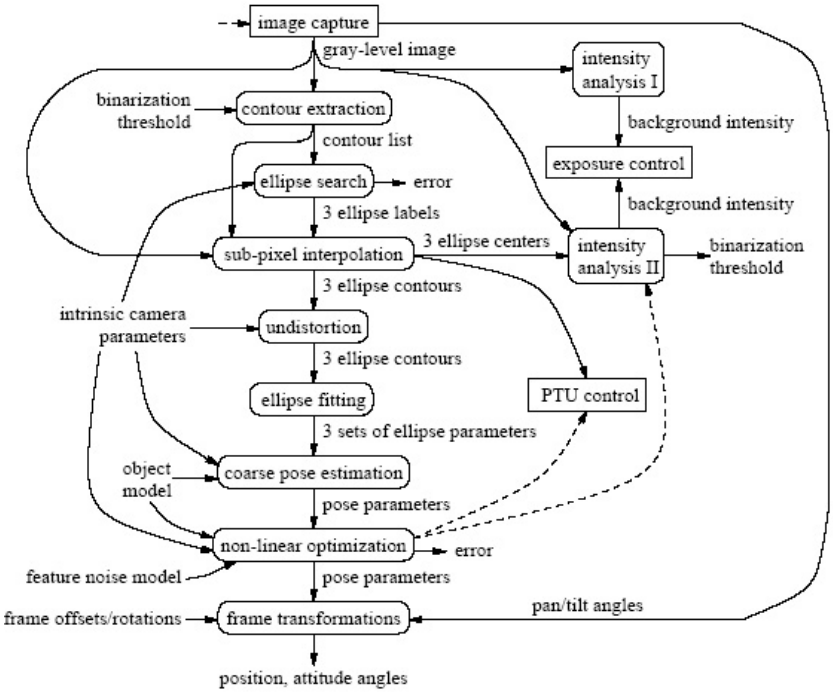


Fig. 3. Dataflow in the vision system.

and circle radii. The estimation is optimized by minimizing the reprojection error:

$$\min_{\boldsymbol{a}} \sum_{i=1}^{12} \left(\frac{d_i - c_{i \bmod 4}}{\sigma_{i \bmod 4}} \right)^2 \quad \begin{array}{l} \boldsymbol{d} = (u_{e1}, v_{e1}, l_{a1}, l_{b1}, \cdots, u_{e3}, v_{e3}, l_{a3}, l_{b3}) \\ \boldsymbol{\sigma} = (\sigma_c, \sigma_c, \sigma_l, \sigma_l) \\ \boldsymbol{c} = \left(\hat{u}_e(\boldsymbol{a}), \hat{v}_e(\boldsymbol{a}), \hat{l}_a(\boldsymbol{a}), \hat{l}_b(\boldsymbol{a}) \right) \end{array} \tag{3}$$

This function is non-linear and minimized iteratively using the fast-converging Levenberg-Marquardt method [6]. It's initialized with the pose parameters from the first estimate. The uncertainties of the ellipse centers σ_c and axes σ_l are known from separate noise measurements. Finally, the pose parameters are converted to helicopter position and attitude using angles from the PTU and known frame offsets and rotations. The PTU control runs in parallel to the image processing using pixel coordinates of the pattern center as input, aiming at centering the landing pad in the frame as soon as it's localized.

Two methods for analyzing image intensities are implemented. The first estimates the background intensity of the reference pattern based on the assumption being the brightest surface in the image. When the landing pad is detected, the second method is applied. It computes the background inten-

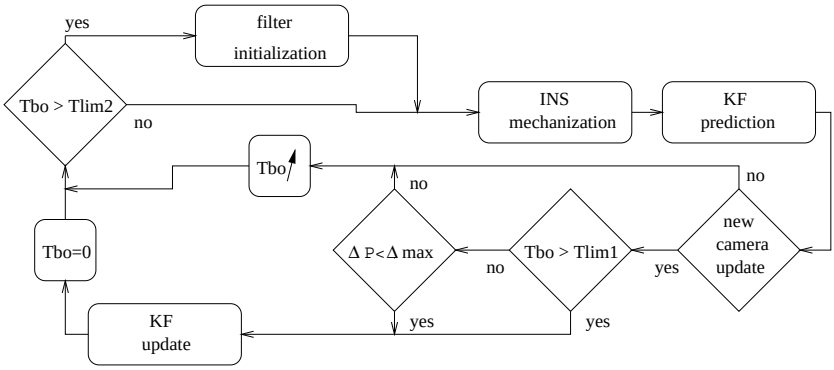


Fig. 4. The filter architecture.

sity and the binarization threshold based on the intensity distribution of the pattern. The exposure controller controls the camera shutter time and iris aiming at keeping the background intensity in a certain range.

3 Sensor Fusion

The position and attitude estimates provided by the vision system can not be fed directly into the controller due to their intrinsic lack of robustness: the field of view can be temporarily occluded (for example by the landing gear), the illumination conditions can change dramatically just by moving few meters (sun reflections, shades, etc.). On the other hand, vision readings are very accurate, when available.

Hence, a navigation filter based on a Kalman filter (KF) has been developed, fusing highly accurate 3D position estimates from the vision system with inertial data provided by the on-board accelerometers and angular rate gyros. Besides filtering out a large part of the noise and outliers, the filters provides a satisfying dead reckoning capability, sufficient to complete the landing even when the vision system is "blind"[1], see Fig. 8.

The implementation of the KF is done using the *error state space* or *indirect* formulation with *feedback mechanization* (Fig. 4). The states of the filter are the estimated inertial navigation system (INS) errors. The three observations are given by the difference between the INS position and the position from the vision system (lateral, longitudinal and vertical position relative to the pattern). The advantage of the indirect formulation versus the direct formulation (position, velocity and attitude are among the state

[1] During the last 50 cm before touch down the vision system is often "blind" due to two factors: (a) the shade of the helicopter covers part of the pattern at touch down, and (b) when the distance of the camera to the pattern is very small it is very hard for the controller of the pan/tilt unit to keep the pattern in the picture.

variables in the filter) lies in the fact that the INS errors have much slower dynamics than the navigation variables and are very well described by linear equations. In this particular application we have to deal with black-out camera periods in the order of seconds. The indirect KF is quite robust in this respect, in fact an effective indirect filter can be developed with a sample periods of the order of half a minute [5]. The estimated errors are fed back into the mechanization algorithm to avoid unbounded growth of the INS errors.

The inertial measuring unit (IMU) used in this application is integrated in the Yamaha Attitude Sensor (YAS) and it is composed of three accelerometers and three rate gyros. The output rate is 200 Hz for the gyros and 66 Hz for the accelerometers. The filter runs at 50 Hz and the inertial sensors are sampled at the same frequency. Both gyro and accelerometer outputs are prefiltered at 25 Hz to take into account the information available between the filter samples.

The filtered IMU outputs are used in the INS mechanization step to calculate the position, velocity and attitude by solving the inertial navigation equations (4) where $\boldsymbol{r}^n$ and $\boldsymbol{v}^n$ are the position and velocity vectors, $\boldsymbol{C}_b^n$ is the direction cosine matrix of the attitude angles. $\boldsymbol{f}^b$ and $\boldsymbol{\Omega}_{ib}^b$ are the accelerometers and gyros outputs, $\boldsymbol{g}^n$ is the gravity vector and $\boldsymbol{\omega}_{ie}^n$ the Earth rotation rate.

$$\begin{aligned} \dot{\boldsymbol{r}}^n &= \boldsymbol{v}^n \\ \dot{\boldsymbol{v}}^n &= \boldsymbol{C}_b^n \boldsymbol{f}^b - (2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n) \times \boldsymbol{v}^n + \boldsymbol{g}^n \\ \dot{\boldsymbol{C}}_b^n &= \boldsymbol{C}_b^n (\boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{in}^b) \end{aligned} \quad (4)$$

$$\begin{aligned} \delta\dot{\boldsymbol{r}} &= -\boldsymbol{\omega}_{en} \times \delta\boldsymbol{r} + \delta\boldsymbol{v} \\ \delta\dot{\boldsymbol{v}} &= -(\boldsymbol{\omega}_{ie} + \boldsymbol{\omega}_{in}) \times \delta\boldsymbol{v} - \boldsymbol{\psi} \times f + \delta\boldsymbol{a} \\ \dot{\boldsymbol{\psi}} &= -\boldsymbol{\omega}_{in} \times \boldsymbol{\psi} \\ \delta\dot{\boldsymbol{a}} &= -\beta\delta\boldsymbol{a} \end{aligned} \quad (5)$$

Several filter configurations have been tested, the final implementation is a 12-state KF with 9 navigation error states and 3 accelerometer biases. The dynamic model of the KF is based on the error dynamics equations (5) where $\delta\boldsymbol{r}$, $\delta\boldsymbol{v}$ and $\boldsymbol{\psi}$ are the position, velocity and attitude error vectors and $\delta\boldsymbol{a}$ are the accelerometer biases.

In Fig. 4 the filter architecture is shown. The black-out time (T_{bo}) is the time elapsed since the last valid update. When a new update from the vision system is available, it is first compared with the predicted position and if the difference (ΔP) is smaller than the maximum allowed tolerance (Δ_{max}) it is passed to the KF as a new observation. This consistency check is active only if $T_{bo} < T_{lim1}$ because the uncertainty of the system increases each time the update is not made. The KF prediction equations are applied at each time step. When there is a black-out from the vision system the uncertainty of the INS error system represented by its covariance grows unbounded until it's not possible to believe in it anymore. T_{lim1} is the maximum black-out time after that the new vision update is passed directly to the filter. The covariance update equation reduces the covariance of the error system and the decrease is large when the covariance of the measurement is small. The covariance of the vision system measurement is quite small, this means that

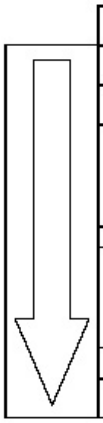
immediately after the update the uncertainty of the INS error system is low and the consistency check can be activated again. Of course, it can happen that the first update after the black-out is an outlier and in this case it can't be detected. For stability reasons the landing is aborted after a maximum black-out time T_{lim2} and the filter is reinitialized.

4 Flight Controls

The requirements set on the flight control system during landing are the following:

1. The landing mode should be engaged from any point where the landing pad is visible, that means approximately within a 20 m radius hemisphere, centered on the pattern.
2. Once the landing mode is engaged, the helicopter state should be compatible with the proper functionality of the vision system, until touchdown, this means that during the approach phase the following should be considered : (a) the helicopter's position and attitude should not be such as to cause physical occlusion of the visual field; this may happen due to the landing gear skids or the mechanical limitations of the pan/tilt unit; (b) the regions where the accuracy of the vision system is worst should be avoided, if possible; (c) the helicopter velocity and angular rates should not saturate the pan/tilt unit capability for compensation: too high angular rates of the visual beam may result in blurred images; (d) the position of the dominant light source (sun) should be considered, to avoid full reflections.
3. The wind direction has to be taken into account: tailwind landings should be avoided.
4. The control system should be dimensioned for wind levels up to 10 m/s.
5. The engine should be shut down autonomously, once touch-down is detected. The detection should be timely, since early detections cause high touch down loads and late detections can cause ground resonance.
6. The vertical velocity at touch down should be of the same order of magnitude as a proper manual landing.

In the following, the landing procedure is described. Fig. 5 shows the sequence of control modes and the triggering conditions. As soon as the navigation filter provides state estimates, the helicopter turns towards the pattern to guarantee occlusion-free view of the pattern and flies to a point located 5 meters on the vertical of the desired touch down point (P_{TD}). Once the helicopter is on top of P_{TD}, the heading is changed for landing, taking into consideration the illumination conditions (sun from the side is optimal) and the wind conditions (optimum with head-wind). The final descent is conducted at a constant sink rate of 20 cm/s. At 8 cm from the ground, the



Mode	Horizontal control	Yaw control	Altitude control	Logical condition for Mode Transition
READY				
AIM	hold	$\psi \rightarrow \psi_{AIM}$	hold	$\lvert\psi - \psi_{AIM}\rvert < 5°$
APPROACH	linearly moving to P_D @ $V_{HOR} < 0.5$ m/s	hold	Descending to PD $h \rightarrow h_{D1}$ (5 m) @ VZ = -50 cm/s	$\lvert P - P_{D1}\rvert < 2$ m $\lvert h - h_{D1}\rvert < 0.4$ m $\lvert V_Z\rvert < 10$ cm/s $\lvert V_{HOR}\rvert < 0.3$ m $\lvert\psi - \psi_{AIM}\rvert < 3°$
ALIGN	hold	$\psi \rightarrow \psi_{TOUCHDOWN}$	Hold	$(\psi - \psi_{TOUCHDOWN}) < 3°$
DESCEND	hold	hold	$h \rightarrow h_{D2}$ (1 m) @ VZ = -20 cm/s	$\lvert P - P_{D2}\rvert < 0.25$ m $\lvert h - h_{D2}\rvert < 0.1$ m $\lvert V_{HOR}\rvert < 0.3$ m $\lvert V_Z\rvert < 0.1$ m/s vision data VALID
TOUCH DOWN	hold	hold	VZ = -20 cm/s	$h < 0.1$ m
SHUT OFF	hold	hold	Throttle back descending ramp	

Fig. 5. Mode sequence leading to touch down. Note that all transitions are one-directional: once a mode is exited it can not be re-entered.

throttle command is ramped down, inducing loss of lift and touch down.

The control laws of the helicopter consist of an inner loop (pitch, roll and yaw angle control, and vertical velocity control) and an outer loop (position and velocity control). The inner loop consists of the Yamaha Attitude Control System (YACS), the properties of which have been identified with a dedicated system identification session. The control equations of the outer loop can be summarized as following:

$$\begin{aligned} \theta_C &= K_{px}\delta X + K_{pvx}\delta V_X + K_{ivx}\delta V_{X\mathrm{sum}} \\ \Delta\phi_C &= K_{py}\delta Y + K_{pvy}\delta V_Y + K_{ivy}\delta V_{Y\mathrm{sum}} \\ V_{ZC} &= K_{ivz}\delta V_{Z\mathrm{sum}} + K_{pvz}(V_{Z\mathrm{target}} - V_Z) \\ V_{Z\mathrm{target}} &= \mathrm{limit}(0.75\delta Z, V_{Z\mathrm{min}}, V_{Z\mathrm{max}}) \\ \omega_C &= \mathrm{limit}(K_{pw}\delta\psi, -26\,\mathrm{deg/s}, 26\,\mathrm{deg/s}) \end{aligned} \tag{6}$$

where the subscripted K are control gains, the δ are control errors, the subscript *sum* indicates the integral terms, θ_C is the commanded pitch angle, $\Delta\phi_C$ is the commanded roll angle variation, ω_C is the commanded yaw rate and V_{ZC} is the commanded vertical velocity[2].

5 Experimental Results

The helicopter used for experimentation is a slightly modified Yamaha RMAX (Fig. 1). It has a total length of 3.6 m (incl. main rotor) and a take-off weight

[2] During the descent and touch down phases, the gains of the velocity terms (K_{pvx} and K_{pvx}) are increased by one fifth and the integral terms in the horizontal control are activated, for faster and more precise position control.

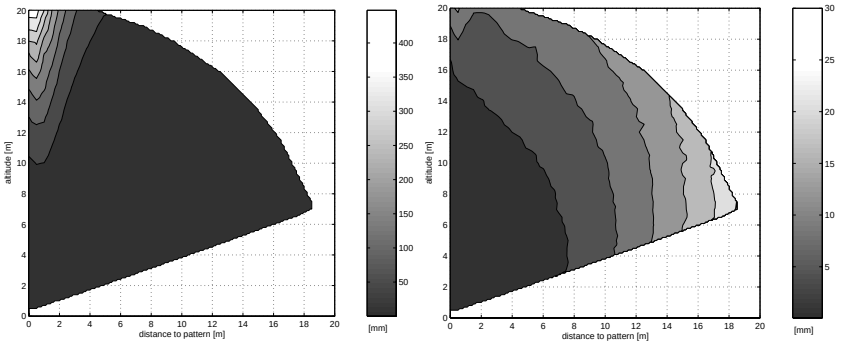


Fig. 6. RMS error in horizontal (left) and vertical position (right) from simulation.

of 95 kg, including 30 kg available for payload. The vision navigation system consists of two PC104 stacks with PIII 700 MHz processors, the inertial sensors of the YAS, and a single standard CCD camera with approx. 45 degrees horizontal angle of view which is mounted on an off-the-shelf pan/tilt unit (PTU). One of the two computers is dedicated to sensor management and low level control of the helicopter, the other one for image processing and control of the camera and the PTU. The two computers communicate over a RS232C serial link. They are built inside a shock and vibration isolated box, which also includes a precision GPS, a barometric altitude sensor, a compass, a video recorder, a video transmitter, and a wireless Ethernet bridge. The PTU is mechanically limited to 111 degrees tilt and ±180 degrees pan, the max. angular rate is 300 degrees/s and the resolution 0.051 degrees. It is mounted on a vibration isolating platform on the underside of the helicopter body.

We estimated the RMS error of the vision system in position and attitude depending on the relative position to the pattern in leveled flight. For each position 1000 samples were generated using a feature noise model that included noise from image formation, digitization, and segmentation. We developed a method to analyze noise in ellipse center position and semi-axis length. Errors introduced by the transformation from the camera frame into the body frame were not considered in simulation. Fig. 6 shows the RMS errors (1σ) in horizontal and vertical position. The error doesn't change much in the simulated envelope (a 20 m radius hemisphere, with a "blind" sector from azimuth 0 to 15 degrees) due to different triplet sizes and sub-pixel feature extraction. For pitch and roll the error behaves similar to the horizontal error, with a maximum RMS value of ≈1 degree, the error in heading is negligible.

The actual accuracy of the vision system was evaluated through an in-flight comparison with a navigation solution based on the YAS and a precision RTK GPS which supplied horizontal/vertical position with 10 mm/15 mm

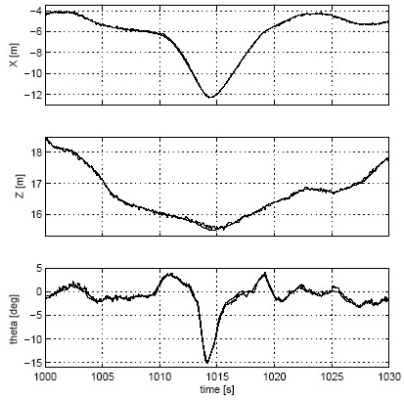


Fig. 7. Position and heading estimates from the vision system vs. estimates based on YAS and precision RTK GPS observations.

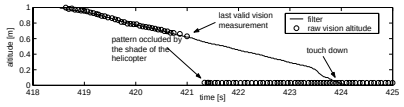


Fig. 8. Altitude estimates from the navigation filter when losing vision.

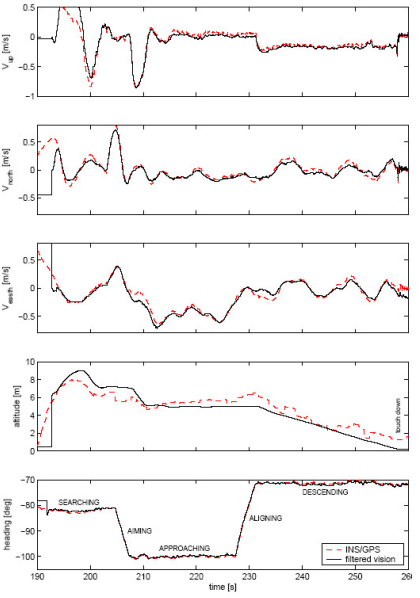


Fig. 9. Time histories and control modes during autonomous landing. The dashed lines show readings from a standard INS/DGPS unit.

uncertainty (1σ). We found good agreement between measured and simulated errors. Fig. 7 shows timeplots for distance, altitude, and pitch angle at typical starting points for autonomous landing. Position and attitude estimates were provided with an average rate of 20 Hz (using 384×288 pixels images) and an average latency of 110 ms (including delays from capturing PAL signals).

Some tens of autonomous landings were conducted from different relative positions to the landing pad within the specified envelope, on grass and snow fields, with different wind and illumination conditions. A sample of the results is available in Fig. 10 . The vertical velocity at touch down ranged between 18 and 35 cm/s, this corresponds to load factors of about 1.4 g on grass fields. The horizontal velocity at touch down was in the order of magnitude of 15 cm/s. The average touch down point precision was about 42 cm (13 % of rotor diameter). Thanks to the pan/tilt camera and a robust controller, considerable wind levels can be handled. Successful landings have been performed with wind levels on ground up to 30 km/h (2 min average), with gusts of 45 km/h.

Test ID	Wind Speed [km/h]	V_Z [cm/s]	V_{NORTH} [cm/s]	V_{EAST} [cm/s]	V_{HOR} [cm/s]	P_{TD} MEASURED (X_{TD}, Y_{TD}) [cm]	P_{TD} TARGET (X_{TD}, Y_{TD}) [cm]	$P_{TD\text{-}ERROR}$ [cm]	ψ_{TD} [°]	$\psi_{TD\text{-}TARGET}$ [°]
128/2/1	15	**-31**	-5	-24	**25**	(51,-17)	(94,-34)	**46**	161	160
128/3/1	14	**-37**	-20	17	**26**	(49,-37)	(94,-34)	**45**	162	160
128/4/1	13	**-35**	0	-1	**1**	(43,-31)	(87,-50)	**48**	151	150
128/4/2	13	**-24**	-23	-13	**26**	(91,-28)	(87,-50)	**22**	152	150
506/5/1	30	**-28**	-7	0	**7**	(41,-33)	(50,-87)	**54**	122	120
506/6/1	22	**-25**	-5	-11	**12**	(47,-53)	(50,-87)	**34**	123	120
506/6/2	26	**-27**	1	0	**1**	(28,-41)	(50,-87)	**50**	121	120
506/8/1	30	**-18**	-7	-17	**18**	(63,-47)	(50,-87)	**42**	121	120

Fig. 10. Flight test results from several autonomous landings.

6 Acknowledgements

We thank Piotr Rudol and Mariusz Wzorek for providing the pan-tilt controller and Anke Knöppler for work on camera calibration. This work was supported by the Wallenberg Foundation, Sweden.

References

1. P. Doherty. Advanced research with autonomous unmanned aerial vehicles. In *Proc. of the 9th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 731–732, June 2004.
2. 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 7th International Symposium on Distributed Autonomous Robotic Systems*, pages 221–230, June 2004.
3. Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab . http://www.vision.caltech.edu/bouguetj/calib_doc.
4. K. Kanatani. *Geometric Computation for Machine Vision*. Oxford University Press, 1995.
5. P.S. Maybeck. Stochastic models, estimation, and control. In *Mathematics in Science and Engineering*, volume 141-1, pages 289–367. Academic Press, 1979.
6. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.
7. S. Saripalli, J.F. Montgomery, and G. Sukhatme. Visually-guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–380, June 2003.

LINKÖPINGS UNIVERSITET

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

Datum
Date

March 1st, 2007

Språk
Language

[ ] Svenska/Swedish
[X] Engelska/English
[ ] _____

Rapporttyp
Report category

[X] Licentiatavhandling
[ ] Examensarbete
[ ] C-uppsats
[ ] D-uppsats
[ ] Övrig rapport

_____

URL för elektronisk version

ISBN 978-91-85715-35-0

ISRN LiU-Tek-Lic-2007:16

Serietitel och serienummer
Title of series, numbering

ISSN 0280-7971

Linköping Studies in Science and Technology

Thesis No. 1307

Titel
Title

Navigation Functionalities for an Autonomous UAV Helicopter

Författare
Author

Gianpaolo Conte

Sammanfattning
Abstract

This thesis was written during the WITAS UAV Project where one of the goals has been the development of a software/hardware architecture for an unmanned autonomous helicopter, in addition to autonomous functionalities required for complex mission scenarios. The algorithms developed here have been tested on an unmanned helicopter platform developed by Yamaha Motor Company called the RMAX.

The character of the thesis is primarily experimental and it should be viewed as developing navigational functionality to support autonomous flight during complex real-world mission scenarios. This task is multidisciplinary since it requires competence in aeronautics, computer science and electronics.

The focus of the thesis has been on the development of a control method to enable the helicopter to follow 3D paths. Additionally, a helicopter simulation tool has been developed in order to test the control system before flight-tests.

The thesis also presents an implementation and experimental evaluation of a sensor fusion technique based on a Kalman filter applied to a vision based autonomous landing problem.

Extensive experimental flight-test results are presented.

Nyckelord
Keywords

Unmanned Aerial Vehicle, Control System, Path Following, Path Planning, Sensor Fusion, Vision Based Landing, Kalman Filter, Real-Time.

Department of Computer and Information Science
Linköpings universitet

Linköping Studies in Science and Technology
Faculty of Arts and Sciences - Licentiate Theses

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)

No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.

No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.

No 104 **Shamsul I. Chowdhury: S**tatistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.

No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.

No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.

No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.

No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.

No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.

No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.

No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.

No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.

No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.

No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.

No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.

No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.

No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.

No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.

No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.

No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.

No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.

No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.

No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.

No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.

No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.

No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.

No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.

No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge-Bases, 1991.

No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.

No 318 **Lena Strömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.

No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.

No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.

No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.

No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.

No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Sytems, 1992.

No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.

No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.

No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.

No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.

No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.

No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.

No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.

No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.

No 402 **Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406 **Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414 **Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agentteoretiskt perspektiv, 1994.
No 417 **Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436 **Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437 **Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440 **Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94 **Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94 **Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441 **Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446 **Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450 **Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451 **Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452 **Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455 **Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94 **Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462 **Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464 **Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469 **Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473 **Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475 **Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476 **Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478 **Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95 **Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylkprogram, 1995.
No 482 **Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488 **Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489 **Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497 **Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498 **Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503 **Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95 **Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95 **Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513 **Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517 **Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518 **Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522 **Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538 **Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545 **Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546 **Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96 **Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549 **Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550 **Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557 **Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558 **Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561 **Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563 **Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567 **Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.
No 575 **Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
No 576 **Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
No 587 **Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
No 589 **Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 591 **Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
No 595 **Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
No 597 **Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

No 598 **Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.
No 599 **Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
No 607 **Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
No 609 **Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
FiF-a 4 **Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
FiF-a 6 **Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
No 615 **Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
No 623 **Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
No 626 **David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
No 627 **Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
No 629 **Gunilla Ivefors:** Krigsspel coh Informationsteknik inför en oförutsägbar framtid, 1997**.**
No 631 **Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
No 639 **Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.
No 640 **Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
No 643 **Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.
No 653 **Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.
FiF-a 13 **Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 674 **Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
No 676 **Jan Håkegård**: Hiera rchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
No 668 **Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
No 675 **Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
FiF-a 14 **Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
No 695 **Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
No 700 **Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
FiF-a 16 **Marie-Therese Christiansson:** Inter-organistorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 712 **Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
No 719 **Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
No 723 **Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
No 725 **Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
No 730 **Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
No 731 **Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
No 733 **Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
No 734 **Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
FiF-a 21 **Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
FiF-a 22 **Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
No 737 **Jonas Mellin:** Predictable Event Monitoring, 1998.
No 738 **Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
FiF-a 25 **Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
No 742 **Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
No 748 **Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
No 751 **Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader,1999.
No 752 **Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
No 753 **Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
No 754 **Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
No 766 **Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769 **Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775 **Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30 **Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787 **Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788 **Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790 **Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791 **Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800 **Anders Subotic:** Software Quality Inspection, 1999.
No 807 **Svein Bergum**: Managerial communication in telework, 2000.

No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
No 823 **Lars Hult:** Publika Gränsytor - ett designexempel, 2000.
No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.
No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
FiF-a 37 **Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av ”knowledge management”, 2000.
FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.
FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
No 863 **Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management inInformation-providing Dialogue systems, 2001.
FiF-a 47 **Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer: Informationssystemarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
No 906 **Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.
No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
FiF-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
FiF-a-51 **Per Oscarsson:**Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
No 938 **Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.
No 942 **Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
FiF-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.
No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002
No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.
No 1001 **Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.
No 988 **Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.
FiF-a 62 **Lennart Ljung:** Utveckling av en projektivitetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.
No 1003 **Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.
No 1005 **Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.
No 1008 **Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.
No 1010 **Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.
No 1015 **Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.
No 1018 **Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.
No 1022 **Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.
FiF-a 65 **Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.

No 1024 **Aleksandra Tešanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.
No 1034 **Arja Vainio-Larsson**: Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.
No 1033 **Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
FiF-a 69 **Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.
No 1049 **Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.
No 1052 **Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.
No 1054 **Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.
FiF-a 71 **Emma Eliason:** Effektanalys av IT-systems handlingsutrymme, 2003.
No 1055 **Carl Cederberg:** Experiments in Indirect Fault Injection with Open Source and Industrial Software, 2003.
No 1058 **Daniel Karlsson:** Towards Formal Verification in a Component-based Reuse Methodology, 2003.
FiF-a 73 **Anders Hjalmarsson:** Att etablera och vidmakthålla förbättringsverksamhet - behovet av koordination och interaktion vid förändring av systemutvecklingsverksamheter, 2004.
No 1079 **Pontus Johansson:** Design and Development of Recommender Dialogue Systems, 2004.
No 1084 **Charlotte Stoltz:** Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
FiF-a 74 **Björn Johansson:** Deciding on Using Application Service Provision in SMEs, 2004.
No 1094 **Genevieve Gorrell:** Language Modelling and Error Handling in Spoken Dialogue Systems, 2004.
No 1095 **Ulf Johansson:** Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
No 1099 **Sonia Sangari:** Computational Models of Some Communicative Head Movements, 2004.
No 1110 **Hans Nässla:** Intra-Family Information Flow and Prospects for Communication Systems, 2004.
No 1116 **Henrik Sällberg:** On the value of customer loyalty programs - A study of point programs and switching costs, 2004.
FiF-a 77 **Ulf Larsson:** Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
No 1126 **Andreas Borg:** Contribution to Management and Validation of Non-Functional Requirements, 2004.
No 1127 **Per-Ola Kristensson:** Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
No 1132 **Pär-Anders Albinsson:** Interacting with Command and Control Systems: Tools for Operators and Designers, 2004.
No 1130 **Ioan Chisalita:** Safety-Oriented Communication in Mobile Networks for Vehicles, 2004.
No 1138 **Thomas Gustafsson:** Maintaining Data Consistency im Embedded Databases for Vehicular Systems, 2004.
No 1149 **Vaida Jakoniené:** A Study in Integrating Multiple Biological Data Sources, 2005.
No 1156 **Abdil Rashid Mohamed**: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
No 1162 **Adrian Pop:** Contributions to Meta-Modeling Tools and Methods, 2005.
No 1165 **Fidel Vascós Palacios:** On the information exchange between physicians and social insurance officers in the sick leave process: an Activity Theoretical perspective, 2005.
FiF-a 84 **Jenny Lagsten:** Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
No 1166 **Emma Larsdotter Nilsson:** Modeling, Simulation, and Visualization of Metabolic Pathways Using Modelica, 2005.
No 1167 **Christina Keller:** Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.
No 1168 **Cécile Åberg:** Integration of organizational workflows and the Semantic Web, 2005.
FiF-a 85 **Anders Forsman:** Standardisering som grund för informationssamverkan och IT-tjänster - En fallstudie baserad på trafikinformationstjänsten RDS-TMC, 2005.
No 1171 **Yu-Hsing Huang:** A systemic traffic accident model, 2005.
FiF-a 86 **Jan Olausson:** Att modellera uppdrag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
No 1172 **Petter Ahlström**: Affärsstrategier för seniorbostadsmarknaden, 2005.
No 1183 **Mathias Cöster**: Beyond IT and Productivity - How Digitization Transformed the Graphic Industry, 2005.
No 1184 **Åsa Horzella**: Beyond IT and Productivity - Effects of Digitized Information Flows in Grocery Distribution, 2005.
No 1185 **Maria Kollberg**: Beyond IT and Productivity - Effects of Digitized Information Flows in the Logging Industry, 2005.
No 1190 **David Dinka**: Role and Identity - Experience of technology in professional settings, 2005.
No 1191 **Andreas Hansson**: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
No 1192 **Nicklas Bergfeldt:** Towards Detached Communication for Robot Cooperation, 2005.
No 1194 **Dennis Maciuszek:** Towards Dependable Virtual Companions for Later Life, 2005.
No 1204 **Beatrice Alenljung:** Decision-making in the Requirements Engineering Process: A Human-centered Approach, 2005
No 1206 **Anders Larsson:** System-on-Chip Test Scheduling and Test Infrastructure Design, 2005.
No 1207 **John Wilander:** Policy and Implementation Assurance for Software Security, 2005.
No 1209 **Andreas Käll:** Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
No 1225 **He Tan:** Aligning and Merging Biomedical Ontologies, 2006.
No 1228 **Artur Wilk:** Descriptive Types for XML Query Language Xcerpt, 2006.
No 1229 **Per Olof Pettersson:** Sampling-based Path Planning for an Autonomous Helicopter, 2006.
No 1231 **Kalle Burbeck:** Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, 2006.
No 1233 **Daniela Mihailescu:** Implementation Methodology in Action: A Study of an Enterprise Systems Implementation Methodology, 2006.
No 1244 **Jörgen Skågeby:** Public and Non-public gifting on the Internet, 2006.
No 1248 **Karolina Eliasson:** The Use of Case-Based Reasoning in a Human-Robot Dialog System, 2006.
No 1263 **Misook Park-Westman:** Managing Competence Development Programs in a Cross-Cultural Organisation-What are the Barriers and Enablers, 2006.
FiF-a 90 **Amra Halilovic:** Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.
No 1272 **Raquel Flodström**: A Framework for the Strategic Management of Information Technology, 2006.

No 1277 **Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Embedded Systems, 2006.
No 1283 **Håkan Hasewinkel:** A Blueprint for Using Commercial Games off the Shelf in Defence Training, Education and Research Simulations, 2006.
FiF-a 91 **Hanna Broberg:** Verksamhetsanpassade IT-stöd - Designteori och metod, 2006.
No 1286 **Robert Kaminski:** Towards an XML Document Restructuring Framework, 2006
No 1293 **Jiri Trnka:** Prerequisites for data sharing in emergency management, 2007.
No 1302 **Björn Hägglund:** A Framework for Designing Constraint Stores, 2007.
No 1303 **Daniel Andreasson**: Slack-Time Aware Dynamic Routing Schemes for On-Chip Networks, 2007.
No 1305 **Magnus Ingmarsson:** Modelling User Tasks and Intentions for Service Discovery in Ubiquitous Computing, 2007.
No 1306 **Gustaf Svedjemo**: Ontology as Conceptual Schema when Modelling Historical Maps for Database Storage, 2007.
No 1307 **Gianpaolo Conte**: Navigation Functionalities for an Autonomous UAV Helicopter, 2007.