

Kimon P. Valavanis   Anibal Ollero  
Randal Beard   Leslie Piegl  
Paul Oh   Hyunchul Shim  
*Editors*

# Selected Papers from the 2nd Inter- national Symposium on UAVs, Reno, Nevada, U.S.A., June 8–10, 2009



Springer

Selected papers from the 2nd International  
Symposium on UAVs, Reno, Nevada, U.S.A.  
June 8–10, 2009

Kimon P. Valavanis · Randal Beard · Paul Oh ·  
Aníbal Ollero · Leslie A. Piegl · Hyunchui Shim  
Editors

Selected papers from the  
2nd International Symposium  
on UAVs, Reno, Nevada, U.S.A.  
June 8–10, 2009

Previously published in *Journal of Intelligent  
and Robotic Systems*,  
Volume 57, Issues 1–4 2010



*Editors*

Kimon P. Valavanis  
Department of Electrical  
and Computer Engineering, CMK 300  
School of Engineering  
and Computer Science  
University of Denver  
Denver CO 80208, USA  
Kimon.valavanis@du.edu

Randal Beard  
Brigham Young University  
Dept. Electrical & Computer Engineering  
Provo UT 84602, USA  
beard@ee.byu.edu

Paul Oh  
Drexel University  
Applied Engineering Technology Program  
3001 One Drexel Plaza, Market St.  
Philadelphia PA 19104, USA

Aníbal Ollero  
Universidad de Sevilla  
Escuela Superior de Ingenieros  
Dept. Ingeniera Sistemas y Automática  
Camino de los Descubrimientos,  
s/n 41092 Sevilla, Spain  
aollero@cartuja.us.es

Leslie A. Piegl  
University of South Florida  
Dept. Computer Science & Engineering  
4202 E. Fowler Ave.  
Tampa FL 33620, USA

Hyunchui Shim  
Korea Advanced Institute of Science  
& Technology (KAIST)  
Department of Aerospace Engineering  
373-1 Guseong-dong Daejeon  
305-701 Yuseong-gu  
Republic of South Korea  
hcshim@kaist.ac.kr

ISBN 978-90-481-8763-8  
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2009943828

© Springer Science+Business Media B.V. 2010

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# **Journal of Intelligent and Robotic Systems**

**Volume 57 · Numbers 1–4 · March 2010**

**Special Issue:** Selected papers from the 2nd International Symposium on UAVs, Reno,  
U.S.A. June 8–10, 2009

**Guest Editors:** Kimon P. Valavanis, Randy Beard, Paul Y. Oh, Anibal Ollero, Les A. Piegl,  
Hyunchul Shim

## **EDITORIAL**

### **From the Editor-in-Chief**

K.P. Valavanis 1

### **Accurate Modeling and Robust Hovering Control for a Quad-rotor VTOL Aircraft**

J. Kim · M.-S. Kang · S. Park 9

### **Modeling and System Identification of the muFly Micro Helicopter**

D. Schafroth · C. Berme · S. Bouabdallah · R. Siegwart 27

### **Vision-based Position Control of a Two-rotor VTOL miniUAV**

E. Rondon · S. Salazar · J. Escareno · R. Lozano 49

### **A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance**

C. Goerzen · Z. Kong · B. Mettler 65

### **An Efficient Path Planning and Control Algorithm for RUAV's in Unknown and Cluttered Environments**

K. Yang · S.K. Gan · S. Sukkarieh 101

### **On the Generation of Trajectories for Multiple UAVs in Environments with Obstacles**

A. Alves Neto · D.G. Macharet · M.F.M. Campos 123

### **Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs**

E. Koyuncu · N.K. Ure · G. Inalhan 143

### **A Cost Effective Tracking System for Small Unmanned Aerial Systems**

M. Kontitsis · K. Valavanis 171

### **Vision-Based Road-Following Using Proportional Navigation**

R.S. Holt · R.W. Beard 193

### **A Vision-Based Automatic Landing Method for Fixed-Wing UAVs**

S. Huh · D.H. Shim 217

### **A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks**

A. Cesetti · E. Frontoni · A. Mancini · P. Zingaretti · S. Longhi 233

### **Autonomous Vision-Based Helicopter Flights Through Obstacle Gates**

F. Andert · F.-M. Adolf · L. Goormann · J.S. Dittrich 259

### **Exploring the Effect of Obscurants on Safe Landing Zone Identification**

K.W. Sevcik · N. Kuntz · P.Y. Oh 281

**Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller**

K.E. Wenzel · P. Rosset · A. Zell **297**

**Landing and Perching on Vertical Surfaces with Microspines for Small Unmanned Air Vehicles**

A. Lussier Desbiens · M.R. Cutkosky **313**

**Automating Human Thought Processes for a UAV Forced Landing**

P. Eng · L. Mejias · X. Liu · R. Walker **329**

**Autonomous Autorotation of Unmanned Rotorcraft using Nonlinear Model Predictive Control**

K. Dalamagkidis · K.P. Valavanis · L.A. Piegł **351**

**Multimodal Interface Technologies for UAV Ground Control Stations**

I. Maza · F. Caballero · R. Molina · N. Peña · A. Ollero **371**

**Multi-UAV Simulator Utilizing X-Plane**

R. Garcia · L. Barnes **393**

**UAS Flight Simulation with Hardware-in-the-loop Testing and Vision Generation**

J. Saunders · R. Beard **407**

**Multi-UAV Cooperation and Control for Load Transportation and Deployment**

I. Maza · K. Kondak · M. Bernard · A. Ollero **417**

**Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter**

S. Erhard · K.E. Wenzel · A. Zell **451**

**A Rotary-wing Unmanned Air Vehicle for Aquatic Weed Surveillance and Management**

A.H. Göktoğan · S. Sukkarieh · M. Bryson · J. Randle · T. Lupton · C. Hung **467**

**Development and Evaluation of a Chase View for UAV Operations in Cluttered Environments**

J.T. Hing · K.W. Sevcik · P.Y. Oh **485**

**Design Methodology of a Hybrid Propulsion Driven Electric Powered Miniature Tailsitter Unmanned Aerial Vehicle**

M. Aksugur · G. Inalhan **505**

**Abstracted/Indexed** in Academic OneFile, Compendex, CompuScience, CSA/Proquest, Current Abstracts, Current Contents/Engineering, Computing and Technology, Ergonomics Abstracts, Gale, Google Scholar; Journal Citation Reports/Science Edition, OCLC, PsycINFO, Science Citation Index Expanded (SciSearch), SCOPUS, Summon by Serial Solutions, TOC Premier; VINITY - Russian Academy of Science.

**Instructions for Authors** for *J Intell Robot Syst* are available at <http://www.springer.com/10846>.

## From the Editor-in-Chief

**Kimon P. Valavanis**

Published online: 7 November 2009  
© Springer Science + Business Media B.V. 2009

Dear Colleagues,

I wish you a Happy New Year, full of health, happiness, prosperity, success, and may all your wishes become reality.

I take this opportunity to thank all the past, currently serving and joining members of the Editorial Board for what they have done, for what they are doing and for what they will do to help improve JINT. I express my most sincere gratitude to the Springer team (Nathalie, Anneke, Gabriela, Joey and now Marie Beth) for ‘keeping up’ with us, and to Dina for keeping me ‘under control’.

We start this New Year with high hopes and with new initiatives. As a start, and for the first time, the whole Editorial Board members and our photos are included in this Volume.

From now on we will publish 16 issues per year, one Volume every quarter, with issues 3 and 4 in each Volume being double issues, thus we will still have 12 printings per year. The overwhelming number of submitted papers and all other related activities are forcing us to respond to our readers and accommodate them in the best possible way. In parallel, we will have the flexibility to publish special issues every year! We still aim at publishing the best papers, we are still very selective, we will continue to thrive for excellence.

JINT is improving. This has been the third year in a row where our impact factor is higher than the year before. Even though I have heard arguments that impact factors

---

K. P. Valavanis (✉)  
Journal of Intelligent and Robotic Systems,  
Department of Electrical and Computer Engineering,  
School of Engineering and Computer Science,  
University of Denver, Denver, CO 80208, USA  
e-mail: kvalavan@du.edu

may not reflect reality at all times, I take it as a sign of finding our place under the sun. Somehow, I believe that in the next few years we will have the respect we certainly deserve. We should keep up the good work and work even harder.

The issue at hand is a special issue on Unmanned Aircraft Systems. It includes peer reviewed extended and enhanced versions of papers accepted and presented at the 2<sup>nd</sup> International Symposium on UAVs, which took place in Reno, NV, in June of 2009. The list of papers is included. The areas covered by the papers are:

- UAS Modeling, Control and Identification
- UAS Navigation, Path Planning and Tracking
- UAS Vision and Vision-Based Systems
- Landing and Forced Landing
- Simulation Platforms and Testbeds
- UAS Applications

I hope you enjoy the issue.



Kimon P. Valavanis  
Editor-in-Chief

**JINT EDITORIAL BOARD-EFFECTIVE JANUARY 2010**

**FOUNDING EDITOR**

Dr. Spyros Tzafestas  
National Technical University of Athens, School of ECE  
e-mail: tzafesta@softlab.ntua.gr



**EDITOR-IN-CHIEF**

Dr. Ing. Prof. Kimon P. Valavanis  
University of Denver, CO, USA  
e-mail: Kimon.Valavanis@du.edu



**ASSISTANT TO THE EDITOR-IN-CHIEF**

Ms. Dina Fragkedaiki  
ELKE, Technical University of Crete  
e-mail: fragkedaki@gmail.com



**ADVISORY BOARD**

Dr. Toshio Fukuda  
(*Asia/Oceania*)  
fukuda@mein.nagoya-u.ac.jp



Dr. Gerd Hirzinger  
(*Europe*)  
gerd.hirzinger@dlr.de



Dr. George Vachtsevanos  
(*The Americas*)  
gjv@ece.gatech.edu



**BOOK REVIEW EDITOR**

Dr. Yannis Phllis  
Technical University of Crete  
e-mail: phllis@dpem.tuc.gr



**EDITORS-AT-LARGE**

**The Americas**

Dr. Nikos Papanikopoulos  
University of Minnesota  
e-mail: npapas@cs.umn.edu



**Europe**

Dr. Roland Siegwart  
ETH Zurich  
e-mail: rsiegwart@ethz.ch



**Asia/Oceania**

Dr. Shuzhi Sam Ge  
The National University of Singapore  
e-mail: elegesz@nus.edu.sg



**SENIOR EDITORS**

Dr. Krzysztof Kozlowski  
Poznan University of Technology  
e-mail: krzysztof.kozlowski@put.poznan.pl



Dr. Vassilis Kouikoglou  
Technical University of Crete  
e-mail: kouik@dpem.tuc.gr



Dr. Les A. Piegl  
University of South Florida  
e-mail: lap@cse.usf.edu



**Dr. Sauro Longhi**  
Università Politecnica  
delle Marche  
e-mail: sauro.longhi@univpm.it



**Dr. Alfredo Weitzenfeld**  
Instituto Tecnologico Autonomo  
de Mexico  
e-mail: alfredo@itam.mx



**Dr. Bum-Jae You**  
Korean Institute for Science  
and Technology  
e-mail: ybj@kist.re.kr



**Dr. Bernard Mettler**  
University of Minnesota  
e-mail: mettler@aem.umn.edu



**Dr. Paul Oh**  
Drexel University  
e-mail: paul@coe.drexel.edu



**Dr. Zoran Vukic**  
University of Zagreb  
e-mail: zoran.vukic@fer.hr



**Dominique Sauter**  
Nancy-University, France  
e-mail: dominique.sauter@cran.uhp-nancy.fr



## EDITORS

**Dr. Beno Benhabib**  
University of Toronto  
e-mail: benhabi@mie.utoronto.ca



**Dr. Stjepan Bogdan**  
University of Zagreb  
e-mail: stjepan.bogdan@fer.hr



**Dr. Sesh Commuri**  
University of Oklahoma  
e-mail: scommuri@ou.edu



Dr. Aydan M. Erkmen  
Middle East Technical University  
e-mail: aydan@metu.edu.tr



Dr. Rafael Fierro  
University of New Mexico  
e-mail: rfierro@ece.unm.edu



Dr. Hui Jun Gao  
Harbin Institute of Technology  
e-mail: hjgao@hit.edu.cn



Dr. Fumiya Iida  
CS & AI, MIT  
e-mail: iida@csail.mit.edu



Dr. Sanjay Joshi  
UC Davis  
e-mail: maejoshi@ucdavis.edu



Dr. Abderrahmane Kheddar  
France  
e-mail: kheddar@iup.univ-evry.fr, abderrahmane.kheddar@aist.go.jp



Dr. Derek Kingston  
WP Air Force Base  
Derek.Kingston@WPAFB.  
AF.MIL

Dr. Fumitoshi Matsuno  
Kyoto University  
e-mail: matsuno@me.kyoto-u.ac.jp



Dr. Giovanni Muscato  
Università degli Studi  
di CATANIA  
e-mail: gmuscato@dees.unict.it



Dr. Dan Popa  
ARRI, UT Arlington  
e-mail: popa@arri.uta.edu



Dr. Nilanjan Sarkar  
Vanderbilt University  
e-mail: nilanjan.sarkar@vanderbilt.edu



Dr. Javier Ruiz-del-Solar  
Universidad de Chile  
e-mail: jruizd@cec.uchile.cl



Dr. Yuan Yan Tang  
Hong Kong Baptist University  
e-mail: yytang@comp.hkbu.edu.hk



Dr. Costas Tzafestas  
National Technical University of Athens  
e-mail: ktzaf@softlab.ece.ntua.gr



Dr. Miklos Vajta  
University of Twente  
e-mail: m.vajta@math.utwente.nl



Dr. Ljubo Vlacic  
Griffith University  
e-mail: l.vlacic@griffith.edu.au



Dr. Long Wang  
Peking University  
e-mail: longwang@pku.edu.cn



Dr. Jing Xiao  
U of North Carolina — Charlotte  
e-mail: xiao@uncc.edu



Dr. Nikos Tsourveloudis  
Technical University of Crete  
e-mail: nikost@dDEM.tuc.gr



Dr. Ali Goktogan  
Australian Centre for Field Robotics, USYD  
e-mail: a.goktogan@cas.edu.au



Dr. Ahmad Rad  
Simon Fraser University  
e-mail: arad@sfu.ca



## Accurate Modeling and Robust Hovering Control for a Quad-rotor VTOL Aircraft

Jinhyun Kim · Min-Sung Kang · Sangdeok Park

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 17 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** Quad-robot type (QRT) unmanned aerial vehicles (UAVs) have been developed for quick detection and observation of the circumstances under calamity environment such as indoor fire spots. The UAV is equipped with four propellers driven by each electric motor, an embedded controller, an Inertial Navigation System (INS) using three rate gyros and accelerometers, a CCD (Charge Coupled Device) camera with wireless communication transmitter for observation, and an ultrasonic range sensor for height control. Accurate modeling and robust flight control of QRT UAVs are mainly discussed in this work. Rigorous dynamic model of a QRT UAV is obtained both in the reference and body frame coordinate systems. A disturbance observer (DOB) based controller using the derived dynamic models is also proposed for robust hovering control. The control input induced by DOB is helpful to use simple equations of motion satisfying accurately derived dynamics. The developed hovering robot shows stable flying performances under the adoption of DOB and the vision based localization method. Although a model is incorrect, DOB method can design a controller by regarding the inaccurate part of the model

---

J. Kim

Department of Mechanical Engineering, Seoul National University of Technology,  
Seoul, South Korea  
e-mail: jinhyun@snut.ac.kr

M.-S. Kang

Department of Mechatronics Engineering, Hanyang University,  
Ansan, South Korea  
e-mail: wowmecha@gmail.com

S. Park (✉)

Division of Applied Robot Technology, Korea Institute of Industrial Technology,  
Ansan, South Korea  
e-mail: sspark@kitech.re.kr

and sensor noises as disturbances. The UAV can also avoid obstacles using eight IR (Infrared) and four ultrasonic range sensors. This kind of micro UAV can be widely used in various calamity observation fields without danger of human beings under harmful environment. The experimental results show the performance of the proposed control algorithm.

**Keywords** Hovering control · QRT (quad-rotor type) UAV (unmanned aerial vehicle) · VTOL (vertical take-off and landing)

## 1 Introduction

Recently, researches on Unmanned Aerial Vehicles (UAVs) have been vigorously being performed for calamity observation such as woods and building fires, meteorological observation, patrol, and spraying agricultural chemicals to military purpose such as reconnaissance, monitoring, and communication etc. Moreover, the technology development of the UAVs is getting faster according to the rapid progress of electronic and computer technology. Micro UAVs can be operated on wide area regardless of the effect of ground configuration. The merit of UAVs is maximized for the practical use in the places where it is dangerous and difficult to approach. Further, micro UAVs are much cheaper and safer in dangerous tasks than piloted aircrafts.

Micro UAVs are classified into two categories, fixed and rotary wing types. The rotary wing type UAVs are more advantageous than the fixed wing type ones in the sense of VTOL(Vertical Take-off and Landing), omni-directional flying, and hovering performances. Rotary wing type UAVs are classified into quad-rotor type (QRT), co-axial helicopter, and helicopter etc. QRT UAVs have the simplest mechanical structure among them, and that is the reason why a QRT UAV is considered in this study as a flying platform.

QRT UAVs with various shape and size have been developed for commercial and research purposes, and some models are being sold as RC toys. Autonomous QRT UAVs equipped with high technology sensors are also being developed for special purposes, and researches for analysis and control of the flying robots are being done vigorously.

V. Mistler et al. derived the dynamic model for a four rotors helicopters, and developed a dynamic feedback controller [1]. P. McKerrow obtained the dynamic model for theoretical analysis of a dragonflyer [2]. A. Mokhtari and A. Benallegue obtained a nonlinear dynamic model of a QRT UAV for state variable control based on Euler angle and an open loop position state observer, emphasized attitude control rather than the translational motion of the UAV [3]. P. Castillo et al. performed autonomous take-off and hovering and landing control of a QRT UAV by synthesizing a controller using the Lagrangien model based on the Lyapunov analysis [4]. S. Bouabdallah et al. performed the design, dynamic modelling, sensing and control of an indoor micro QRT UAV [5]. S. Bouabdallah and R. Siegwart suggested two nonlinear control techniques: a backstepping and a sliding-mode control [6].

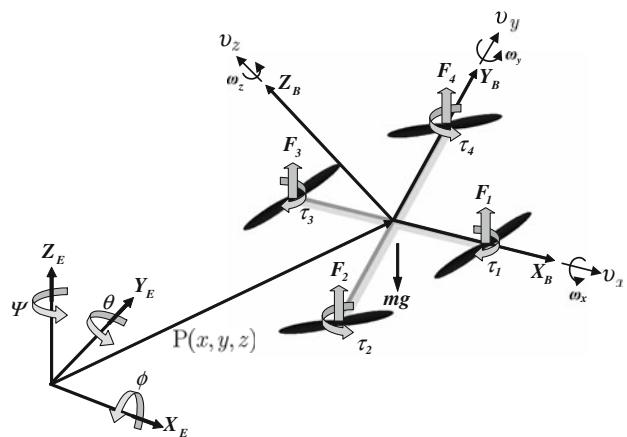
Flight control methods utilizing vision systems are also studied. E. Altuğ et al. proposed a visual feedback control method of a QRT UAV using a camera equipped on the UAV as the main sensor for attitude estimation [7]. T. Hamel et al. proposed a vision based visual servo controller for performing trajectory tracking tasks of under-actuated systems like QRT UAVs [8]. T. Hamel and R. Mahony proposed a vision based controller which performs visual servo control by positioning a camera onto a fixed target for the hovering of a QRT UAV [9].

Strictly speaking, earlier dynamic models are lack of mathematical rigorousness. They have dynamic inconsistency as they are expressed using both the reference and body coordinate systems without proper transformation [3, 6, 10–15]. This discrepancy mainly comes from the classical aerodynamic description based on the reference frame which is convenient in the case of using mechanical gyros, while the strip-down type gyros which are widely used in small QRT UAVs due to their compactness are much closer to describe the dynamic model using body frame coordinate system. Therefore, it is required to derive rigorous dynamic models of QRT UAVs both in the reference and body frame coordinate systems.

In the previous research, the linear and angular equations of motion is expressed in the reference frame and in the body frame coordinate, respectively. In some cases, they induced incorrect dynamic equations without consideration of the difference between two coordinates. [3, 6, 10–15]. In addition, for the controller, the angular equations of motion derived in the body frame coordinate is misused as the equations of motion derived in the reference frame without adequate assumptions.

In this paper, the above mentioned two problems will be solved. For the moving base system which is not fixed in an inertial frame, it is not convenient to derive the dynamic formulation using the Lagrangian in terms of the velocities expressed in a body-fixed frame. In UAV systems, the sensor information and actuator forces are exerted on a body-fixed frame, so it is more natural to write up the dynamics using body-fixed velocities. To supplement this, we used quasi-Lagrange approach [16]. The quasi-Lagrange method can give the equations of motion in terms of the body-fixed velocities. Finally the composite equations of motion for a QRT UAV will be derived strictly. In addition, using rigorous assumption—that is the roll and pitch angles are remain near zeros—the angular equations of motion derived in the body frame is induced the equations of motion in the reference frame. To guarantee the assumption, the robust controller based on disturbance observer(DOB) [17] is proposed. DOB is a kind of internal loop compensator which enforces desired nominal model on the original complex dynamic equations. The control input induced by DOB will be helpful to use simple equations of motion as mentioned above satisfying the accurate derived dynamics. In this work, rigorous dynamic models of a QRT UAV are obtained both in the reference and body frame coordinate systems. A disturbance observer(DOB) based controller using the derived dynamic models is also proposed for hovering control. And a vision based localization for the developed QRT UAV are introduced. The performance of the proposed control scheme is verified through experiments using QRT UAVs.

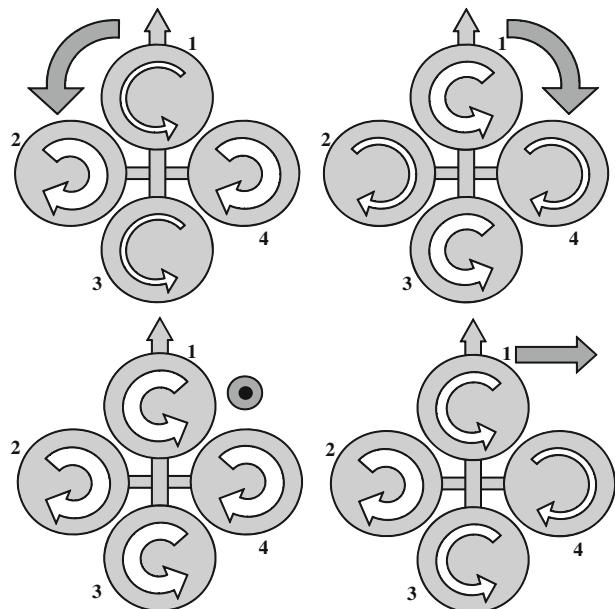
Section 2 shows the basic structure and the dynamics of the UAV. The DOB for flight control algorithms is given in Section 3. Section 4 shows the experimental setup and experimental results including the vision based localization schemes, and followed by concluding remarks in Section 5.

**Fig. 1** Coordinate system

## 2 Dynamic Equations of Motion

Figure 1 shows coordinate systems of QRT UAVs. The motion of QRT UAVs is induced by the combination of quad rotors as shown in Fig. 2. The state vectors for the QRT UAVs are described as following:

$$\begin{aligned}\eta &= [\eta_1^T, \eta_2^T]^T; \quad \eta_1 = [x, y, z]^T; \quad \eta_2 = [\phi, \theta, \psi]^T; \\ \mathbf{v} &= [\mathbf{v}_1^T, \mathbf{v}_2^T]^T; \quad \mathbf{v}_1 = [v_x, v_y, v_z]^T; \quad \mathbf{v}_2 = [\omega_x, \omega_y, \omega_z]^T;\end{aligned}$$

**Fig. 2** QRT motion principle

where the position and orientation of a QRT UAV,  $\eta$ , are described relative to the inertial reference frame, while the linear and angular velocities of a QRT UAVs,  $v$ , are expressed in the body-fixed frame.  $x$ ,  $y$ , and  $z$  mean the linear positions of a QRT UAV with respect to inertial reference frame.  $\phi$ ,  $\theta$ , and  $\psi$  represent the roll, pitch, and yaw angles of the QRT UAV in the inertial reference frame.

The inertial and body-fixed velocity relation can be represented with a QRT UAV Jacobian matrix [18].

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1(\eta_2) & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2(\eta_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \iff \dot{\eta} = \mathbf{J}(\eta)v, \quad (1)$$

where

$$\mathbf{J}_1(\eta_2) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \quad (2)$$

and

$$\mathbf{J}_2(\eta_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}. \quad (3)$$

In above equations,  $c(\cdot)$  and  $s(\cdot)$  denote  $\cos(\cdot)$  and  $\sin(\cdot)$ , respectively. Notice that  $\mathbf{J}_2(\eta_2)$  is undefined for a pitch angle of  $\theta = \pm 90^\circ$ . However, in the QRT UAVs, the system goes into uncontrollable states as approaching the pitch angle. Hence, in this paper, we do not consider the pitch angle. In addition,  $\mathbf{J}_2(\eta_2)$  is not orthogonal matrix. Consequently,  $\mathbf{J}_2^{-1}(\eta_2) \neq \mathbf{J}_2^T(\eta_2)$ .

For the moving base system which is not fixed in an inertial frame, it is not convenient to derive the dynamic formulation using the Lagrangian in terms of the velocities expressed in a body-fixed frame. In QRT UAVs, the sensor information and actuator forces are exerted on a body-fixed frame, so it is more natural to write up the dynamics using body-fixed velocities. To supplement this, we used quasi-Lagrange approach [16]. The quasi-Lagrange method can give the equations of motion in terms of the body-fixed velocities.

## 2.1 Quasi-Lagrange Equations of Motion

The Lagrangian in general form is defined as

$$L = T - V, \quad (4)$$

$$T = \frac{1}{2} \mathbf{v}^T \mathbf{M} \mathbf{v} = \frac{1}{2} m \mathbf{v}_1^T \mathbf{v}_1 + \frac{1}{2} \mathbf{v}_2^T \mathbf{I} \mathbf{v}_2, \quad (5)$$

$$V = -mgz, \quad (6)$$

where  $m$  and  $\mathbf{I}$  are mass and system inertia, respectively. Normally, we design the QRT UAV body as axisymmetric, so the inertia is defined as Eq. 7, and especially  $I_{xx} = I_{yy}$ .

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (7)$$

With the extended Hamiltonian principle, Eq. 4 gives following differential equations:

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \mathbf{v}_1} \right) + \mathbf{v}_2 \times \frac{\partial L}{\partial \mathbf{v}_1} - \mathbf{J}_1^T \frac{\partial L}{\partial \boldsymbol{\eta}_1} = \boldsymbol{\tau}_1 \quad (8)$$

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \mathbf{v}_2} \right) + \mathbf{v}_2 \times \frac{\partial L}{\partial \mathbf{v}_2} + \mathbf{v}_1 \times \frac{\partial L}{\partial \mathbf{v}_1} - \mathbf{J}_2^T \frac{\partial L}{\partial \boldsymbol{\eta}_2} = \boldsymbol{\tau}_2 \quad (9)$$

Then,

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}\mathbf{v} + \mathbf{g} = \boldsymbol{\tau}. \quad (10)$$

Finally, the 6 independent equations of motions are obtained as following:

$$m[\dot{v}_x - v_y \omega_z + v_z \omega_y - gs\theta] = 0, \quad (11)$$

$$m[\dot{v}_y - v_z \omega_x + v_x \omega_z + gc\theta s\phi] = 0, \quad (12)$$

$$m[\dot{v}_z - v_x \omega_y + v_y \omega_x + gc\theta c\phi] = u_1, \quad (13)$$

$$I_{xx}\dot{\omega}_x + (I_{zz} - I_{yy})\omega_y\omega_z = u_2, \quad (14)$$

$$I_{yy}\dot{\omega}_y + (I_{xx} - I_{zz})\omega_z\omega_x = u_3, \quad (15)$$

$$I_{zz}\dot{\omega}_z = u_4, \quad (16)$$

where

$$\boldsymbol{\tau} = \begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ F_1 + F_2 + F_3 + F_4 & (-F_2 + F_4)l & (-F_1 + F_3)l \\ (-F_2 + F_4)l & (-F_1 + F_3)l & (-F_1 + F_2 - F_3 + F_4)\lambda \end{bmatrix}, \quad (17)$$

and  $l$  is the distance between the motor and the center of mass.

Equation 17 can be rewritten as the form of matrix like below:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ -\lambda & \lambda & -\lambda & \lambda \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \mathbf{T} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}. \quad (18)$$

## 2.2 Earth-Fixed Vector Representation

Until now, we derive the dynamic equation of motion related to the body fixed coordinate frame. However, for the control purpose, it is more convenient to use the dynamic equations derived in earth-fixed coordinate frame like below:

$$\mathbf{M}_{\eta}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \mathbf{C}_{\eta}(\mathbf{v}, \boldsymbol{\eta})\dot{\boldsymbol{\eta}} + \mathbf{g}_{\eta}(\boldsymbol{\eta}) = \boldsymbol{\tau}_{\eta}(\boldsymbol{\eta}). \quad (19)$$

To express the dynamic equations in earth-fixed coordinate frame like Eq. 19, we need following relationship

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{J}(\boldsymbol{\eta})\mathbf{v} &\iff \mathbf{v} &= \mathbf{J}^{-1}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} \\ \ddot{\boldsymbol{\eta}} &= \mathbf{J}(\boldsymbol{\eta})\dot{\mathbf{v}} + \dot{\mathbf{J}}(\boldsymbol{\eta})\mathbf{v} \iff \dot{\mathbf{v}} &= \mathbf{J}^{-1}(\boldsymbol{\eta})[\ddot{\boldsymbol{\eta}} - \dot{\mathbf{J}}(\boldsymbol{\eta})\mathbf{v}] \end{aligned} \quad (20)$$

Then, the system matrices are defined as below:

$$\begin{aligned}\mathbf{M}_\eta(\eta) &= \mathbf{J}^{-T}(\eta)\mathbf{M}\mathbf{J}^{-1}(\eta) \\ \mathbf{C}_\eta(v, \eta) &= \frac{1}{2}\dot{\mathbf{M}}_\eta(\eta) \\ \mathbf{g}_\eta(\eta) &= \mathbf{J}^{-T}(\eta)\mathbf{g}(\eta) \\ \boldsymbol{\tau}_\eta(\eta) &= \mathbf{J}^{-T}(\eta)\boldsymbol{\tau}\end{aligned}\quad (21)$$

Finally, we can derive the equations of motion in earth-fixed coordinate frame.

$$m\ddot{x} = (s\psi s\phi + c\psi c\phi s\theta)u_1 \quad (22)$$

$$m\ddot{y} = (-c\psi s\phi + s\theta s\psi c\phi)u_1 \quad (23)$$

$$m(\ddot{z} + g) = c\theta c\phi u_1 \quad (24)$$

$$\mathbf{M}_{\eta_2}\ddot{\eta}_2 + \frac{1}{2}\dot{\mathbf{M}}_{\eta_2}\dot{\eta}_2 = \begin{bmatrix} u_2 \\ u_3 c\phi - u_4 s\phi \\ -u_2 s\theta + u_3 c\theta s\phi + u_4 c\theta c\phi \end{bmatrix} \quad (25)$$

where

$$\mathbf{M}_{\eta_2} = \begin{bmatrix} I_{xx} & 0 & -I_{xx}s\theta \\ 0 & I_{yy}c^2\phi + I_{zz}s^2\phi & (I_{yy} - I_{zz})c\phi c\theta s\phi \\ -I_{xx}s\theta & (I_{yy} - I_{zz})c\phi c\theta s\phi & I_{xx}s^2\theta + I_{yy}c^2\theta s^2\phi + I_z c^2\theta c^2\phi \end{bmatrix}. \quad (26)$$

### 2.3 Equations of Motion of a QRT UAV

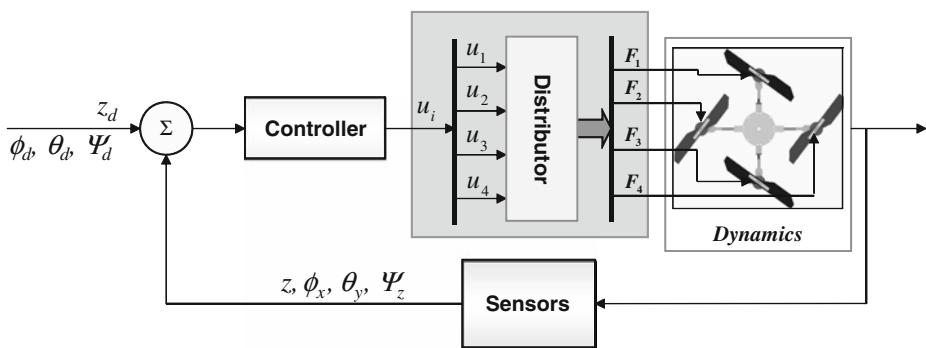
As shown in previous subsections, the linear equations of motion of a QRT UAV are simple in earth-fixed reference frame, while the angular equations are advantageous to express in body-fixed reference frame. According to the above analysis, finally, the following equations are derived.

$$\begin{cases} m\ddot{x} = (s\psi s\phi + c\psi c\phi s\theta)u_1 \\ m\ddot{y} = (-c\psi s\phi + s\theta s\psi c\phi)u_1 \\ m(\ddot{z} + g) = c\theta c\phi u_1 \\ I_{xx}\dot{\omega}_x + (I_{zz} - I_{yy})\omega_y\omega_z = u_2, \\ I_{yy}\dot{\omega}_y + (I_{xx} - I_{zz})\omega_z\omega_x = u_3, \\ I_{zz}\dot{\omega}_z = u_4. \end{cases} \quad (27)$$

As shown in the Eq. 27, for the linear motions, all the states are subordinated to the control parameter  $u_1$ , hence only one state is controllable and the others are subjected to the controlled linear motion and angular motions. In this paper, for the hovering control, we only consider and control the  $z$ -directional linear motions.

Especially, the hovering control with  $\phi \approx 0$  and  $\theta \approx 0$  can make the dynamics much simpler form like Eq. 28, and it is easy to design the controller.

$$\begin{aligned}m(\ddot{z} + g) &= u_1 \\ I_{xx}\ddot{\phi} &= u_2 - (I_{zz} - I_{yy})\dot{\theta}\dot{\psi}, \\ I_{yy}\ddot{\theta} &= u_3 - (I_{xx} - I_{zz})\dot{\psi}\dot{\phi}, \\ I_{zz}\ddot{\psi} &= u_4.\end{aligned}\quad (28)$$



**Fig. 3** Basic structure of the controller for the UAV

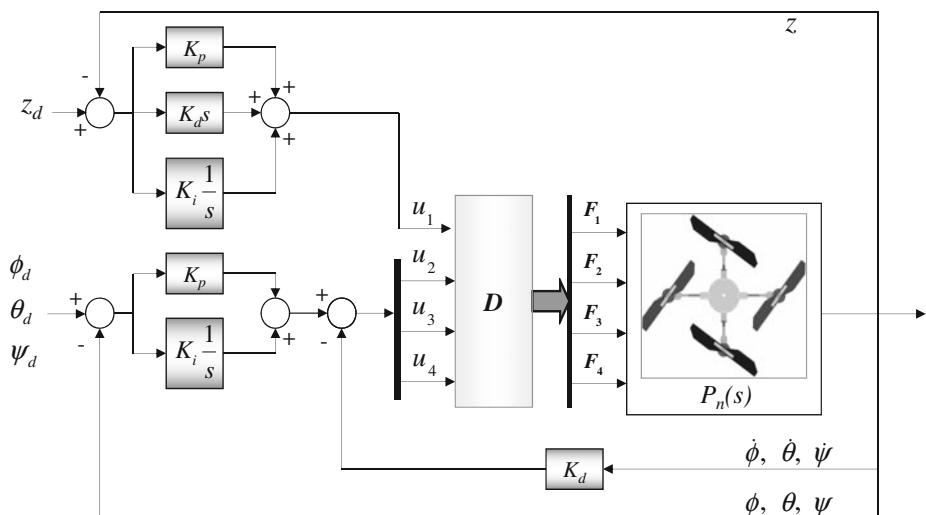
To satisfy the conditions where Eq. 28 is valid,  $\phi$  and  $\theta$  have to stay near the neutral positions, which are zeros. So, for the following section, the robust controller based on the disturbance observer for a QRT UAV is proposed.

### 3 Controller Design

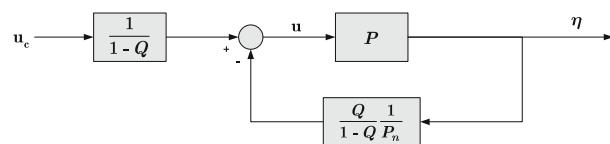
Figure 3 shows the basic structure for the flight control of the UAV.

Now, let us consider following composite dynamic equations of motion.

$$\begin{bmatrix} m(\ddot{z} + g) \\ I_{xx}\ddot{\phi} \\ I_{yy}\ddot{\theta} \\ I_{zz}\ddot{\psi} \end{bmatrix} + \Delta = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \quad (29)$$

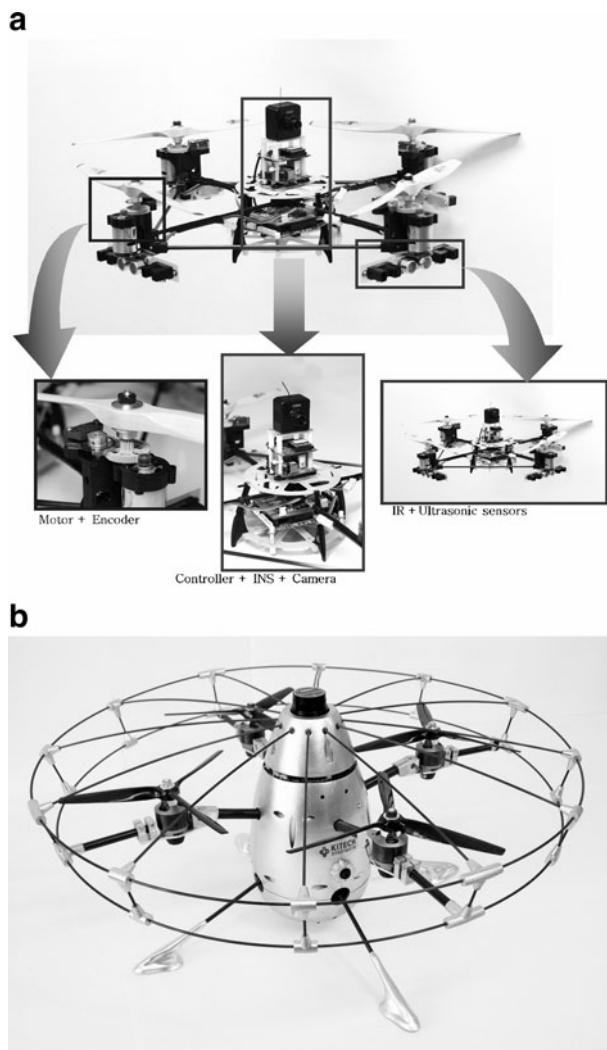


**Fig. 4** PID controller for controlling the altitude and attitude of a QRT UAV

**Fig. 5** Disturbance observer

where the disturbance,  $\Delta$ , is defined as

$$\Delta = \begin{bmatrix} \delta_3 \\ \delta_4 + (I_{zz} - I_{yy})\omega_y\omega_z \\ \delta_5 + (I_{xx} - I_{zz})\omega_z\omega_x \\ \delta_6 \end{bmatrix}. \quad (30)$$

**Fig. 6** The developed QRT Unmanned aerial vehicles.: **a** version 1; and, **b** version 2

**Table 1** The specification and parameters of the QRT UAV version 2

Description	Value
Weight	2.2 kg
Diameter	573.6 mm
Height	170 mm
Distance between the motor and the C.G	172.5 mm
Propeller	9"×7"
Moment of inertia in x-axis, $I_{xx}$	16.76337 gm <sup>2</sup>
Moment of inertia in y-axis, $I_{yy}$	16.76337 gm <sup>2</sup>
Moment of inertia in z-axis, $I_{zz}$	23.1447 gm <sup>2</sup>

And  $\delta_i$  mainly comes from the dynamic inconsistency. For the hovering control, the amount of disturbance is relatively small. Hence, it does not give instability, but poor performance, which may violate the assumption,  $\phi \approx 0$  and  $\theta \approx 0$ . This phenomenon can be resolved using DOB control input.

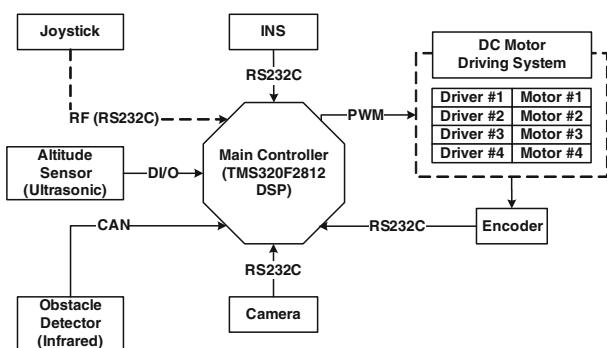
### 3.1 Basic Controller

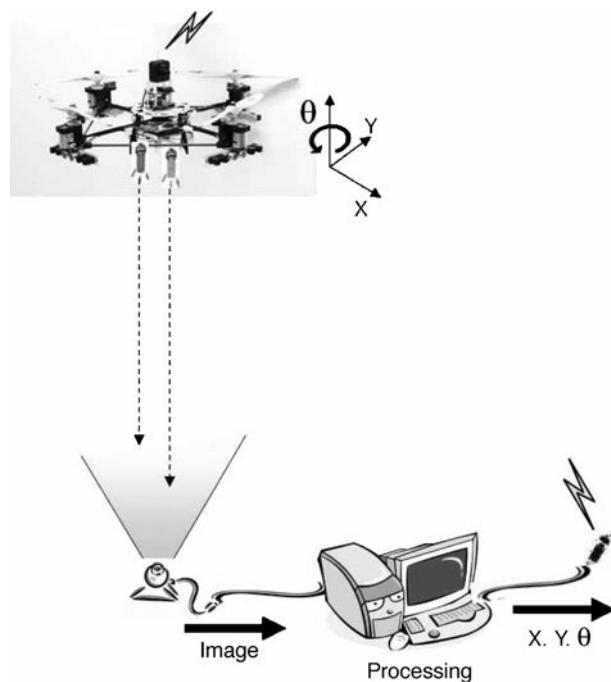
The control algorithms for the altitude and attitude, roll, pitch and yaw, of the UAV are designed based on PID controllers as shown in Fig. 4. The control input  $u_1$  for controlling the altitude  $z$  of the UAV with respect to the reference input  $z_d$  is designed as

$$u_1 = K_{p1}(z_d - z) + K_{d1} \frac{d(z_d - z)}{dt} + K_{I1} \int_0^t (z_d - z) d\tau. \quad (31)$$

The control inputs  $u_j$  ( $j = 2, 3, 4$ ) for controlling the attitude  $(\phi, \theta, \psi)$  of the UAV with respect to the reference inputs  $(\phi, \theta, \psi)_d$  are designed, respectively given as:

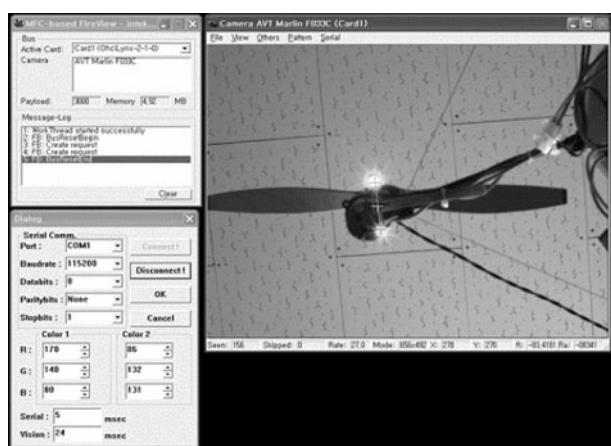
$$u_j = K_{pj}[(\phi, \theta, \psi)_d - (\phi, \theta, \psi)] - K_{dj}(\dot{\phi}, \dot{\theta}, \dot{\psi}) + K_{Ij} \int_0^t [(\phi, \theta, \psi)_d - (\phi, \theta, \psi)] d\tau - \bar{u}_j. \quad (32)$$

**Fig. 7** Schematic view of the embedded controller

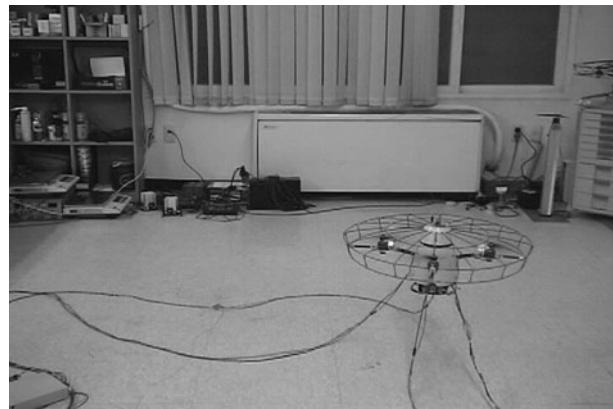
**Fig. 8** The scheme of the vision based localization

$\bar{u}_j$  are the PD control inputs for avoiding obstacles in roll and pitch axis given as:

$$\begin{aligned}\bar{u}_2 &= [K_{pr}(d_{rd} - d_r) - K_{dr}\dot{d}_r]U(d_{rd} - d_r) - [K_{pl}(d_{ld} - d_l) - K_{dl}\dot{d}_l]U(d_{ld} - d_l) \\ \bar{u}_3 &= [K_{pb}(d_{bd} - d_b) - K_{db}\dot{d}_b]U(d_{bd} - d_b) - [K_{pf}(d_{fd} - d_f) - K_{df}\dot{d}_f]U(d_{fd} - d_f) \\ \bar{u}_4 &= 0\end{aligned}\quad (33)$$

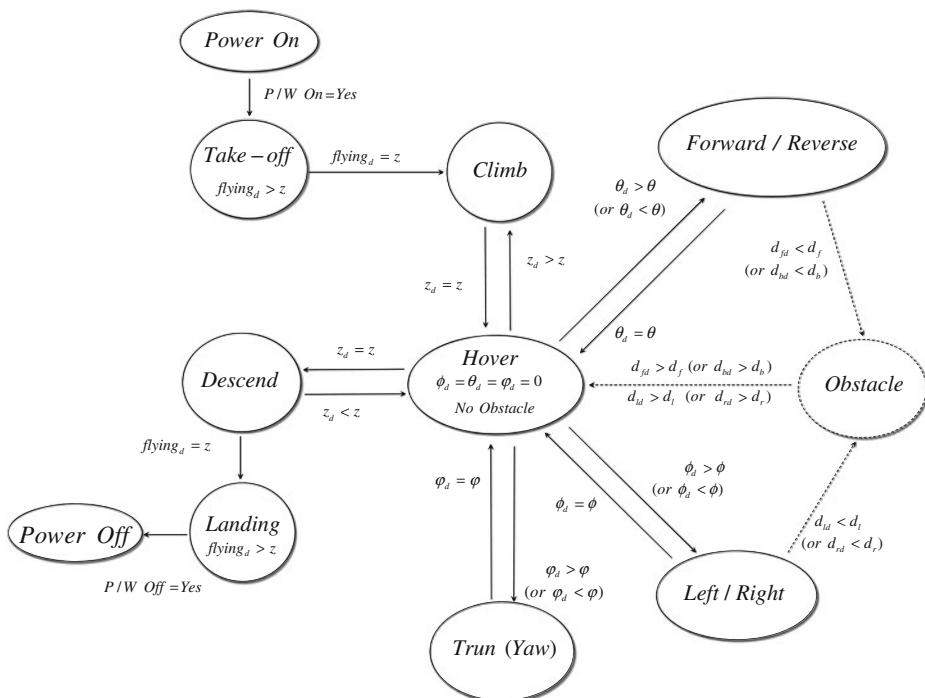
**Fig. 9** The result of the vision based localization

**Fig. 10** Experiments of the QRT Unmanned aerial vehicle

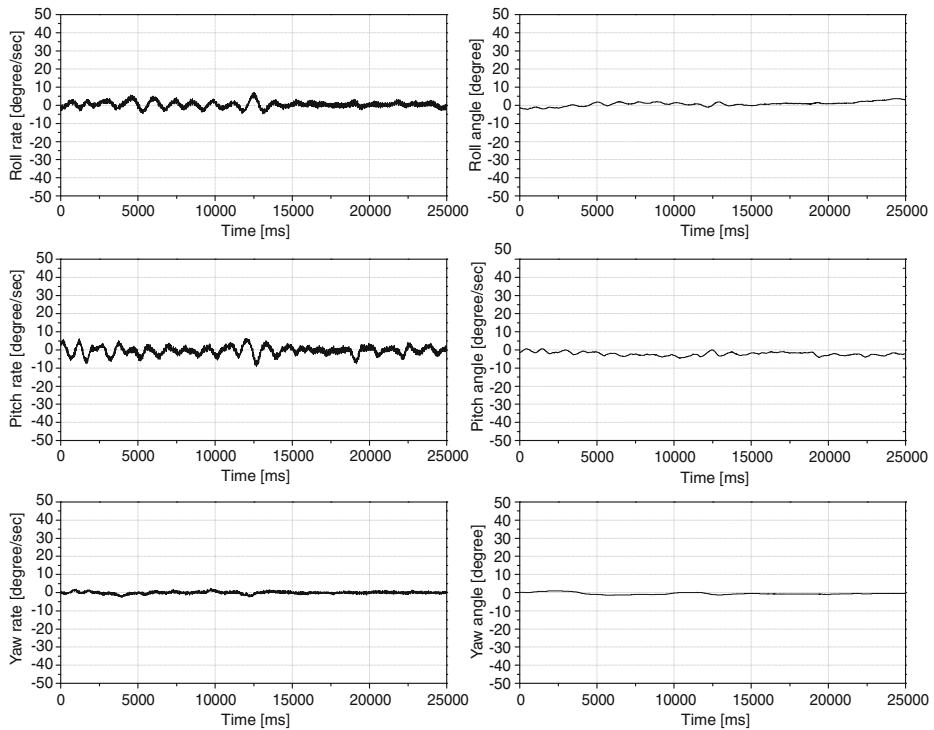


where  $d_{md}$  ( $m = f, b, l, r$ ) are reference distances for avoiding obstacles to forward, backward, left, and right direction, and  $d_m$  ( $m = f, b, l, r$ ) are distances of obstacles to the four direction.  $U(d_{md} - d_m)$  ( $m = f, b, l, r$ ) are step functions as follows:

$$U(d_{md} - d_m) = \begin{cases} 1, & \text{for } d_{md} < d_m \\ 0, & \text{for } d_{md} > d_m \end{cases} \quad (m = f, b, l, r) \quad (34)$$



**Fig. 11** The control flow of flying robot system



**Fig. 12** Experimental results for hovering and attitude control of the developed UAV: hovering of the UAV

### 3.2 Disturbance Observer Based Disturbance Compensation

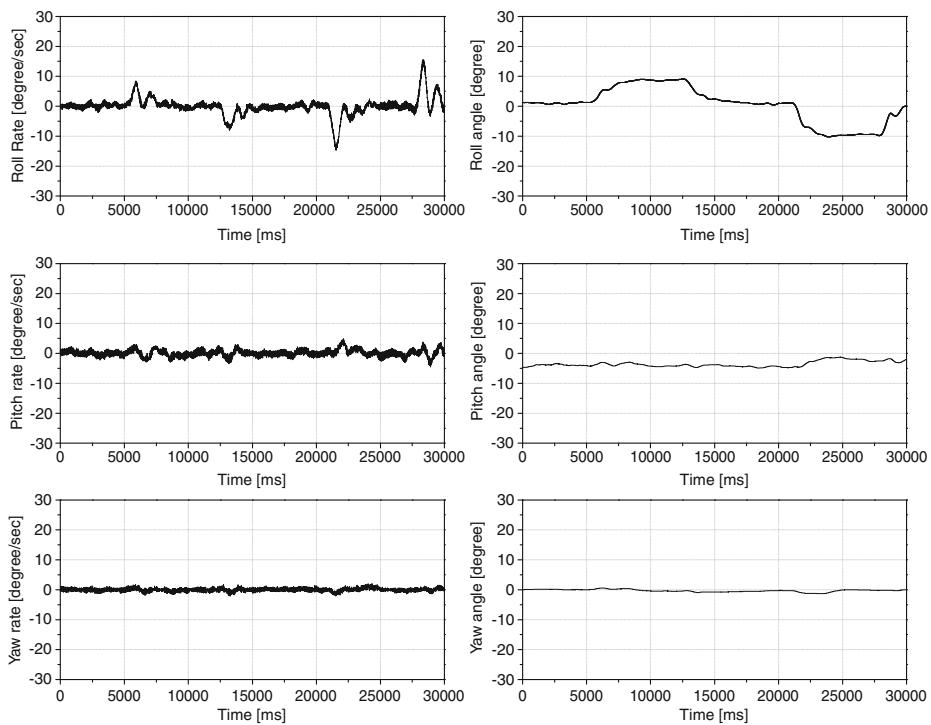
The control algorithms Eqs. 31 and 32 control the altitude and attitude of the UAV using sensor signals. The flight performance of the UAV, however, is not good in real flight because of the sensor noises and disturbances. The DOB based control algorithm, as shown in Fig. 5, controls the response of the plant  $P(s)$  to follow that of the model plant  $P_n(s)$  even though disturbances, sensor noise, and modeling uncertainty are applied to the plant.

Figure 5 shows the block diagram of disturbance observer. It can be expressed as

$$u(s) = \frac{1}{1-Q} u_c(s) - \frac{Q}{1-Q} \frac{1}{P_n} \eta(s). \quad (35)$$

DOB based disturbance compensator can be used to both for altitude and attitude control in the same way. In the cases the nominal models are given in the form of force-mass and inertia-moment dynamic models without friction, respectively given as  $P_n = \frac{1}{ms^2}$  for the linear motion, and  $P_n = \frac{1}{I_{xx,yy,zz}s^2}$  for the angular motion. And, for the  $Q$ -filter, we use well-known  $Q_{31}$  filter as following:

$$Q_{31}(s) = \frac{1 + 3\tau s}{(1 + \tau s)^3}, \quad (36)$$



**Fig. 13** Experimental results for hovering and attitude control of the developed UAV: desired roll angle of the UAV =  $\pm 10^\circ$

Then, the internal loop will be conducted as

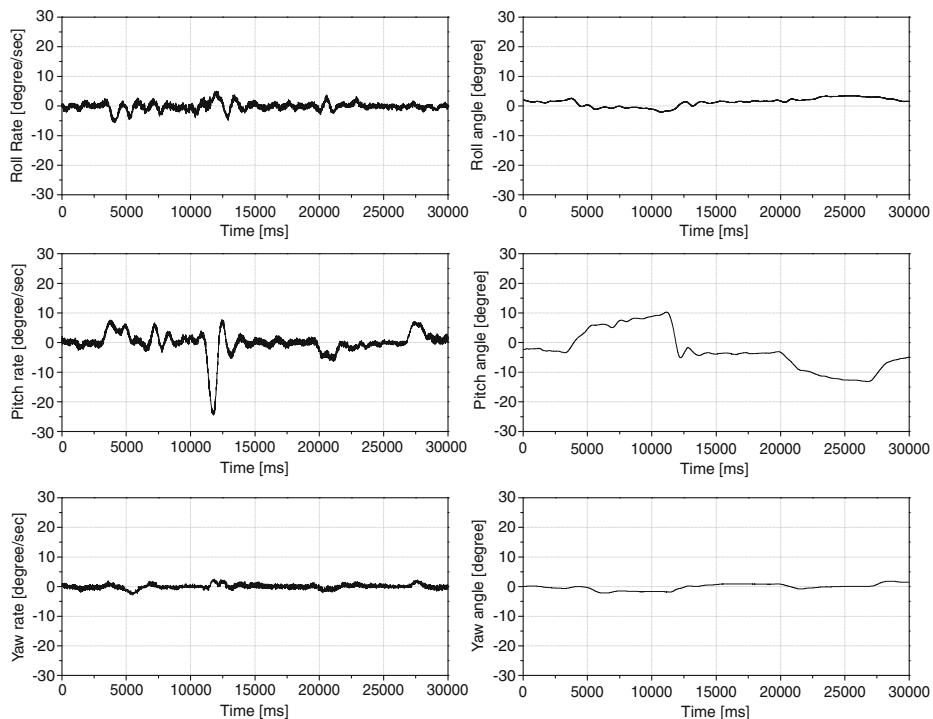
$$\begin{bmatrix} m(\ddot{z} + g) \\ I_x \ddot{\phi} \\ I_y \ddot{\theta} \\ I_z \ddot{\psi} \end{bmatrix} = \begin{bmatrix} u_{c1} \\ u_{c2} \\ u_{c3} \\ u_{c4} \end{bmatrix}. \quad (37)$$

Using Eq. 37, we can easily design the linear controller for each axis independently.

## 4 Experimental Results

### 4.1 Experimental Setup

Figure 6 shows the fabricated QRT UAVs for the purpose of calamity observation in indoor environment. The UAVs have 4 rigid blades driven by each BLDC motor mounted at each end of a crossing body frame. The Encoders for measuring speed are mounted on the motors. The UAVs are also equipped with an IMU( Inertial Measurement Unit) composed of three rate gyros for sensing attitude, eight IRs and four ultrasonic range sensors for obstacle detection, and one additional ultrasonic sensor for altitude measuring. A CCD camera with wireless RF(Radio Frequency) transmitter is mounted on the top of the UAV for the observation task. Table 1 gives



**Fig. 14** Experimental results for hovering and attitude control of the developed UAV: desired pitch angle of the UAV =  $\pm 10^\circ$

the specification of the developed UAV. Figure 7 shows the schematic view of the embedded controller using a TMS320F2812 DSP controller.

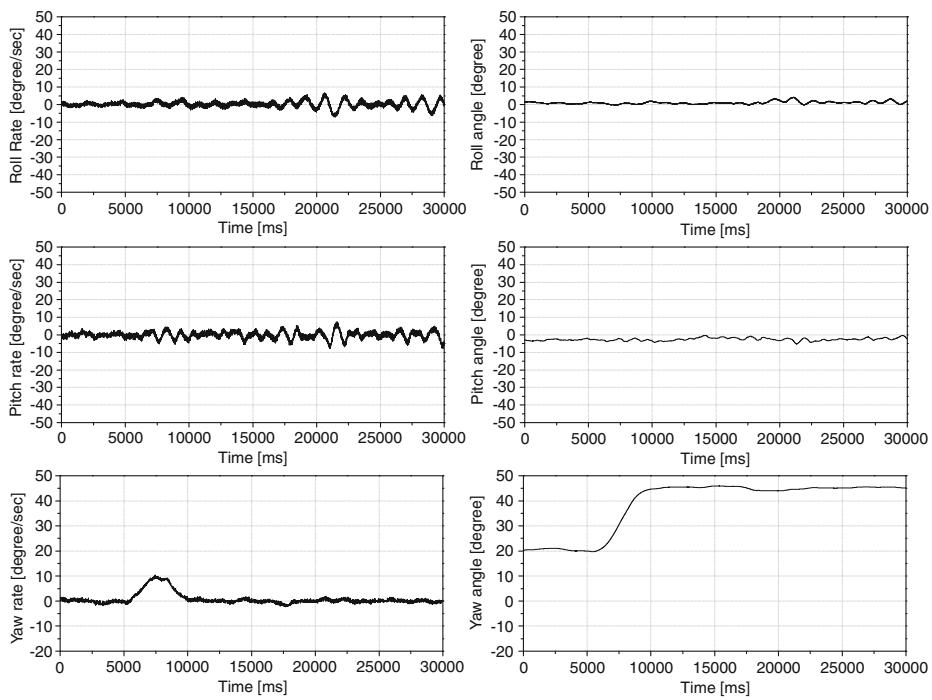
#### 4.2 Vision Based Localization

Figure 8 shows the scheme of the vision based localization. As the payload of the UAV is limited, red and green LED markers are attached at the bottom of the UAV, and a CCD camera is put on the ground. The CCD camera gets the image of the LED markers, and the image processor analyzes the color distribution of the markers, extract the center of each marker and finds the position and orientation of the markers as shown in Fig. 9 [19]. It takes 25ms for the image processing, and 5ms for serial communication.

#### 4.3 Experiments and Results

Figure 10 shows the experiments of the hovering control of the QRT UAV.

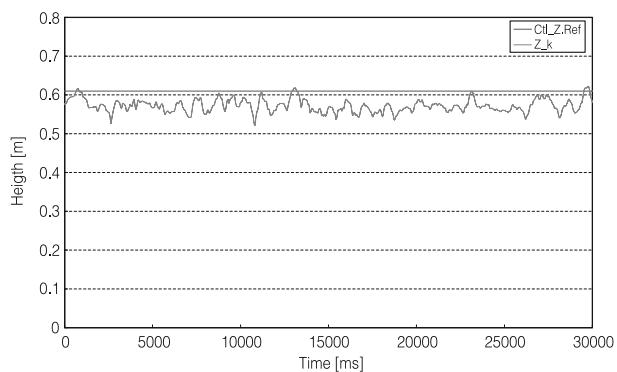
Figure 11 shows the control flow of flying robot system. If detecting obstacles, the flying robot system avoids the obstacles by moving in the opposite direction. All flying movements except for takeoff states start on the hovering state, and if a certain movement is finished, it will be come back into the hovering state again.



**Fig. 15** Experimental results for hovering and attitude control of the developed UAV: desired yaw angle of the UAV = +45°

Figures 12, 13, 14, 15, 16 show the experimental results for hovering performance using attitude and altitude control. The left column of Fig. 12 shows roll, pitch, and yaw rates (up to down) and the right column shows roll, pitch, and yaw angles during hovering. Figures 13–15 show three angle rates and angles for pitch, roll, and yaw controls. As shown in the Fig. 16, the performance of the altitude control is very satisfactory, even though the sonar sensor is used. The results show that the proposed algorithm for both the altitude and attitude control of the UAV works well.

**Fig. 16** Experimental results for hovering and altitude control of the developed UAV



For the hovering control, in this paper, a DOB controller with PID is proposed. With DOB, we can derive the linear equation, Eq. 37 using internal loop compensator, Eq. 36. As depicted in Fig. 4, a PID controller is used for the altitude and attitude of QRT UAVs. For the yaw controller, the I-gain is set to 0, because of the sensor noise. Normally the electrical compass is used for sensing the yaw motion, but the electrical compass is easily affected from other electrical systems, e.g., motor and battery.

## 5 Conclusion

In this study, a QRT UAV is developed for calamity observation in indoor environments. The hovering robot captures the image of targets under harmful environments, and sends the image to the operator on the safe site. The rigorous dynamic models of a QRT UAV were obtained both in the reference and body frame coordinate systems using the quasi-Lagrange equation. In many previous researches, they misused two subset motions as the whole body dynamics, however, at that case, the states of the whole body dynamics cannot be decoupled. A disturbance observer (DOB) based controller using the derived dynamic models was also proposed for hovering control and the vision based localization method. The control input induced by DOB will be helpful to use simple equations of motion satisfying the accurate derived dynamics. In addition, the UAV can also avoid obstacles using eight IR and four ultrasonic range sensors. The experiments were carried out to show the validity of the proposed control algorithm. The results show the attitude and height control results are very stable.

## References

1. Mistler, V., Benallegue, A., M'Sirdi, N.: Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In: Proceedings of IEEE International Workshop on Robot and Human Interactive Communication, pp. 586–593 (2001)
2. McKerrow, P.: Modeling the draganflyer for-rotor helicopter. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 3596–3601 (2004)
3. Mokhtari, A., Benallegue, A.: Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 2359–2366 (2004)
4. Castillo, P., Lozano, R., Dzul, A.: Stabilization of a mini-rotorcraft having four rotors. In: Proc., IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2693–2698 (2004)
5. Bouabdallah, S., Murrieri, P., Siegwart, R.: Design and control of an indoor micro quadrotor. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 4393–4398 (2004)
6. Bouabdallah, S., Siegwart, R.: Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 2259–2264 (2005)
7. Altug, E., Ostrowski, J., Mahony, R.: Control of a quadrotor helicopter using visual feedback. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 72–77 (2002)
8. Hamel, T., Mahony, R., Chriette, A.: Visual servo trajectory tracking for a four rotor vtol aerial vehicle. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 2781–2786 (2002)
9. Hamel, T., Mahony, R.: Pure 2d visual servo control for a class of under-actuated dynamic system. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 2229–2235 (2004)
10. Castillo, P., Dzul, A., Lozano, R.: Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Trans. Control Syst. Technol.* **12**(4), 510–516 (2004)
11. Guenard, N., Hamel, T., Moreau, V.: Dynamic modeling and intuitive control strategy for an x4-flyer. In: Proc. International Conf. on Control and Automation, pp. 141–146 (2005)

12. Kondak, K., Bernard, M., Meyer, N., Hommel, G.: Autonomously flying vtol-robots: modeling and control. In: Proc., IEEE Int. Conf. on Robotics and Automation, pp. 736–741 (2007)
13. Tayebi, A., McGilvray, S.: Attitude stabilization of a four-rotor aerial robot. In: Proc., IEEE Int. Conf. on Decision and Control, pp. 1216–1221 (2004)
14. Tayebi, A., McGilvray, S.: Attitude stabilization of a vtol quadrotor aircraft. IEEE Trans. Control Syst. Technol. **14**(3), 562–571 (2004)
15. Waslander, S., Hoffmann, G., Jang, J., Tomlin, C.: Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning. In: Proc., IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 468–473 (2005)
16. Meirovitch, L.: Dynamics and Control of Structures. Wiley, New York (1990)
17. Kaneko, K., Ohnishi, K., Komoriya, K.: A design method for manipulator control based on disturbance observer. In: Proc., IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1405–1412 (1994)
18. Fossen, T.I.: Guidance and Control of Ocean Vehicles. Wiley, New York (1994)
19. Yoon, K.-J., Jang, G.-J., Kim, S.-H., Kweon, I.-S.: Color landmark based self-localization for indoor mobile robots. J Control Autom Syst Eng **7**(9), 749–757 (2001)

# Modeling and System Identification of the muFly Micro Helicopter

Dario Schafroth · Christian Bermes ·  
Samir Bouabdallah · Roland Siegwart

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 27 October 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** An accurate mathematical model is indispensable for simulation and control of a micro helicopter. The nonlinear model in this work is based on the rigid body motion where all external forces and moments as well as the dynamics of the different hardware elements are discussed and derived in detail. The important model parameters are estimated, measured or identified in an identification process. While most parameters are identified from test bench measurements, the remaining ones are identified on subsystems using the linear prediction error method on real flight data. The good results allow to use the systems for the attitude and altitude controller design.

**Keywords** Coaxial micro helicopter · Nonlinear model · Parameter identification

## 1 Introduction

The European Framework project muFly has been initiated to develop a fully autonomous micro helicopter designated for tasks such as surveillance and security, search and rescue, as well as inspection and exploration. Target capabilities of the system are autonomous take off/hover/landing, obstacle avoidance and way point following. For such capabilities a fast and precise feedback control is needed on the low level, which makes an appropriate simulation model of the helicopter indispensable for the controller design. While there exist different other works on modeling conventional [1, 2] and coaxial helicopters [3, 4] it is necessary to adapt the model to the specialties of the used platform. One of those specialties is the stabilizer bar mounted on the helicopter, which passively stabilizes the system and

---

D. Schafroth (✉) · C. Bermes · S. Bouabdallah · R. Siegwart  
Autonomous System Lab, Swiss Federal Institute of Technology  
Zurich (ETHZ), Zurich, Switzerland  
e-mail: sdario@ethz.ch

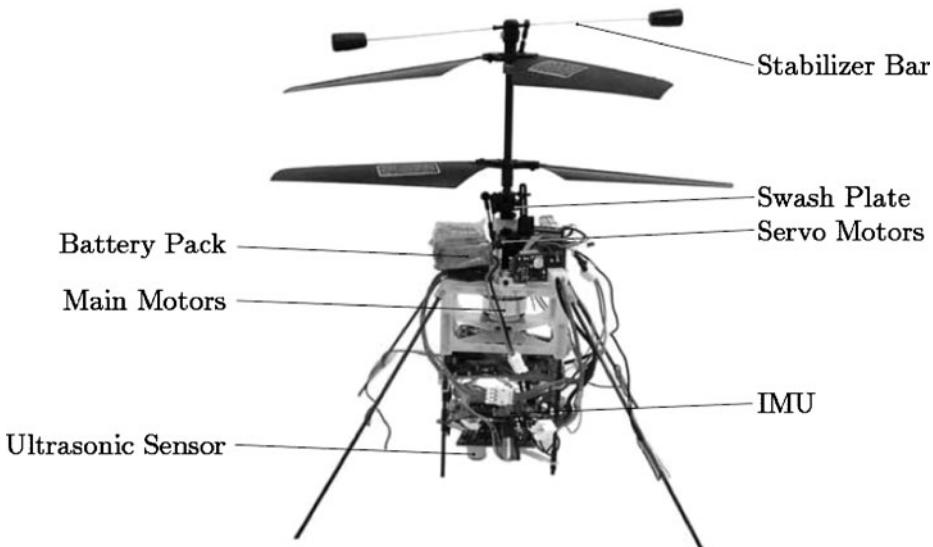
strongly influences the dynamics. Further devices, such as the electro motor and drive train are not considered in the found works, but are important for the designated control tasks. Therefore, a custom-made dynamic model is developed for muFly. The goal is to have a structured model reflecting accurately the physics of the different subsystems for simulations, but on the other side be as simple as possible since it will be used for the controller design later on.

The nonlinear model is based on the rigid body motions, where all the existing external forces and moments are discussed and derived. This is followed by defining the dynamics of the single mechanical devices such as swash plate, electro motors or stabilizer bar to complete the model. The developed model is then verified on real flight data and the system parameters are identified. This process is very important since a model with unknown or wrong parameters is worthless. For simplification the system identification is done on the linearized model using the prediction error method (PEM) [5]. Linearization allows to decouple the model in different subsystems and therefore strongly reduces the complexity. Using the linearized model is adequate since muFly will only operate close to the hover point. High velocities are not foreseen for the missions.

The paper is organized as follows. In Section 2 the muFly micro helicopter and its hardware setup is shown, followed by the nonlinear model in Section 3. The identification process with the PEM is presented in Section 4 and the identification and verification results are shown in Section 5.

## 2 The muFly Helicopter

muFly is a 17 cm in span, 15 cm in height coaxial helicopter with a mass of 95 g (Fig. 1). The two main rotors are driven by two lightweight brushless DC (BLDC)



**Fig. 1** The second prototype of the muFly helicopter

motors and are counter rotating to compensate the resulting torque due to aerodynamical drag. This allows to control the yaw by differential speed variation of the two rotors, whereas the altitude can be controlled varying the rotor speed simultaneously. The motor speed is reduced by a gear to achieve a higher torque on the rotor side.

A benefit of the coaxial setup is that one rotor can be used to help stabilizing the helicopter using a stabilizer bar. Such devices are often found on RC-Models and is mounted on the upper rotor. The helicopter is steered by a conventional swash plate actuated by two servos and powered by a lithium polymer battery. All the signals to the servos and motor controllers are pulse position modulated (PPM) signals, scaled to  $u_{\text{serv}} = \{-1, 1\}$  and  $u_{\text{mot}} = \{0, 1\}$ .

Sensors mounted on the platform are an inertial measurement unit (IMU) and an ultrasonic distance sensor for the measurement of the distance to the ground. With this setup, the attitude angles and the height over ground can be measured. Further, an omnidirectional camera in combination with lasers is in development, that will be used for the measurement of the horizontal position. The sensor data is processed by a dsPIC microprocessor and send to the ground station by a serial connection using a bluetooth module. To minimize time delays, the actuator inputs are send in the same package as the sensor data.

### 3 Nonlinear Modeling

The goal of the model presented in this paper is to be as simple as possible, since it will be used for the controller design. On the other hand, the physics and dynamics of the different devices mounted on the muFly helicopter have to be reflected accurately. The complete nonlinear model is described in the following sections.

#### 3.1 Coordinate Frames, Kinematics and Dynamics

As common two coordinate frames, the inertial frame  $J$  and the body-fixed frame  $B$  are defined for the model. Whereas the body-fixed frame is placed in the Center of Gravity (CoG) and moves with the helicopter, the inertial frame is fixed to the original location of the body-fixed frame. The frames are defined as it is common in aviation and are shown in the schematic view of the helicopter in Fig. 2.

Corresponding to these definitions the transformation from the inertial frame  $J$  to the body-fixed frame  $B$

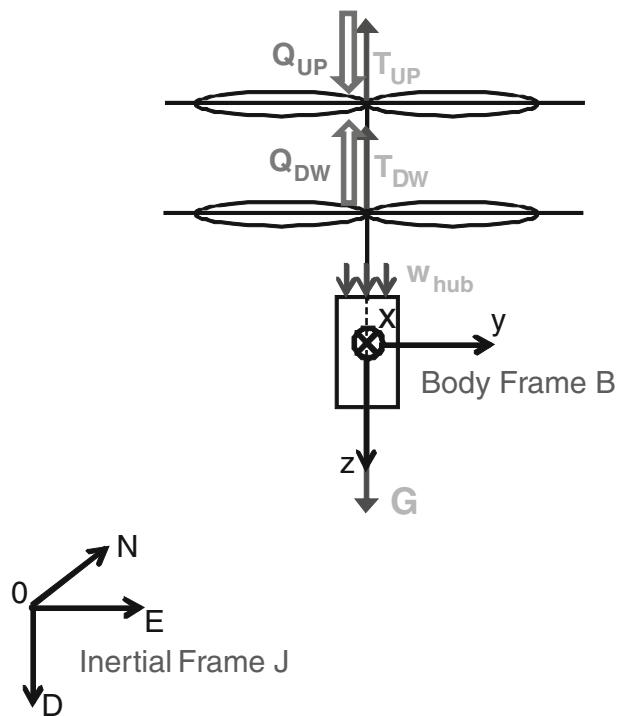
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underline{\underline{\mathbf{A}}}_{BJ} \cdot \begin{bmatrix} N \\ E \\ D \end{bmatrix} \quad (1)$$

is given by the transformation matrix

$$\underline{\underline{\mathbf{A}}}_{BJ} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\theta s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \quad (2)$$

obtained by the application of the Euler angles ( $c\alpha = \cos(\alpha)$  and  $s\alpha = \sin(\alpha)$ ).

**Fig. 2** Coordinate frame convention and the forces and moments acting in hover (back view)



It is well known that this transformation matrix is not valid for angular quantities. Therefore the angular velocities  $p, q, r$  have to be transformed using the transformation matrix

$$\underline{\underline{R}}_{JB} = \begin{bmatrix} 1 & \frac{s\phi s\theta}{s\theta} & \frac{s\phi s\theta}{s\theta} \\ 0 & \frac{s\phi}{s\theta} & -\frac{s\phi}{s\theta} \\ 0 & \frac{s\phi}{s\theta} & \frac{s\phi}{s\theta} \end{bmatrix} \quad (3)$$

that can be used to obtain a differential equation for the time derivatives of the roll, pitch and yaw angles  $\phi, \theta$  and  $\psi$ :

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \underline{\underline{R}}_{JB} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (4)$$

The next step is to set up the rigid body dynamics equations in the CoG in the body fixed coordinate frame. Using Newtonian Mechanics, the differential equations for the rigid body motion in the body-fixed frame become

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \mathbf{F} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (5)$$

and

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underline{\underline{\mathbf{I}}}^{-1} \left( \mathbf{M} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \underline{\underline{\mathbf{I}}} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \quad (6)$$

with the body velocities  $u, v, w$ , the system mass  $m$ , the body inertia tensor  $\underline{\underline{\mathbf{I}}}$  and the total external force and moment vectors  $\mathbf{F}$  and  $\mathbf{M}$ . For simplification, the inertia tensor  $\underline{\underline{\mathbf{I}}}$  has only diagonal elements. Such a simplification is feasible as a result of the symmetric design of muFly.

So far the equations of motion are independent of the flying platform and can be found in literature [2]. Now the platform dependent total external force  $\mathbf{F}$  and moment  $\mathbf{M}$  have to be defined.

### 3.2 Forces and Moments

In a stable hover position, as shown in Fig. 2, the thrust forces from the two rotors  $T_{\text{up}}$  and  $T_{\text{dw}}$  equal the gravitational force  $G$  caused by the mass of the helicopter and the integrated aerodynamical drag force on the fuselage  $W_{\text{hub}}$  due to the down wash of the rotors. The moments acting on the helicopter are the two drag torques  $Q_{\text{up}}$  and  $Q_{\text{dw}}$  from the counter rotating rotors (incl. stabilizer bar), which, if unbalanced, lead to a yaw motion of the helicopter.

In free flight, additional forces and moments result from the aerodynamical drag due to the motion through the air, but since the helicopter is small and reaches only low velocities, those can be neglected. More important are the moments resulting from the tilting of the tip path plane (TPP)—the plane the rotor blade tips are running in [6]—on the lower rotor caused by a cyclic pitch input from the swash plate [7] and on the upper rotor due to the stabilization mechanism. These moments are used for the steering and stabilization of the helicopter.

Summarized, the force vector  $\mathbf{F}$  and moment vector  $\mathbf{M}$  are

$$\begin{aligned} \mathbf{F} &= \mathbf{T}_{\text{up}} + \mathbf{T}_{\text{dw}} + \mathbf{G} + \mathbf{W}_{\text{hub}}, \\ \mathbf{M} &= \mathbf{Q}_{\text{up}} + \mathbf{Q}_{\text{dw}} + \mathbf{r}_{\text{Cup}} \times \mathbf{T}_{\text{up}} + \mathbf{r}_{\text{Cdw}} \times \mathbf{T}_{\text{dw}}, \end{aligned} \quad (7)$$

where  $\mathbf{r}_{\text{Cdw}}$  and  $\mathbf{r}_{\text{Cup}}$  are the vectors from the CoG to the hub of the lower respectively upper rotor. In order to achieve a drift free flight, an alignment of the CoG with the rotor axis is necessary, therefore the two vectors are assumed to have only a  $z$ -component.

The rotor thrust vector  $\mathbf{T}_i$  and torque vector  $\mathbf{Q}_i$  we define as  $\mathbf{T}_i = T_i \cdot \mathbf{n}_{Ti}$  and  $\mathbf{Q}_i = Q_i \cdot \mathbf{n}_{Qi}$ , with  $i = \{\text{dw}, \text{up}\}$  for the lower and upper rotor. In hover, the thrust magnitude  $T_i$  of a rotor with radius  $R$  can be defined as [8]

$$T_i = c_{Ti} \pi \rho R^4 \Omega_i^2 = c_{Ti} k_T \Omega_i^2, \quad (8)$$

with the air density  $\rho$ , the rotational velocity  $\Omega_i$  and the thrust coefficient  $c_{Ti}$ . While the air density  $\rho$  is assumed to be constant and the rotational velocity  $\Omega_i$  depends on the motor input, the thrust coefficient  $c_{Ti}$  depends on different elements such as the rotor setup and the flow characteristics. However for a given rotor the thrust coefficient can be evaluated and due to its definition assumed to be constant over the used range of rotational rotor velocities. Another factor that influences the thrust coefficient is the inflow velocity of the rotor. This concerns mainly the lower rotor

since it operates in the down wash of the upper rotor. Due to this down wash, the inflow velocity is increased and the thrust coefficient decreased [9]. This has to be considered by identifying the thrust coefficient of the lower rotor. The influence of the flow velocity due to movement in free flight are neglected in the model, since low inflow velocities do not significantly change the coefficients and high velocities are not foreseen for the dedicated missions.

The same assumptions are true for the torque coefficients  $c_{Qi}$  that are used to describe the drag torque values  $Q_i$  of the rotors and stabilizer bar:

$$Q_i = c_{Qi} \pi \rho R^5 \Omega_i^2 = c_{Qi} k_Q \Omega_i^2. \quad (9)$$

The direction of the thrust vector  $\mathbf{n}_i$  is perpendicular to the TPP [6] and can be defined using tilt angles around the x- and y- axis as shown in Fig. 3.

Using two rotational transformations in series, the vector is expressed in the body-fixed frame as

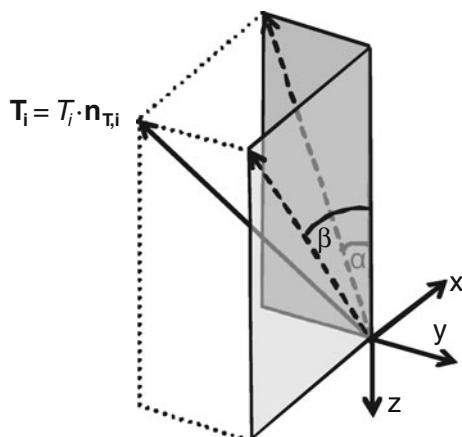
$$\mathbf{n}_{Ti} = \begin{bmatrix} \cos \alpha \sin \beta \\ \sin \alpha \\ -\cos \alpha \cos \beta \end{bmatrix}. \quad (10)$$

The direction of the rotor torque is equal to the z-axis, since the torque mainly acts on the rotor axis. It has to be considered that one rotor turns clockwise, while the other one turns counterclockwise.

$$\mathbf{n}_{Q_{up}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{n}_{Q_{dw}} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad (11)$$

The drag force value on the fuselage  $W_{hub}$  due to the down wash of the rotors is small, but nevertheless considered in the model as a help for the identification (see

**Fig. 3** Visualization of the tilted thrust vector with tilt angles  $\alpha$  and  $\beta$



Section 4.2). It is assumed to be parallel to the  $z$ -axis and constant, since the down wash is nearly the same around hover:

$$\mathbf{W}_{\text{hub}} = \begin{bmatrix} 0 \\ 0 \\ W_{\text{hub}} \end{bmatrix}. \quad (12)$$

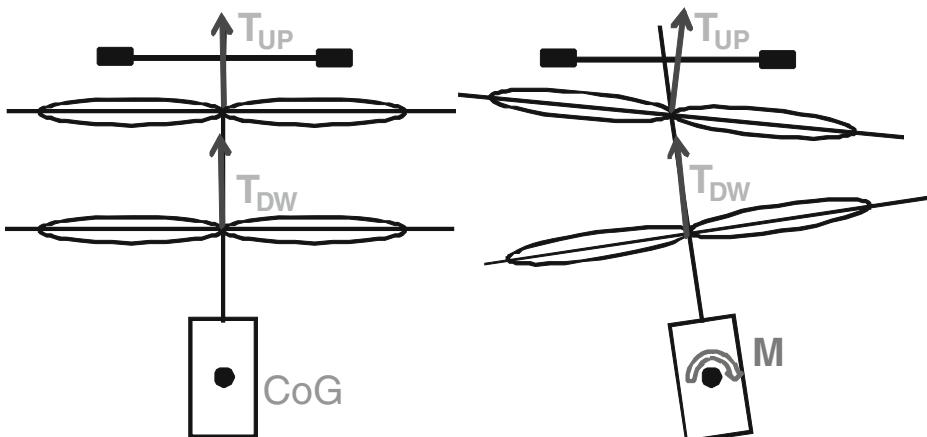
The last force is the gravitational force  $\mathbf{G}$  caused by the mass of the helicopter. This force is always directed parallel to the  $D$ -axis (positive direction) of the inertial frame with the value  $G = mg$ . However the force is needed in the body-fixed frame, thus it has to be transformed using the transformation matrix from Eq. 2

$$\mathbf{G} = \underline{\underline{\mathbf{A}}}_{BJ} \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = mg \cdot \begin{bmatrix} -\sin \theta \\ \cos \theta \sin \phi \\ \cos \theta \cos \phi \end{bmatrix} \quad (13)$$

With all forces and moments defined, the next step is to model the dynamics of the stabilizer bar, swash plate and the electro motors.

### 3.3 Stabilizer Bar and Swash Plate

An important part of the system model is the stabilizer bar. In simple words this stabilization mechanism gives cyclic pitch inputs, similar to the swash plate, to the upper rotor to stabilize the helicopter in flight. The stabilizer bar has a high inertia and lags behind a roll or pitch movement of the fuselage as shown in Fig. 4. Through a rigid connection to the rotor, this time delay results in a cycling pitching of the rotor blades and therefore to a tilting of the TPP. If the stabilizer bar is adjusted correctly the thrust vector shows in the opposite direction of the roll or pitch movement causing a redress moment.



**Fig. 4** The principle of the stabilizer bar. Due to the high inertia the stabilizer bar lags behind the roll/pitch movement, applies a cyclic pitch input to the rotor and creates a redress moment

The stabilizer bar following the roll/pitch movement can be modeled as a first order element as

$$\begin{aligned}\dot{\eta}_{\text{bar}} &= \frac{1}{T_{f,\text{up}}} (\phi - \eta_{\text{bar}}), \\ \dot{\zeta}_{\text{bar}} &= \frac{1}{T_{f,\text{up}}} (\theta - \zeta_{\text{bar}}),\end{aligned}\quad (14)$$

with angles  $\eta_{\text{bar}}$  and  $\zeta_{\text{bar}}$ . The tilt angles of the thrust vector in the body-fixed frame is the difference between the two angles  $\eta_{\text{bar}}$  and  $\zeta_{\text{bar}}$  scaled by a factor  $l_{\text{up}}$ , since we are interested in the tilt angle of the TPP and not that of the stabilizer bar. Thus the equations for the tilting angles for the thrust vector are

$$\begin{aligned}\alpha_{\text{up}} &= l_{\text{up}} (\eta_{\text{bar}} - \phi) \\ \beta_{\text{up}} &= l_{\text{up}} (\zeta_{\text{bar}} - \theta).\end{aligned}\quad (15)$$

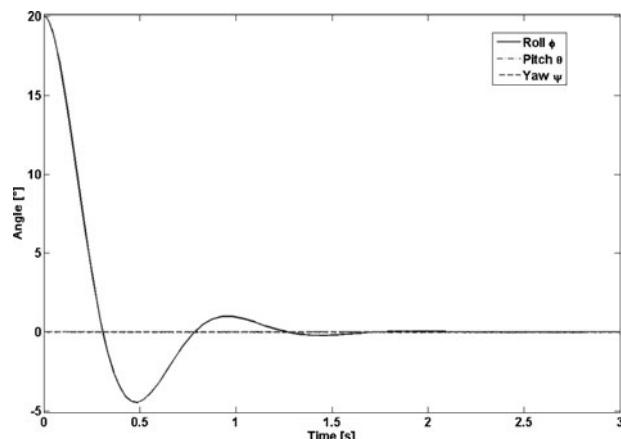
The influence of the modeled stabilizer bar is shown in Fig. 5 where the reaction of the helicopter to an initial displacement in the roll angle is plotted. After a short time period the helicopter is back in the hover position.

The idea of the swash plate model is the same as for the stabilizer bar. The reaction from the servo input (PPM signal) to the change in the TPP is also modeled by a first order system. Hereby all the dynamics of the servos and rotor are covered by the time constant  $T_{f,\text{dw}}$ . The tilting angles of the lower rotor are modeled as

$$\begin{aligned}\dot{\alpha}_{\text{dw}} &= \frac{1}{T_{f,\text{dw}}} (-l_{\text{dw}} u_{\text{serv}2} \cdot \theta_{\text{SPmax}} - \alpha_{\text{dw}}), \\ \dot{\beta}_{\text{dw}} &= \frac{1}{T_{f,\text{dw}}} (-l_{\text{dw}} u_{\text{serv}1} \cdot \theta_{\text{SPmax}} - \beta_{\text{dw}}),\end{aligned}\quad (16)$$

with the time constant  $T_{f,\text{dw}}$ , scaling factor  $l_{\text{dw}}$ , maximal swash plate tilting angle  $\theta_{\text{SPmax}}$  and servo inputs  $u_{\text{serv},i}$ .

**Fig. 5** The reaction of the helicopter to an initial roll displacement of 20° with the modeled stabilizer bar



### 3.4 Electro Motor

The used BLDC motors are controlled by off-the-shelf motor controllers. They have the drawback that the motors rotational speed is not given and motor speed control is not possible. Instead of measuring the speed externally, for instance with an optical device, the motor dynamics are included directly in the model. This allows to use the model for every kind of motor without the help of any external sensor.

The differential equations for electro motors are well known [10] and can be simplified to an equation for the motor speed  $\omega$  in the form of

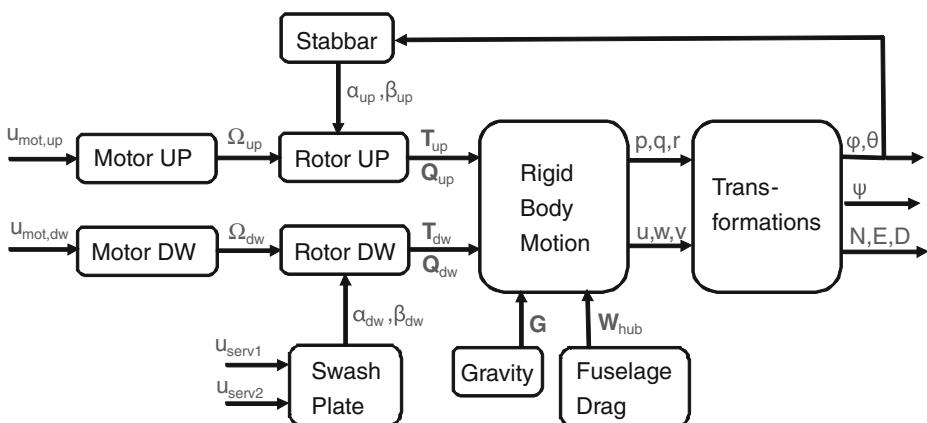
$$J_{\text{mot}} \dot{\omega} = \frac{\kappa_M U - \kappa_M \kappa_E \omega}{R_\Omega} - d_R \omega - M_L, \quad (17)$$

with the moment of inertia  $J_{\text{mot}}$ , electrical and mechanical motor constants  $\kappa_E$  and  $\kappa_M$ , the input voltage  $U$ , the electrical resistance  $R_\Omega$ , the friction coefficient  $d_R$  and the external torque  $M_L$ .

In the case of the helicopter, the external torque  $M_L$  is the drag torque value  $Q_i$  of the rotor as shown in Eq. 9. Further, for the model the rotor speed is more important than the motor speed, thus the equation is expanded by the gears to obtain the final equation for the rotor speed  $\Omega_i$

$$J_{\text{drive}} \dot{\Omega}_i = \frac{\kappa_M U_{\text{bat}} u_{\text{mot},i} - \kappa_M \kappa_E i_{\text{gear}} \Omega_i}{i_{\text{gear}} R_\Omega} - d_R \Omega_i - \frac{c_{Q_i} k_Q \Omega_i^2}{i_{\text{gear}}^2 \cdot \eta_{\text{gear}}} \quad (18)$$

with the gear ratio  $i_{\text{gear}}$ , the efficiency of the gear  $\eta_{\text{gear}}$ , the battery voltage  $U_{\text{bat}}$  and the motor input  $u_{\text{mot},i}$ .



**Fig. 6** muFly dynamic model block diagram

### 3.5 Model Summary

With this set of differential equations, the nonlinear model is complete. The full model consists of 18 states and four inputs (two motors and two servos):

$$\begin{aligned} x &= [x, y, z, u, v, w, \phi, \theta, \psi, p, q, r, \alpha_{dw}, \beta_{dw}, \alpha_{up}, \beta_{up}, \Omega_{dw}, \Omega_{up}]^T, \\ u &= [u_{mot,dw}, u_{mot,up}, u_{serv1}, u_{serv2}]^T. \end{aligned} \quad (19)$$

As an overview, the system is shown as a block diagram in Fig. 6. The equations of the complete system are given in the Appendix A.1.

## 4 Parameter Identification

After setting up the physical equations, the missing parameters have to be adjusted to the real helicopter. Without appropriate parameters, the best model is worthless.

### 4.1 Mechanical Parameters

The identification process is a non trivial task, and it is hard to find the correct parameters, especially since most of the parameters are coupled. In order to minimize the complexity of the identification on flight data, it is necessary to measure or estimate as many parameters as possible beforehand. While some parameters such as the mass, maximal swash plate angle, gear ratio, rotor diameter and body inertias can be easily measured or taken from the CAD design (all the equipment is modeled), it is more difficult to identify parameters such as the aerodynamical coefficients, electro motor constants or the time constants for the stabilizer bar and the swash plate.

### 4.2 Aerodynamical Parameters

The aerodynamical coefficients are calculated from the measurements on a coaxial rotor test-bench designed for blade optimization [11].

On this test-bench it is possible to identify the coefficients  $c_{Ti}$  and  $c_Q$  by measuring the torque and thrust of the lower and upper rotor. Even if it would have been possible, the torque coefficient of the stabilizer bar  $c_{Q_{up,stab}}$  is not measured and only estimated instead. This parameter is identified later together with the fuselage drag  $W_{hub}$  to achieve hovering condition with the recorded motor inputs for hovering. In other words, they are used as tuning parameters to correct the errors from the measurements to fulfill the equations in hover (index ‘hov’)

$$\begin{aligned} T_{up,hov} + T_{dw,hov} &= G + W_{hub}, \\ Q_{up,rot,hov} + Q_{up,stab,hov} &= Q_{dw,hov}, \end{aligned} \quad (20)$$

where as the drag torque of the upper rotor  $Q_{up}$  is split in the rotor part  $Q_{up,rot}$  and stabilizer bar  $Q_{up,stab}$ .

### 4.3 Electro Motor

The parameters for the low cost off-the-shell electro motors are not available, thus experimental data has to be used for the identification. Those motor measurements have been done by our partner CEDRAT [12] by applying a constant voltage, varying the external torques on the motors and measuring the rotational speed and current. The motor constants are identified using the stationary solution  $\dot{\omega} = 0$  of Eq. 17 and the Least-Square method (LS) [13]. Result plots for the identification are shown in Fig. 7.

### 4.4 Prediction Error Method

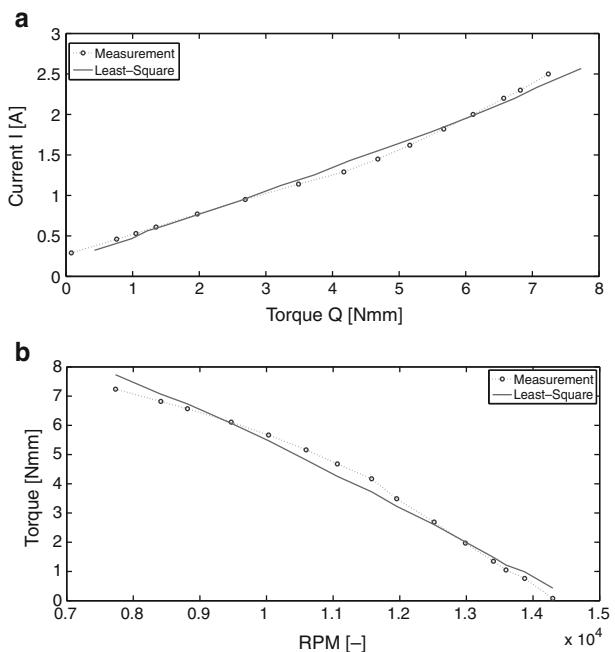
The remaining unknown parameters are identified dynamically using real flight data. Since the equations are not static anymore, a dynamic identification process has to be used. In this case, a linear Prediction Error Method (PEM) is chosen. This method is widely used in aeronautics and compares the measurement data vector  $\mathbf{y}(t)$  with the predicted output of the dynamic model  $\hat{\mathbf{y}}(t|t-1, \Theta)$  using the last  $t-1$  measurements and the parameter vector  $\Theta$  [14]. The difference between the measurement and the prediction

$$\epsilon(t, \Theta) = \mathbf{y}(t) - \hat{\mathbf{y}}(t|t-1, \Theta) \quad (21)$$

is called the prediction error. The covariance matrix  $\underline{\underline{\mathbf{R}}}_N(\Theta)$  is defined using all errors  $\epsilon(1, \Theta), \dots, \epsilon(N, \Theta)$ :

$$\underline{\underline{\mathbf{R}}}_N(\Theta) = \frac{1}{N} \sum_{t=1}^N \epsilon(t, \Theta) \epsilon^T(t, \Theta) \quad (22)$$

**Fig. 7** Least square result plots for the electro motor parameter identification



The wanted parameter vector  $\Theta_{\min}$  is now calculated such that the loss function  $V_N(\Theta)$  is minimized:

$$V_N(\Theta) = \det \left( \underline{\underline{R}}_N(\Theta) \right). \quad (23)$$

For applying this method, the model has to be linearized around an operation point, in this case hover. Since only sensors for the attitude angles and the distance to the ground are mounted, the identification of  $x/y$ - dynamics of the model is not possible yet and is future work. Therefore the linearized model is reduced by the horizontal linear motion states, resulting in a state-space system of 14 states. It is almost impossible to identify all the parameters at once for such a high order system, therefore the system is split into four decoupled subsystems: pitch, roll, heave and yaw. As an example the subsystems for pitch and heave are shown in Eqs. 24 and 25, 26, respectively:

$$A_{\text{pitch}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ c_{T_{\text{up}}} k_T l_{\text{up}} \Omega_{\text{up},0}^2 z_{\text{up}} & 0 & c_{T_{\text{dw}}} k_T \Omega_{\text{dw},0}^2 z_{\text{dw}} & -c_{T_{\text{up}}} k_T l_{\text{up}} \Omega_{\text{up},0}^2 z_{\text{up}} \\ I_{yy} & 0 & I_{yy} & I_{yy} \\ 0 & 0 & -\frac{1}{T_{f,\text{dw}}} & 0 \\ -\frac{1}{T_{f,\text{up}}} & 0 & 0 & \frac{1}{T_{f,\text{up}}} \end{bmatrix},$$

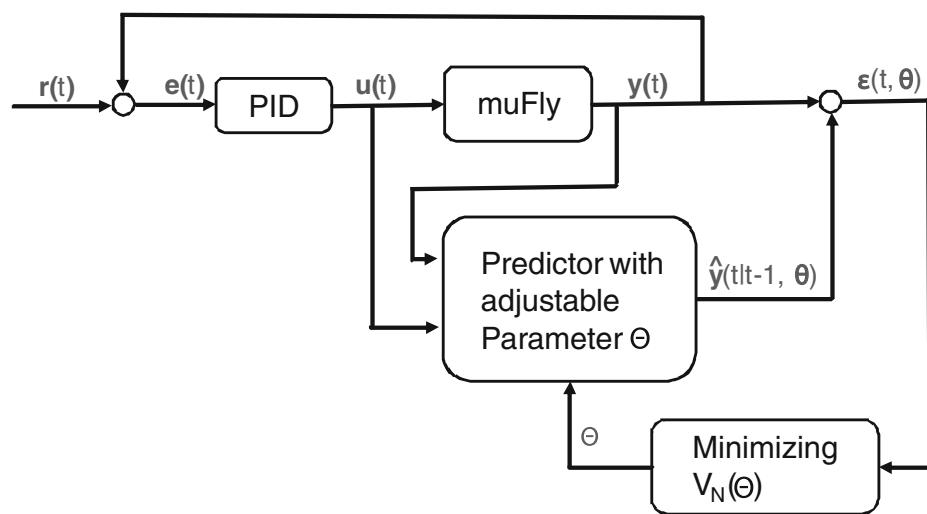
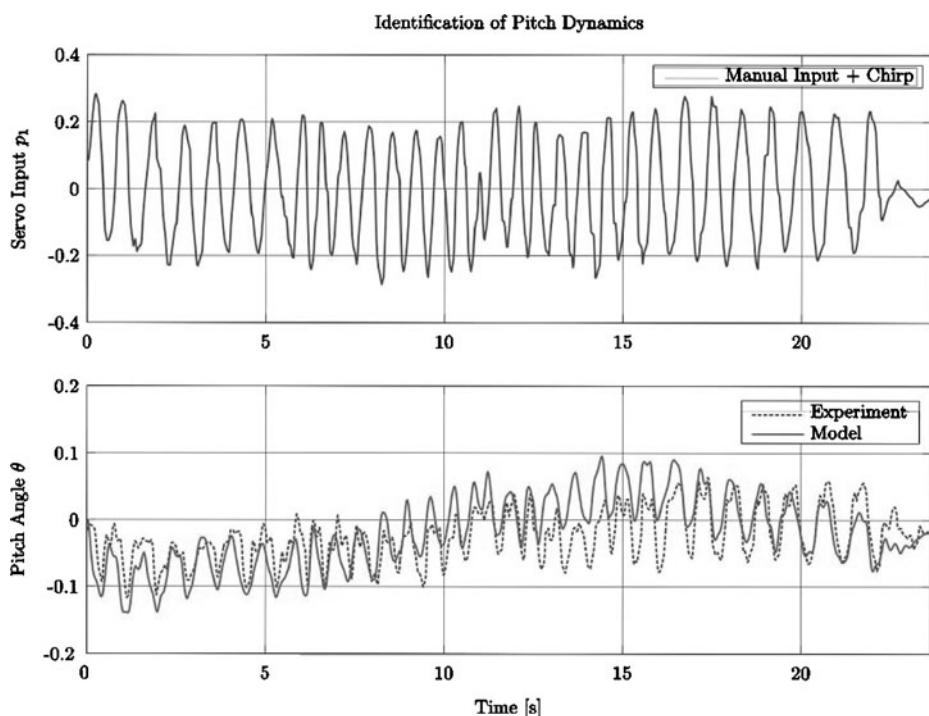
$$B_{\text{pitch}} = \begin{bmatrix} 0 \\ 0 \\ \frac{l_{\text{dw}} \theta_{\text{SPmax}}}{T_{f,\text{dw}}} \\ 0 \end{bmatrix} C_{\text{pitch}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} D_{\text{pitch}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (24)$$

The states and the input for the pitch subsystems are  $x_{\text{pitch}} = [\theta, q, \beta_{\text{dw}}, \beta_{\text{up}}]^T$  and  $u_{\text{pitch}} = u_{\text{serv1}}$ .

$$A_{\text{heave}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{2c_{T_{\text{dw}}} k_T \Omega_{\text{dw},0}}{m} & -\frac{2c_{T_{\text{up}}} k_T \Omega_{\text{up},0}}{m} \\ 0 & 0 & M_{\text{dw}} & 0 \\ 0 & 0 & 0 & M_{\text{up}} \end{bmatrix},$$

$$B_{\text{heave}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\kappa_M U_{\text{bat}}}{i_{\text{gear}} J_{\text{drive,dw}} R_\Omega} & 0 \\ 0 & \frac{\kappa_M U_{\text{bat}}}{i_{\text{gear}} J_{\text{drive,up}} R_\Omega} \end{bmatrix},$$

$$C_{\text{heave}} = [1 \ 0 \ 0 \ 0], D_{\text{heave}} = [0 \ 0], \quad (25)$$

**Fig. 8** Identification process block diagram**Fig. 9** Identification result of the pitch subsystem

with

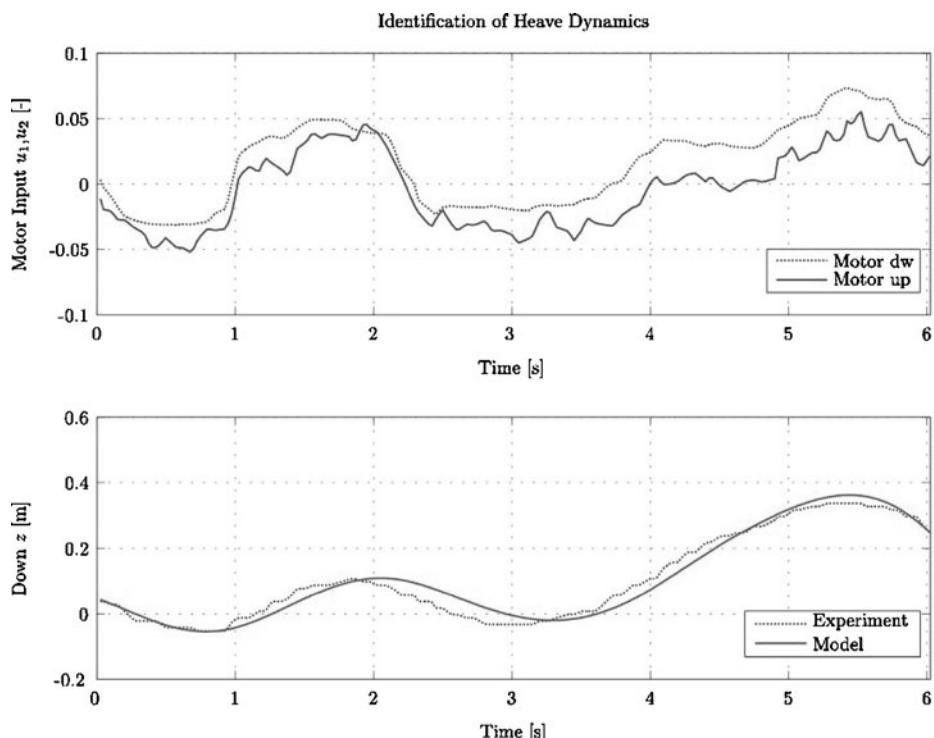
$$\begin{aligned} M_{\text{dw}} &= \frac{1}{J_{\text{drive,dw}}} \cdot \left( -d_R - \frac{2c_{Q_{\text{dw}}}k_Q\Omega_{\text{dw},0}}{\eta_{\text{gear}}i_{\text{gear}}^2} - \frac{\kappa_E K_M}{R_\Omega} \right), \\ M_{\text{up}} &= \frac{1}{J_{\text{drive,up}}} \cdot \left( -d_R - \frac{2c_{Q_{\text{up}}}k_Q\Omega_{\text{up},0}}{\eta_{\text{gear}}i_{\text{gear}}^2} - \frac{\kappa_E K_M}{R_\Omega} \right). \end{aligned} \quad (26)$$

The states and the inputs for the heave subsystem are  $x_{\text{heave}} = [z, w, \Omega_{\text{dw}}, \Omega_{\text{up}}]^T$  and the two motor inputs  $u_{\text{heave}} = [u_{\text{mot,dw}}, u_{\text{mot,up}}]^T$ .

The system for the roll dynamics is similar to the one of the pitch and the yaw dynamics similar to the heave. Therefore it is abandoned to show those systems here and are listed in the Appendix A.2.

The subsystems are identified after each other and it has to be considered that some parameters affect different subsystems. As an example the time constant  $T_{f,\text{dw}}$  affects the roll and pitch subsystem. The time constant is identified on the roll system and kept constant in the pitch identification.

While the pitch and roll subsystems are easy to identify, the heave and yaw subsystems are very challenging due to the two poles at zero (critically stable). Due to this behavior it is difficult to identify the system over a long period of time without divergence, thus a good sequence has to be found. Further, the identification

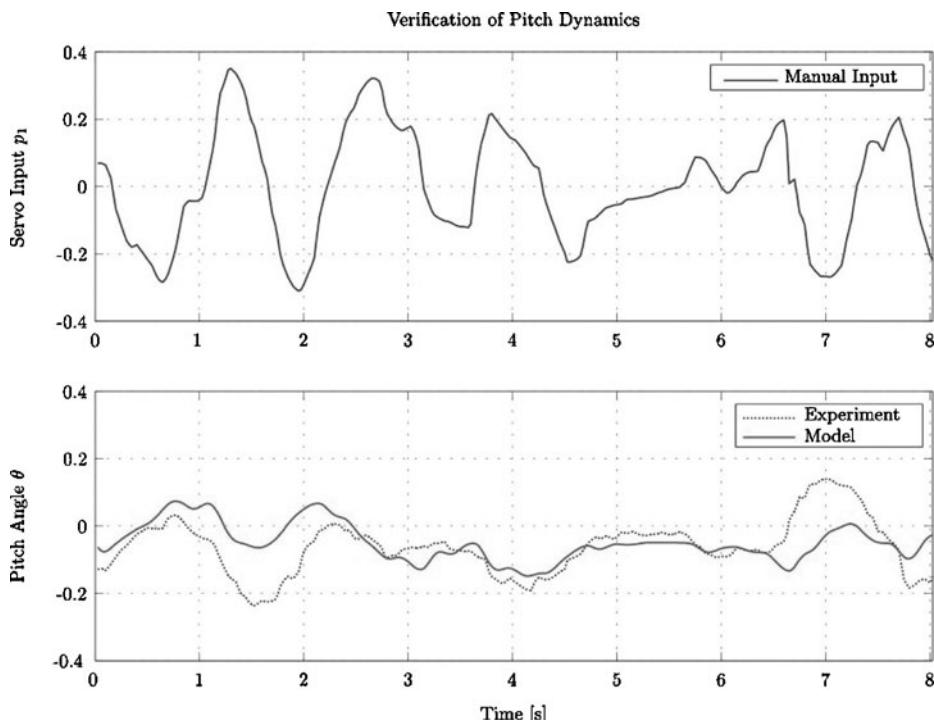


**Fig. 10** Identification result of the heave subsystem

process should start at the hover point and the inputs for hovering have to be determined from the measurements. Deviation in those inputs and in the unknown initial states leads to strong deviation in the result plots. Therefore the initial states and a deviation from the determined hover inputs are used as tuning parameters in the identification process.

#### 4.5 Data Generation

For recording the flight data, the helicopter is steered by a pilot. In order to cover as much frequency bandwidth as possible, a chirp signal is generated and superimposed on the pilot input. However, the helicopter is not adequately controllable in open loop by a pilot. Therefore an additional PID controller is used to control attitude and help the pilot. Effectively, the pilot controls the set point values of the PID controller. The controller is not problematic for the parameter identification, since the actuator signals are recorded and send together with the sensor data. Hence the identification of the system is independent of the controller. An identification process overview is shown as a block diagram in Fig. 8. It shows the reference value input  $r(t)$  of the pilot as the input of the PID controller. The output of the controller  $u(t)$ , the motor and servo inputs, is send to the helicopter and the respective subsystem, where a new output  $\hat{y}(t|t-1, \Theta)$  is predicted. This output is compared to the respective



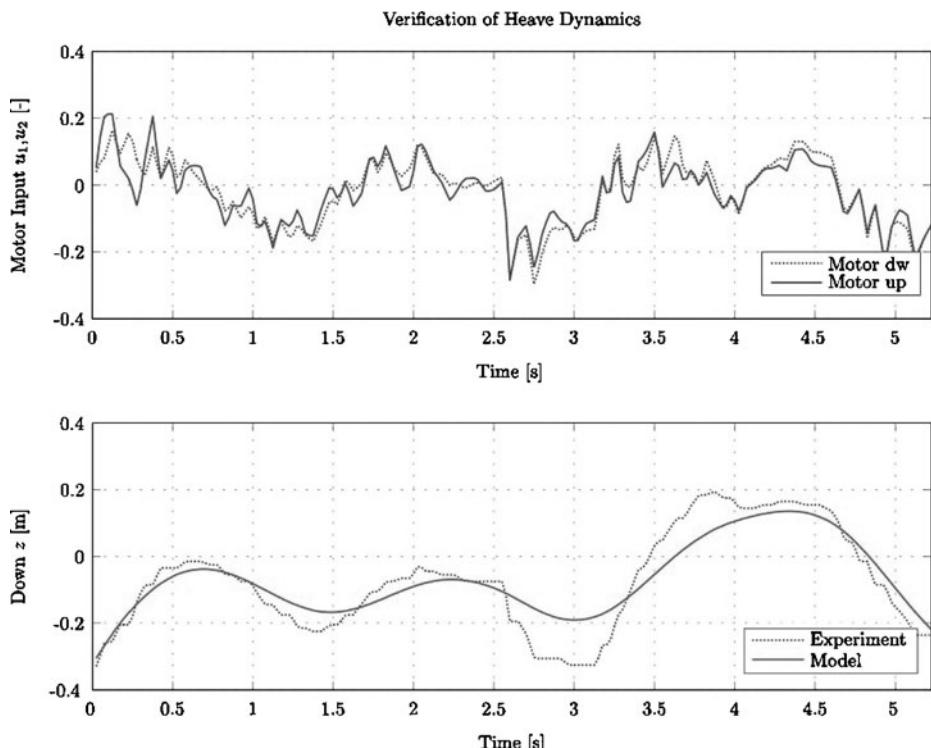
**Fig. 11** Verification of the pitch subsystem

sensor measurement  $\mathbf{y}(t)$  and the error  $\epsilon$  is build and minimized by adjusting the parameters  $\Theta$ .

## 5 Results

The results for the identified subsystems for pitch and heave are shown in Figs. 9 and 10. In general the identified models show a good correlation with the real system. The main deviation is in the amplitude, which is not critical for the controller design, since those differences will be covered by the gain of the controller. More important is the phase, which shows that the subsystems are in the right dimensions. Well visible is the different time span of identification for the two subsystems. While the pitch subsystem can be identified over a time span of more than 20 seconds it is challenging to identify the heave system for a period longer than 10 seconds.

An identification alone is not sufficient, the model and the parameters have to be verified on an other flight sequence. For the verifications only the initial states and inputs are identified, all the parameters are taken from the previous identification. The results for the two subsystems are shown in Figs. 11 and 12. Those verification plots show again an acceptable correlation with the real platform, even if the performance is slightly worse. Here the model underestimates the amplitude,



**Fig. 12** Verification of the heave subsystem

however, the more important phase matches again. All in all the identification and verification show good results and a proper controller design with those systems should be possible. A list of all the identified system parameters is given in Table 1 in the Appendix A.3.

## 6 Summary and Future Work

In this paper the development of a nonlinear model for the coaxial micro helicopter muFly is presented. An appropriate model is essential for the controller design and simulations. The model is based on the rigid body dynamics where all the possible acting forces and moments are discussed. Further the dynamics of the mechanical devices such as the swash plate, electro motor and stabilizer bar are derived, resulting in a complete structured model.

However, a good model with bad parameters is worthless, hence those parameters have to be identified. While most parameters are measured from CAD data or on test benches, some parameters have to be identified on real flight data using identification methods. Since muFly will work mainly around the hover point, a linear prediction error method is chosen to be sufficient. The identification process is discussed and shown in detail, with specialties of using a PID controller to support the pilot or parameters which are not measured, but used as a correction of the measurements.

At last the results are presented showing the performance of the identified and verified models. The identified subsystems are now ready to be used for simulations and controller design. In future the models will be used to design and test altitude and attitude controllers using linear optimal controller techniques such as Linear Quadratic Gaussian (LQG) or  $H_\infty$ . Another idea is to identify the nonlinear model by using a nonlinear identification approach (e.g. based on neural networks), allowing to use nonlinear control techniques. After the completion of the position sensor the identified model will be used for a prediction of the position in a filter approach. The validation and verification of the x/y-dynamics of the model will be part of this process.

**Acknowledgements** The authors would like to thank Mr. P. Sax, Mr. F. Haenni, Mr. M. Buehler, Mr. D. Fenner and Mr. T. Baumgartner for their support. muFly is a STREP project under the Sixth Framework Programme of the European Commission, contract No. FP6-2005-IST-5-call 2.5.2 Micro/Nano Based Sub-Systems FP6-IST-034120. The authors gratefully acknowledge the contribution of our muFly project partners BeCAP at Berlin University of Technology, CEDRAT Technologies, CSEM, Department of Computer Science at University of Freiburg and XSENS Motion Technologies.

## Appendix

### A.1 Nonlinear Model

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underline{\underline{\mathbf{A}}}_{BJ} \cdot \begin{bmatrix} N \\ E \\ D \end{bmatrix} \quad (27)$$

$$\underline{\underline{\mathbf{A}}}_{BJ} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\theta s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \quad (28)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \underline{\underline{\mathbf{R}}}_{JB} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (29)$$

$$\underline{\underline{\mathbf{R}}}_{JB} = \begin{bmatrix} 1 & \frac{s\phi s\theta}{s\theta} & \frac{s\phi s\theta}{s\theta} \\ 0 & \frac{s\theta}{s\phi} & \frac{-s\phi}{s\theta} \\ 0 & \frac{s\phi}{s\theta} & \frac{s\phi}{s\theta} \end{bmatrix} \quad (30)$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \mathbf{F} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (31)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underline{\underline{\mathbf{I}}}^{-1} \left( \mathbf{M} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \underline{\underline{\mathbf{I}}} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \quad (32)$$

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \mathbf{T}_{up} + \mathbf{T}_{dw} + \mathbf{G} + \mathbf{W}_{hub} \quad (33)$$

$$\mathbf{M} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \mathbf{Q}_{up} + \mathbf{Q}_{dw} + \mathbf{r}_{Cup} \times \mathbf{T}_{up} + \mathbf{r}_{Cdw} \times \mathbf{T}_{dw} \quad (34)$$

$$\begin{aligned} F_x &= c_{Tdw} k_T \Omega_{dw}^2 \cos \beta_{dw} \sin \alpha_{dw} - c_{Tup} k_T \Omega_{up}^2 \sin \beta_{up} - mg \sin \theta \\ F_y &= c_{Tdw} k_T \Omega_{dw}^2 \sin \beta_{dw} + c_{Tup} k_T \Omega_{up}^2 \cos \beta \sin \alpha_{up} + mg \cos \theta \sin \phi \\ F_z &= -c_{Tdw} k_T \Omega_{dw}^2 \cos \alpha_{dw} \cos \beta_{dw} - c_{Tup} k_T \Omega_{up}^2 \cos \alpha_{up} \cos \beta_{up} \\ &\quad + W_{hub} + mg \cos \theta \cos \phi \end{aligned} \quad (35)$$

$$\begin{aligned} M_x &= -z_{dw} c_{Tdw} k_T \Omega_{dw}^2 \sin \beta_{dw} - z_{up} c_{Tup} k_T \Omega_{up}^2 \cos \beta_{up} \sin \alpha_{up} \\ M_y &= z_{dw} c_{Tdw} k_T \Omega_{dw}^2 \cos \beta_{dw} \sin \alpha_{dw} - z_{up} c_{Tup} k_T \Omega_{up}^2 \sin \beta_{up} \\ M_z &= (c_{Qup,rot} + c_{Qup,stab}) k_Q \Omega_{up}^2 - c_{Qdw} k_Q \Omega_{dw}^2 \end{aligned} \quad (36)$$

$$\begin{aligned} \alpha_{up} &= l_{up} (\eta_{bar-\phi}) \\ \beta_{up} &= l_{up} (\zeta_{bar-\theta}). \end{aligned} \quad (37)$$

$$\begin{aligned} \dot{\eta}_{bar} &= \frac{1}{T_{f,up}} (\phi - \eta_{bar}), \\ \dot{\zeta}_{bar} &= \frac{1}{T_{f,up}} (\theta - \zeta_{bar}), \end{aligned} \quad (38)$$

$$\begin{aligned}\dot{\alpha}_{\text{dw}} &= \frac{1}{T_{f,\text{dw}}}(-l_{\text{dw}}u_{\text{serv}2} \cdot \theta_{\text{SPmax}} - \alpha_{\text{dw}}), \\ \dot{\beta}_{\text{dw}} &= \frac{1}{T_{f,\text{dw}}}(-l_{\text{dw}}u_{\text{serv}1} \cdot \theta_{\text{SPmax}} - \beta_{\text{dw}})\end{aligned}\quad (39)$$

$$\begin{aligned}\dot{\Omega}_{\text{dw}} &= \frac{1}{J_{\text{drive,dw}}} \left( \frac{\kappa_M U_{\text{bat}} u_{\text{mot,dw}} - \kappa_M \kappa_E i_{\text{gear}} \Omega_{\text{dw}}}{i_{\text{gear}} R_\Omega} - d_R \Omega_{\text{dw}} - \frac{c_Q k_Q \Omega_{\text{dw}}^2}{i_{\text{gear}}^2 \cdot \eta_{\text{gear}}} \right) \\ \dot{\Omega}_{\text{up}} &= \frac{1}{J_{\text{drive,up}}} \left( \frac{\kappa_M U_{\text{bat}} u_{\text{mot,up}} - \kappa_M \kappa_E i_{\text{gear}} \Omega_{\text{up}}}{i R_\Omega} - d_R \Omega_{\text{up}} - \frac{c_Q k_Q \Omega_{\text{up}}^2}{i_{\text{gear}}^2 \cdot \eta_{\text{gear}}} \right)\end{aligned}\quad (40)$$

## A.2 Linear Subsystems for Roll and Yaw

*Roll*

$$A_{\text{roll}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{c_{T_{\text{up}}} k_T l_{\text{up}} \Omega_{\text{up},0}^2 z_{\text{up}}}{I_{xx}} & 0 & -\frac{c_{T_{\text{dw}}} k_T \Omega_{\text{dw},0}^2 z_{\text{dw}}}{I_{xx}} & -\frac{c_{T_{\text{up}}} k_T l_{\text{up}} \Omega_{\text{up},0}^2 z_{\text{up}}}{I_{xx}} \\ I_{xx} & I_{xx} & I_{xx} & I_{xx} \\ 0 & 0 & -\frac{1}{T_{f,\text{dw}}} & 0 \\ -\frac{1}{T_{f,\text{up}}} & 0 & 0 & \frac{1}{T_{f,\text{up}}} \end{bmatrix}$$

$$B_{\text{roll}} = \begin{bmatrix} 0 \\ 0 \\ \frac{l_{\text{dw}} \theta_{\text{SPmax}}}{T_{f,\text{dw}}} \\ 0 \end{bmatrix} \quad C_{\text{roll}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad D_{\text{roll}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (41)$$

The states and the input are  $x_{\text{roll}} = [\phi, p, \alpha_{\text{dw}}, \alpha_{\text{up}}]^T$  and  $u_{\text{roll}} = u_{\text{serv}2}$ .

*Yaw*

$$A_{\text{yaw}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{2c_{Q_{\text{dw}}} k_Q \Omega_{\text{dw},0}}{I_{zz}} & \frac{2c_{Q_{\text{up}}} k_Q \Omega_{\text{up},0}}{I_{zz}} \\ 0 & 0 & M_{\text{dw}} & 0 \\ 0 & 0 & 0 & M_{\text{up}} \end{bmatrix}$$

$$B_{\text{yaw}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\kappa_M U_{\text{bat}}}{i_{\text{gear}} J_{\text{drive,dw}} R_\Omega} & 0 \\ 0 & \frac{\kappa_M U_{\text{bat}}}{i_{\text{gear}} J_{\text{drive,up}} R_\Omega} \end{bmatrix}$$

$$C_{\text{yaw}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad D_{\text{yaw}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (42)$$

with

$$M_{\text{dw}} = \frac{1}{J_{\text{drive,dw}}} (-d_R - \frac{2c_Q k_Q \Omega_{\text{dw},0}}{\eta_{\text{gear}} i_{\text{gear}}^2} - \frac{\kappa_E \kappa_M}{R_\Omega}) \quad (43)$$

and

$$M_{\text{up}} = \frac{1}{J_{\text{drive,up}}} (-d_R - \frac{2c_Q k_Q \Omega_{\text{up},0}}{\eta_{\text{gear}} i_{\text{gear}}^2} - \frac{\kappa_E \kappa_M}{R_\Omega}) \quad (44)$$

The states and the inputs are  $x_{\text{yaw}} = [\psi, r, \Omega_{\text{dw}}, \Omega_{\text{up}}]^T$  and  $u_{\text{yaw}} = [u_{\text{mot,dw}}, u_{\text{mot,up}}]^T$ .

### A.3 The Identified Parameters

**Table 1** Identified parameters

Parameter	Description	Identification	Value	Unit
$m$	Mass	Measured	0.095	kg
$I_{xx}$	Inertia around $x$ -axis	CAD/PEM	$1.24e^{-4}$	$\text{kg m}^2$
$I_{yy}$	Inertia around $y$ -axis	CAD/PEM	$1.30e^{-4}$	$\text{kg m}^2$
$I_{zz}$	Inertia around $z$ -axis	CAD/PEM	$6.66e^{-5}$	$\text{kg m}^2$
$z_{\text{dw}}$	Distance CoG lower rotor hub	CAD	-0.051	m
$z_{\text{up}}$	Distance CoG upper rotor hub	Measured	-0.091	m
$\Theta_{\text{SP,max}}$	Maximal swash plate angle	Measured	15	°
$R$	Rotor radius	Measured	0.0875	m
$c_{T\text{dw}}$	Thrust coefficient lower rotor	Test bench	0.0115	—
$c_{T\text{up}}$	Thrust coefficient upper rotor	Test bench	0.0131	—
$c_{Q\text{dw}}$	Torque coefficient lower rotor	Test bench	0.0018	—
$c_{Q\text{up,rot}}$	Torque coefficient upper rotor	Test bench	0.0019	—
$c_{Q\text{up,stab}}$	Torque coefficient stabilizer bar	Hover/PEM	$3.58e^{-5}$	—
$J_{\text{drive,dw}}$	Drive train inertia (down)	CAD/PEM	$1.914e^{-5}$	$\text{kg m}^2$
$J_{\text{drive,up}}$	Drive train inertia (up)	CAD/PEM	$9.78e^{-6}$	$\text{kg m}^2$
$\kappa_E$	Electrical motor constant	Least square	0.0045	$\text{V}^{-1} \text{s}^{-1}$
$\kappa_M$	Mechanical motor constant	Least square	0.0035	$\text{Nm A}^{-1}$
$d_R$	Motor friction	Least square	$5.2107e^{-7}$	$\text{Nm s}$
$R_\Omega$	Resistance	Least square	1.3811	Ω
$i_{\text{gear}}$	Gear ratio	Measured	1.5	—
$\eta_{\text{gear}}$	Gear efficiency	Measured/PEM	0.84	—
$W_{\text{hub}}$	Drag force on the fuselage	Hover/PEM	0.0108	N
$T_{f,\text{dw}}$	Following time upper rotor	PEM	0.001	s
$T_{f,\text{up}}$	Following time upper rotor	PEM	0.16	s
$l_{\text{dw}}$	Linkage factor upper rotor	PEM	0.41	—
$l_{\text{up}}$	Linkage factor lower rotor	PEM	0.83	—

## References

1. Prouty, R.W.: Helicopter Performance, Stability, and Control. PWS Engineering, Boston (1986)
2. Mettler, B.: Identification Modeling and Characteristics of Miniature Rotorcraft. Kluwer Academic, Dordrecht (2003)
3. Dzul, A., Hamel, T., Lozano, R.: Modeling and nonlinear control for a coaxial helicopter. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 6 (2002)

4. Chen, L., McKerrow, P.: Modelling the lama coaxial helicopter. In: Australasian Conference on Robotics & Automation, Brisbane, Australia (2007)
5. Gerig, M.: Modeling, guidance, and control of aerobatics maneuvers of an autonomous helicopter. Ph.D. thesis, ETH Zurich (2008)
6. Watkinson, J.: The Art of the Helicopter. Elsevier, Amsterdam (2004)
7. Schafroth, D., Bermes, C., Bouabdallah, S., Siegwart, R.: Micro Helicopter Steering: Review and Design for the muFly Project. In: IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Beijing, China (2008)
8. Bramwell, A.: Helicopter Dynamics, 2nd edn. Butterworth Heinemann, Boston (2001)
9. Leishman, J.: Helicopter Aerodynamics, 2nd edn. Cambridge University Press, Cambridge (2006)
10. Mueller, G., Ponick, B.: Elektrische Maschinen Grundlagen elektrischer Maschinen. VCH, Weinheim (2006)
11. Schafroth, D., Bouabdallah, S., Bermes, C., Siegwart, R.: From the test benches to the first prototype of the muFly micro helicopter. *J. Intell. Robot. Syst.* **54**(1–3), 245–260 (2008)
12. CEDRAT: CEDRAT technologies. <http://www.cedrat.com> (2009)
13. Schwarz, H.R., Kaeckler, N.: Numerische Mathematik. Stuttgart; Leipzig. Teubner, Wiesbaden (2004)
14. Moeckli, M.: Guidance and control for aerobatic maneuvers of an unmanned airplane. Ph.D. dissertation, ETH Zurich (2006)

## Vision-based Position Control of a Two-rotor VTOL miniUAV

E. Rondon · S. Salazar · J. Escareno · R. Lozano

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 16 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** In this paper we address the stabilization of the attitude and position of a birotor miniUAV to perform autonomous flight. For this purpose, we have implemented a Kalman-based sensor fusion between inertial sensors (gyros-accelerometers) and the optical flow (OF) provided by the vehicle. This fusion algorithm extracts the translational-OF (TOF) component and discriminates the rotational OF (ROF). The aircraft's position is obtained through an object detection algorithm (centroid tracking). Newton-Euler motion equations were used to deduce the mathematical model of the vehicle. In terms of control we have employed a saturated-based control to stabilize the state of the aircraft around the origin. Experimental autonomous flight was successfully achieved, which validates the sensing strategy as well as the embedded control law.

**Keywords** Optical flow · Nonlinear control · VTOL aircraft

### 1 Introduction

The development of mini UAVs and micro UAVs has recently motivated new research areas in different disciplines, such as electronics, mechanics, aeronautics,

---

E. Rondon (✉) · R. Lozano  
Université de Technologie de Compiègne, Heudiasyc, UMR CNRS 6599, Compiègne, France  
e-mail: rondoned@hds.utc.fr

R. Lozano  
e-mail: rlozano@hds.utc.fr

S. Salazar · J. Escareno  
Laboratoire Franco Mexicain d'Informatique et Automatique,  
UMI-CNRS 3175, Mexico, DF, Mexico

J. Escareno  
e-mail: jescareno@ctrl.cinvestav.mx

automatic control just to mention a few. Such vehicles have great potential for search and rescue missions, homeland security, real-time forest fire monitoring, surveillance of sensitive areas (borders, ports, oil pipelines), remote sensing, etc. Accurate navigation and efficient control of the vehicle are essential to successfully accomplish such missions. Furthermore, small UAVs impose weight and size constraints which exclude the use of heavy sensors. Light sensors are usually less accurate and therefore suitable estimation and control algorithms should be developed, since the performance of the closed loop control system will be as good as the quality of the feedback sensor signal.

Inertial sensors, gyroscopes and accelerometers, provide rotational velocity and linear acceleration respectively. The numerical integration of these signals can provide the attitude, translational position and velocity of the vehicle. These measurements are valid only for a short period of time, since the signal drifts due to inherent noise and bias. Thus, for a reliable attitude measurement, a fusion between accelerometers and gyroscopes is needed. Concerning the position, the GPS is fused with inertial sensors (INS) to provide periodic corrections, in order to obtain a more accurate (drift-less) position measurement. However, the GPS signal is not available for indoor locations or certain urban zones (urban canyon problem). To overcome this location estimation issue, an alternative strategy is the combination of artificial vision with inertial sensors to extract a valid position measurement of the aerial robot.

Previous works have been developed using visual-based sensing estimation.

In [2] a visual control system is presented which aims at maintaining a constant optical flow (OF). The OF is obtained from a camera which is vertically fixed to the platform to avoid the acquisition of OF produced by rotational motion of the flying engine. In [3] the author established a relation between the OF ( $\frac{\text{pixel}}{\text{s}}$ ) and the actual rotation ( $\frac{\text{degrees}}{\text{s}}$ ), through the knowledge of the corresponding inter-pixel angle. After a calibration, the cancelation of the rotational components is obtained by subtracting the gyroscopic value from the OF measurement. Other approach was proposed in [4], where the rotational optical flow (ROF) is canceled by counter-rotation of the OF sensors. The flying object orientation is then stabilized and the visual system detects pure translational OF (TOF). In [5] the authors developed a methodology to compensate the ROF. Using the angular rate they computed the rotational component for each pixel, where an estimated of the OF has been acquired. The rotational component is then subtracted from the measured OF, and an estimation of the angular velocity is finally obtained. Although this approach seems natural, the noise in the measurements is not taken into account. The fusion of inertial and visual information has been addressed by several authors, for different applications. In [6] an eight-rotor mini UAV employs an optical flow feedback for translational motion damping. This particular design obeys to the idea of enhancing the attitude stability by decoupling the attitude and position control and leading to a fully-actuated dynamics. The pitch and roll angular positions are first stabilized around the origin so that the OF measurement is a good estimate of the translational velocity. In [7] the authors have used inertial sensors to improve the OF measurement for obstacle detection. In [8] a gyroscope sensor is used to decompose rotational and translational motion from the visual scene in order to increase the accuracy of the 3D shape reconstruction. Another technique to extract 3D motion from estimated velocity vectors from a number of points of the image

is the Egomotion [9]. In this method the rotational components are eliminated via algebraic operations and the translational components are estimated using a least squares method. However, the algorithms for such technique are computationally expensive.

Most of small UAVs have underactuated dynamics, i.e. the vehicle's horizontal motion is generated by the pitch and roll angular displacements. Hence, the measured OF is produced by both the translational and the rotational motions. An effective visual-based control of the UAV's position requires the use of inertial sensors (accelerometer and gyroscopes) to eliminate the ROF component. Yet another alternative to measure only the translational motion is to use gyrostabilized cameras. However such cameras are expensive and heavy.

In the present paper we consider a platform composed of a two-rotor mini UAV and two external cameras. An alternative strategy is presented to estimate the translational position and velocity of the mini UAV. The flying engine has an inertial measurement unit (IMU) to measure the orientation. The position and velocity are obtained using two external cameras which provide frontal and lateral views of the vehicle. A Kalman filter is proposed to fuse the visual information with the IMU measurement in order to remove the bias due to ROF, depth and sensors noise. The resulting Kalman filter estimate is incorporated into the control algorithm to stabilize the 6-DOF nonlinear dynamics (attitude and position) of the aerial robot. The proposed control strategy has been successfully tested in the real platform.

The paper is organized as follows: Section 2 describes the sensing system and the fusion algorithm. In Section 3, the dynamical model of the vehicle is obtained. Section 4 presents the saturation-based control algorithm to stabilize the flight dynamics. Section 5 describes the experimental setup and shows the results obtained during the autonomous hover flight of the vehicle.

## 2 Position and Velocity Estimation

### 2.1 Inertial Sensors

The output of a gyroscope can be decomposed as follows

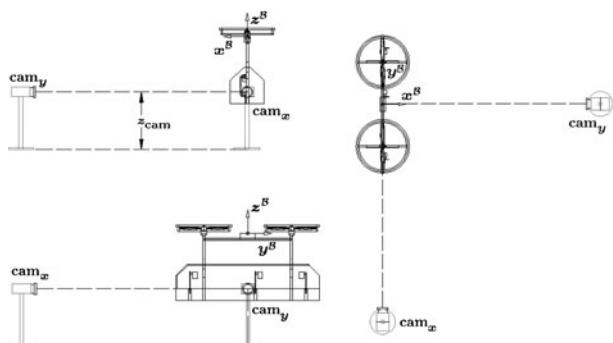
$$\omega_o = \omega_i + \delta\omega_b + \delta\omega_{lo} + \delta\omega_r \quad (1)$$

where  $\omega_o$  is the measured gyroscope output,  $\omega_i$  is the real value,  $\delta\omega_b$  is a known constant bias,  $\delta\omega_{lo}$  is a temperature dependent bias, and  $\delta\omega_r$  is a random-walk process.

### 2.2 Visual Sensors

The goal of our approach is to estimate the position of an aircraft at each image obtained from the two cameras. These cameras are arranged orthogonally, the frontal camera ( $cam_y$ ) and lateral camera ( $cam_x$ ) measure the displacement of the vehicle in the  $Y^BZ^B$  and  $X^BZ^B$  planes respectively (see Figs. 1 and 2).

**Fig. 1** Visual perspective of the cameras: **a** XZ-plane (left-top) **b** YZ-plane (left-bottom) **c** XY-plane (Right)

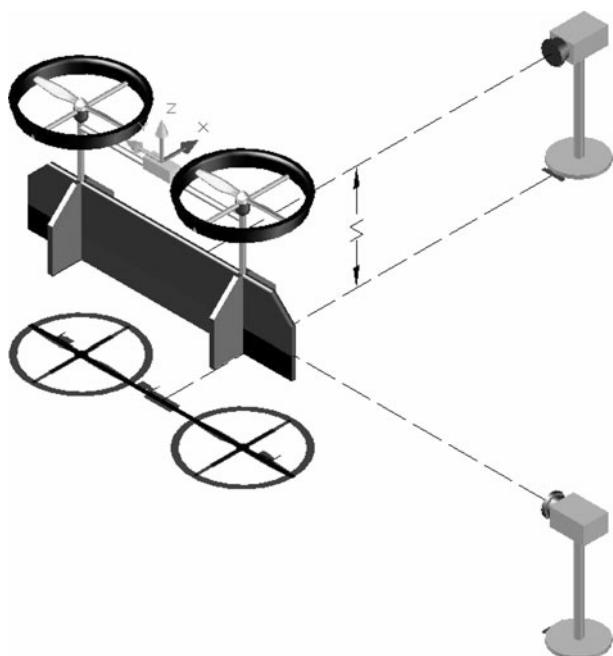


### 2.2.1 Position

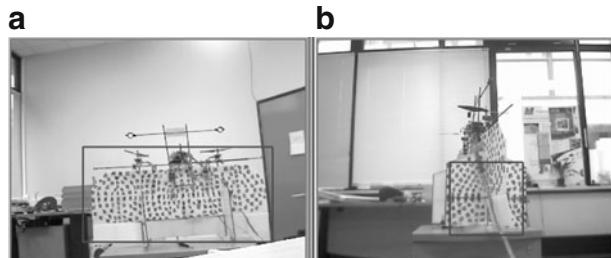
In order to increase the contrast in the image of the camera, the fuselage of the aerial robot is covered with points (see Fig. 3). Notice that for each camera, every point on the aircraft surface has a corresponding projected point on the image plane. Hence, the position of the aircraft can be estimated using the centroid or barycenter of these points. In fact the barycenter's position can be expressed as the initial position plus the displacement during the interval  $[t, t + \Delta t]$ .

$$\begin{aligned} x_{brc}(t + \Delta t) &= x_{brc}(t) + \int_t^{t+\Delta t} V_x dt \\ y_{brc}(t + \Delta t) &= y_{brc}(t) + \int_t^{t+\Delta t} V_y dt \end{aligned} \quad (2)$$

**Fig. 2** Experimental setup



**Fig. 3** Visual perspective of the cameras: **a** Frontal camera (*left*) **b** Lateral camera (*right*)



The method used to detect the object was proposed by Viola and Jones [10], this method is a machine learning approach capable of processing images extremely rapidly. The main motivation for using this approach is the high detection rate which makes this algorithm suitable for real time applications. Before implementing the algorithm a learning stage is needed. For this purpose a “training” set of 1000 images of the aircraft body frame were used, as well as 80 images of the surrounding environment. The low number of environment images is justified by the fact that the environment remains fixed. When the environment may change, a larger number of samples of positive (aircraft) and negative (environment) images are needed to render the detector more robust. The samples were chosen so that the detector is robust with respect to (w.r.t.) slight variations of light intensity, cameras parameters and environment.

### 2.2.2 Optical Flow (OF)

The OF is widely used to compute an approximation of motion field through time-varying image intensity. Among the diverse approaches to compute OF [1], we have implemented the Lucas-Kanade pyramidal algorithm [11] in combination with a texture-detecting algorithm. This method provides an accurate estimation of the motion field. The points are placed on the vehicle’s surface with a specific pattern (see Fig. 3) enabling the computation of the OF. The texture detector also neglects the neighborhood, where the motion field cannot be accurately determined.

The point  $p(x_i, y_i)$  in the image plane of the camera has a corresponding point  $P(X, Y, Z)$  in the real world, which is obtained with a perspective projection of a pinhole camera in the following way:

$$\begin{aligned} x_i &= f \frac{X}{Z} \\ y_i &= f \frac{Y}{Z} \end{aligned} \quad (3)$$

where  $(x_i, y_i)$  represents the point in the image plane,  $f$  is the focal length of the camera and  $Z$  is the distance between the camera and the aircraft. Differentiating both sides of (3) leads us to the familiar equations of OF [12]

$$\begin{bmatrix} OF_{x_i} \\ OF_{y_i} \end{bmatrix} = T_{OF} + R_{OF} \quad (4)$$

with

$$T_{OF} = \frac{1}{Z} \begin{bmatrix} -f & 0 & x_i \\ 0 & -f & y_i \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \quad (5)$$

and

$$R_{OF} = \begin{bmatrix} \frac{x_i y_i}{f} & -\left(f + \frac{x_i^2}{f}\right) & y_i \\ \left(f + \frac{y_i^2}{f}\right) & -\frac{x_i y_i}{f} & -x_i \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (6)$$

where  $OF_j$  is the OF component in the coordinate  $j$  of the point  $p_i$ ,  $V_k$  and  $\omega_k$  are respectively the translational and rotational rates of the body in the coordinate  $k$ . In practice, the origin of the image coordinate frame is not always at the principal point and the scaling for each image axis is different. Therefore, the image coordinates need an additional transformation matrix ( $K$ ), which represents the intrinsic parameters of the camera. Finally, after calibration and considering the weak perspective projection we have:

$$\begin{aligned} x_i &= \frac{s_x f}{Z} X_i + x_0 \\ y_i &= \frac{s_y f}{Z} Y_i + y_0 \end{aligned} \quad (7)$$

where  $s_x$  and  $s_y$  are the scaling factors which take into account the pixel size, and  $(x_0, y_0)$  is the origin of the image coordinate system. This new transformation provides the true equations relating the OF with the 3D motion of the aircraft.

Once the object is detected (see Fig. 3), the visual system is able to sense the 6-DOF of the aircraft. However, we are only interested in translational velocities. Provided that the points and the vehicle share the same motion, then, we can express the mean of the OF as:

$$\begin{aligned} \frac{\sum OF_{x_i}}{n} &= \frac{\sum T_{OFx_i}}{n} + \frac{\sum R_{OFx_i}}{n} \\ \frac{\sum OF_{y_i}}{n} &= \frac{\sum T_{OFy_i}}{n} + \frac{\sum R_{OFy_i}}{n} \end{aligned} \quad (8)$$

with

$$\begin{aligned} \frac{\sum T_{OFx_i}}{n} &= -\frac{V_x}{Z} - \frac{\sum x_i}{n} \frac{V_z}{Z} \\ \frac{\sum R_{OFx_i}}{n} &= \frac{\sum x_i y_i}{n} \omega_x - \frac{\sum (1 + x_i^2)}{n} \omega_y + \frac{\sum y_i}{n} \omega_z \\ \frac{\sum T_{OFy_i}}{n} &= -\frac{V_y}{Z} - \frac{\sum y_i}{n} \frac{V_z}{Z} \\ \frac{\sum R_{OFy_i}}{n} &= \frac{\sum (1 + y_i^2)}{n} \omega_x - \frac{\sum x_i y_i}{n} \omega_y - \frac{\sum x_i}{n} \omega_z \end{aligned} \quad (9)$$

where  $n$  is the number of points having an OF estimation. It is assumed that a rich textured-features selector algorithm is used and that the search zone is on the vehicle's surface.

The average of the measured OF in the image coordinate system can be represented as

$$\begin{aligned}\dot{x} &= -\frac{V_x}{Z} - K_x^x \frac{V_z}{Z} + K_{xy}^x \omega_x - K_{x^2}^x \omega_y + K_y^x \omega_z \\ \dot{y} &= -\frac{V_y}{Z} - K_y^y \frac{V_z}{Z} + K_{y^2}^y \omega_x - K_{xy}^y \omega_y - K_x^y \omega_z\end{aligned}\quad (10)$$

where  $\frac{V_x}{Z}$  and  $\frac{V_y}{Z}$  are the relative velocities of the aircraft in the camera coordinate system,  $\frac{V_z}{Z}$  is the inverse of the Time-To-Contact, known as the relative depth, and  $K_j^i$  is a constant scale factor depending on the intrinsic parameters of the camera. The rotations about the optical axis produce linear contributions to the OF, while both translation and rotation lead to non linear contributions. For rotations, the constant scale factor can be computed as a function of the intrinsic parameters of the camera and the output vectors of the rich textured-features detector. Once the scale factor is computed, the elimination of the rotational OF components is performed by the Kalman filter.

### 2.3 Kalman-based Sensor Fusion

The goal is to remove the ROF obtained during the motion of the vehicle. For this purpose inertial sensors are used in the estimation algorithm. The OF measurement involves uncertainties so that the TOF estimation problem can be set in the Kalman filter framework [13].

Given that the rotational components of the optical flow do not depend on the distance between the aircraft and the camera, it is possible to use the vehicle state to estimate and compensate the contribution of the rotations. The remaining flow will be due to the translational velocity of the aircraft.

The following linear discrete-time state-space representation can be obtained from (2) and (10)

$$\begin{bmatrix} X \\ V \\ W \end{bmatrix}_k = \begin{bmatrix} I & T & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} X \\ V \\ W \end{bmatrix}_{k-1} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_k \quad (11)$$

where  $X$  is the position,  $V$  is the velocity and  $W$  is the angular velocity of the vehicle.  $T$  represents the sampling period and  $v_i$  represents the noise. The measured outputs are the computed barycenters  $X_V$ , the OF measurement  $V_{OF}$ , and the measured angular velocity  $W_{IMU}$ . The outputs are related to the state in (11) as

$$\begin{bmatrix} X_V \\ V_{OF} \\ W_{IMU} \end{bmatrix}_k = \begin{bmatrix} I & 0 & 0 \\ 0 & I & K_R^T \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} X \\ V \\ W \end{bmatrix}_k + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_k \quad (12)$$

where the noises in (11) and (12) are:

- $v = [v_1, v_2, v_3]^T$  is assumed to be a white noise with zero mean and known constant covariance matrix  $Q$ .
- $\nu = [\nu_1, \nu_2, \nu_3]^T$  is assumed to be a white noise with zero mean and known constant covariance matrix  $R$ .
- $v$  and  $\nu$  are assumed to be uncorrelated noises.

The embedded Kalman filter implemented on the aircraft follows the correction-prediction sequence. The correction equation is the first step of the filter and given by

$$\begin{bmatrix} \hat{X} \\ \hat{V} \\ \hat{W} \end{bmatrix}_{k|k} = \begin{bmatrix} \hat{X} \\ \hat{V} \\ \hat{W} \end{bmatrix}_{k|k-1} + K \begin{bmatrix} X_V - \hat{X}_{k|k-1} \\ V_{OF} - \hat{V}_{k|k-1} - K_R^T \hat{W}_{k|k-1} \\ W_{IMU} - \hat{W}_{k|k-1} \end{bmatrix} \quad (13)$$

with

$$K = \begin{bmatrix} K_{x|x} & K_{x|v} & K_{x|i} \\ K_{v|x} & K_{v|v} & K_{v|i} \\ K_{i|x} & K_{i|v} & K_{i|i} \end{bmatrix} \quad (14)$$

The correction of the covariance matrix of the estimated error is computed as follows

$$P_{k|k} = (I - KC)P_{k|k-1}(I - KC)^T + KRK^T \quad (15)$$

with

$$(I - KC) = \begin{bmatrix} I - K_{x|x} & -K_{x|v} & -(K_R^T K_{x|v} + K_{x|i}) \\ -K_{v|x} & I - K_{v|v} & -(K_R^T K_{v|v} + K_{v|i}) \\ -K_{i|x} & -K_{i|v} & I - (K_R^T K_{i|v} + K_{i|i}) \end{bmatrix} \quad (16)$$

$$KRK^T = \begin{bmatrix} K_{x|x} R_x & K_{x|v} R_v & K_{x|i} R_i \\ K_{v|x} R_x & K_{v|v} R_v & K_{v|i} R_i \\ K_{i|x} R_x & K_{i|v} R_v & K_{i|i} R_i \end{bmatrix} \begin{bmatrix} K_{x|x}^T & K_{v|x}^T & K_{i|x}^T \\ K_{x|v}^T & K_{v|v}^T & K_{i|v}^T \\ K_{x|i}^T & K_{v|i}^T & K_{i|i}^T \end{bmatrix} \quad (17)$$

The prediction of the state is obtained from

$$\begin{bmatrix} \hat{X} \\ \hat{V} \\ \hat{W} \end{bmatrix}_{k+1|k} = \begin{bmatrix} I & T & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{V} \\ \hat{W} \end{bmatrix}_{k|k} \quad (18)$$

The estimated error is predicted as follows.

$$P_{k+1|k} = \begin{bmatrix} I & T & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} P_{k|k} \begin{bmatrix} I & 0 & 0 \\ T^t & I & 0 \\ 0 & 0 & I \end{bmatrix} + \begin{bmatrix} Q_x & 0 & 0 \\ 0 & Q_v & 0 \\ 0 & 0 & Q_i \end{bmatrix} \quad (19)$$

with

$$P = \begin{bmatrix} P_{x|x} & P_{x|v} & P_{x|i} \\ P_{v|x} & P_{v|v} & P_{v|i} \\ P_{i|x} & P_{i|v} & P_{i|i} \end{bmatrix} \quad (20)$$

The Kalman gain is directly given by

$$K = P_{k+1|k} C^T (C P_{k+1|k} C^T + R)^{-1} \quad (21)$$

where

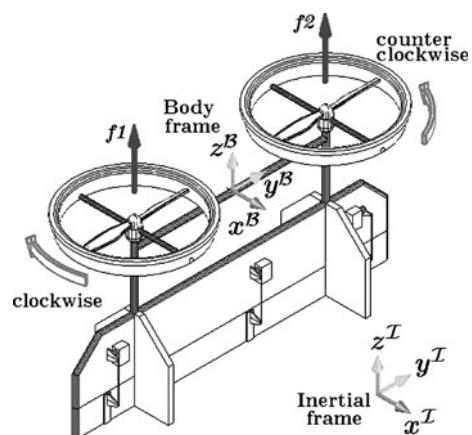
$$C P_{k+1|k} C^T = \begin{bmatrix} I & 0 & 0 \\ 0 & I & K_R^T \\ 0 & 0 & I \end{bmatrix} P_{k+1|k} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & (K_R^T)^T & I \end{bmatrix} \quad (22)$$

### 3 Dynamical Model

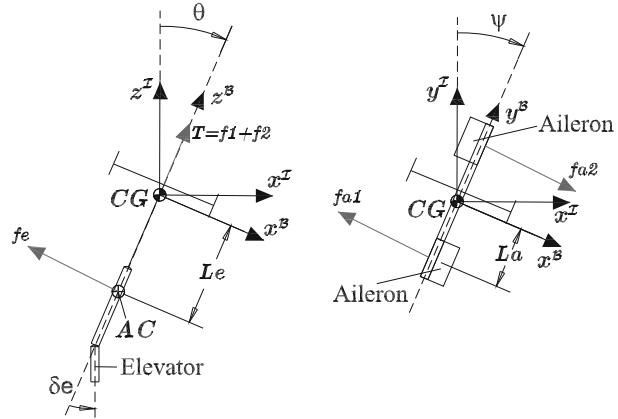
The vehicle used as proof-of-concept platform is mechanically simpler than classical helicopters, it does not have swashplate and has fixed-pitch blades (Figs. 4 and 5). The force  $f_i$  produced by motor  $i$  is proportional to the square of the angular speed, that is  $f_i = k\omega_i^2$ . The left and right motors rotate clockwise and counterclockwise respectively. Gyroscopic effects and aerodynamic torques tend to cancel in trimmed flight. The altitude relies on the vertical thrust, which is the sum of the forces provided by each motor. The rolling torque is obtained by differential motor thrust. The control surfaces are within the propellers airflow, which in combination with a deflection provides a torque. Thus the pitching torque depends on the elevator deflection ( $\delta_e$ ) while the yawing torque is produced by the differential deflection of the ailerons ( $\delta_a$ ).

Let us denote  $\mathcal{I} = \{i, j, k\}$  as the inertial frame,  $\mathcal{B} = \{i_b, j_b, k_b\}$  as the body frame attached to the aircraft (Fig. 4) and  $q_e = [x \ y \ z \ \phi \ \theta \ \psi]^T = [\xi \ \eta]^T$  as the generalized coordinates which describe the vehicle's position and attitude.  $\xi \in \mathbb{R}^3$ , denotes the position of the vehicle's center of gravity, relative to the inertial frame, and  $\eta \in \mathbb{R}^3$  are the three Euler angles (roll, pitch and yaw), which represent the aircraft's orientation.

**Fig. 4** Coordinates systems: inertial and body-fixed frames



**Fig. 5** Forces distribution in the vehicle



The dynamical model is obtained via the Euler-Lagrange formulation. The application of methodology results in the following equations of motion

$$m\ddot{\xi} + mG = \mathcal{R}^{BT} F^B \quad (23)$$

$$\mathbb{I}\ddot{\eta} + \dot{\mathbb{I}}\dot{\eta} - \frac{1}{2} \frac{\partial}{\partial \eta} (\dot{\eta}^T \mathbb{I} \dot{\eta}) = \tau^B \quad (24)$$

where  $m \in \mathbb{R}$  denotes the mass of the vehicle,  $\tau \in \mathbb{R}^3$  denotes the generalized momentum,  $F^B \in \mathbb{R}^3$  is the force which affects the linear motion of the miniUAV,  $G \in \mathbb{R}^3$  denotes gravity acceleration ( $G = -g k_i$ ). The orthonormal transformation matrix  $\mathcal{R}^{BT}$  from the body frame to the inertial frame, with the rotation sequence  $\psi \rightarrow \theta \rightarrow \phi$ , is given as

$$\mathcal{R}^{BT} = \begin{pmatrix} c_\theta c_\psi & -s_\psi c_\theta & s_\theta \\ s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi - s_\psi s_\theta s_\phi & -c_\theta s_\phi \\ s_\psi s_\phi + c_\psi c_\theta c_\phi & c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{pmatrix} \quad (25)$$

where, for convenience, we denote  $\sin(\cdot) \triangleq s_{(\cdot)}$  and  $\cos(\cdot) \triangleq c_{(\cdot)}$ . The term  $\mathbb{I}$  is given by

$$\mathbb{I}(\eta) = \mathcal{W}^T I \mathcal{W} \quad (26)$$

where,  $I \in \mathbb{R}^3$  describes the Inertia tensor matrix, which is diagonal due to the vehicle has two planes of symmetry.  $\mathcal{W}$  is the non-orthonormal transformation that relates the angular velocity vector ( $\Omega$ ) and the Euler rate vector ( $\dot{\eta}_2$ )

$$\Omega = \mathcal{W}\dot{\eta} = \begin{pmatrix} C_\theta C_\psi & -S_\psi & 0 \\ C_\theta S_\psi & C_\psi & 0 \\ S_\theta & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}. \quad (27)$$

Let us define the Coriolis and gyroscopic terms as

$$C(\eta, \dot{\eta}) \dot{\eta} \triangleq \dot{\mathbb{I}}\dot{\eta} - \frac{1}{2} \frac{\partial}{\partial \eta} (\dot{\eta}^T \mathbb{I} \dot{\eta}) \quad (28)$$

Then, the rotational model (24) can be written as

$$\mathbb{I}\ddot{\eta} = \tau^B - C(\eta, \dot{\eta}) \quad (29)$$

where,  $\tau^B$  represent the torque produced by the actuators

$$\tau_\phi = (f_1 - f_2)L_a \quad (30)$$

$$\tau_\theta = f_e L_e \quad (31)$$

$$\tau_\psi = (f_{a1} + f_{a2})L_a \quad (32)$$

where  $l_m$  is the distance from each motor to the center of gravity (CG) while  $l_e$  and  $l_a$  are the distances from the CG to the aerodynamic center of each control surface.

The scalar equations corresponding to the translational dynamic model (23) are

$$\begin{cases} \ddot{x} = \frac{1}{m} Ts_\theta \\ \ddot{y} = -\frac{1}{m} T c_\theta s_\phi \\ \ddot{z} = \frac{1}{m} T c_\theta c_\phi - g \end{cases} \quad (33)$$

with  $T = f_1 + f_2$ . For further control analysis, the following change of input variables is considered

$$\tau^B = \mathbb{I}\tilde{\tau} + C(\eta, \dot{\eta}) \quad (34)$$

with  $\tilde{\tau} = (\tilde{\tau}_\phi, \tilde{\tau}_\theta, \tilde{\tau}_\psi)^T$ . Therefore, the rotational dynamical model (29) is written as

$$\begin{cases} \ddot{\phi} = \tau_\phi \\ \ddot{\theta} = \tau_\theta \\ \ddot{\psi} = \tau_\psi \end{cases} \quad (35)$$

#### 4 Control Strategy

The vehicle's position is computed from two cameras as discussed previously (see Fig. 1). The frontal camera ( $cam_y$ ) provides the feedback to control the altitude  $z$  and the  $y - \phi$  underactuated subsystem, while, the lateral camera ( $cam_x$ ) is devoted to the  $x - \theta$  underactuated system. Thus, the dynamic system (33–35) can be split into the three following subsystems:

$$S_{cam_y} : \begin{cases} \ddot{y} = -\frac{1}{m} F_T \sin \phi \\ \ddot{\phi} = \tilde{\tau}_\phi \\ \ddot{z} = \frac{1}{m} F_T \cos \phi - g \end{cases} \quad (36)$$

$$S_{cam_x} : \begin{cases} \ddot{x} = \frac{1}{m} F_T \sin \theta \\ \ddot{\theta} = \tilde{\tau}_\theta \end{cases} \quad (37)$$

$$S_\psi : \{ \ddot{\psi} = \tilde{\tau}_\psi \quad (38)$$

For simplicity in the control analysis we consider the normalized values of the mass  $m$ , and the inertia moments matrix  $I (I_x, I_y, I_z)$ .

#### 4.1 Frontal Subsystem ( $S_{cam_y}$ )

The control algorithm used for the frontal dynamic subsystem  $S_{cam_y}$ , focuses on the altitude followed by the  $y - \phi$  stabilization. The altitude can be stabilized with a feedback-linearizable input via the thrust  $F_T$ . A suitable choice is

$$F_T = \frac{v_z + g}{\cos(\phi)} \quad (39)$$

where,  $v_z = -a_{z_1}\dot{z} - a_{z_2}(z - z^d)$  with  $a_{z_1}, a_{z_2} > 0$  and  $z_d$  is the desired altitude. From (39) we notice that the controller is valid within  $-\frac{\pi}{2} < \phi < \frac{\pi}{2}$ , which is appropriate for our flight purposes. Using (39) in (36) leads to

$$\ddot{y} = -\tan \phi (v_z + g) \quad (40)$$

Given that the flight of the aircraft evolves close to the vertical, we are able to consider  $\tan \phi \approx \phi$ . Also, note that  $z \rightarrow z^d$  and therefore  $v_z \rightarrow 0$ . As a result the subsystem ( $y - \phi$ ) can be rewritten as

$$\begin{aligned} \ddot{y} &= -g\phi \\ \ddot{\phi} &= \tilde{\tau}_\phi \end{aligned} \quad (41)$$

which is a system composed of four integrators in cascade. To stabilize such system, we have used the following saturation-based control (for details see [14])

$$\tilde{\tau}_\phi = -\sigma_a(z_1) - \sigma_b(z_2) - \sigma_c(z_3) - \sigma_d(z_4) \quad (42)$$

where  $z_1 = \dot{\phi}$ ,  $z_2 = z_1 + \phi$ ,  $z_3 = z_2 + \phi + \dot{y}$ ,  $z_4 = z_3 + \phi + 2\dot{y} + y$  and  $\sigma_\eta$  is a saturation function defined as

$$\sigma_\eta(s) = \begin{cases} \eta & s > \eta \\ s & -\eta \leq s \leq \eta \\ -\eta & s < -\eta \end{cases} \quad (43)$$

#### 4.2 Lateral Subsystem ( $S_{cam_x}$ )

Assuming that the attitude is close to the origin and that the altitude has reached the desired value, the subsystem (37) can be rewritten as

$$\begin{aligned} \ddot{x} &= -g\theta \\ \ddot{\theta} &= \tau_\theta \end{aligned} \quad (44)$$

The same control algorithm is employed to stabilize (44), which is

$$\tau_\theta = -\sigma_a(z_1) - \sigma_b(z_2) - \sigma_c(z_3) - \sigma_d(z_4) \quad (45)$$

where  $z_1 = \dot{\theta}$ ,  $z_2 = z_1 + \theta$ ,  $z_3 = z_2 + \theta + \dot{x}$ ,  $z_4 = z_3 + \theta + 2\dot{x} + x$ .

### 4.3 Heading Subsystem ( $S_\psi$ )

The yaw dynamics is described by the linear double integrator equation:  $\ddot{\psi} = \tau_\psi$ , and whose stabilization can be obtained using the following control input

$$\tilde{\tau}_\psi = -a_{\psi_1}\dot{\psi} - a_{\psi_2}(\psi) \quad (46)$$

with  $a_{\psi_1}, a_{\psi_2} > 0$ .

## 5 Experimental Testbed and Results

The goal of the experiment is to achieve an autonomous stabilized flight in attitude and position for a two-rotor VTOL (Vertical Take-Off and Landing) mini UAV. The visual algorithm provides the position feedback while the embedded inertial system provides the attitude. The system is described next:

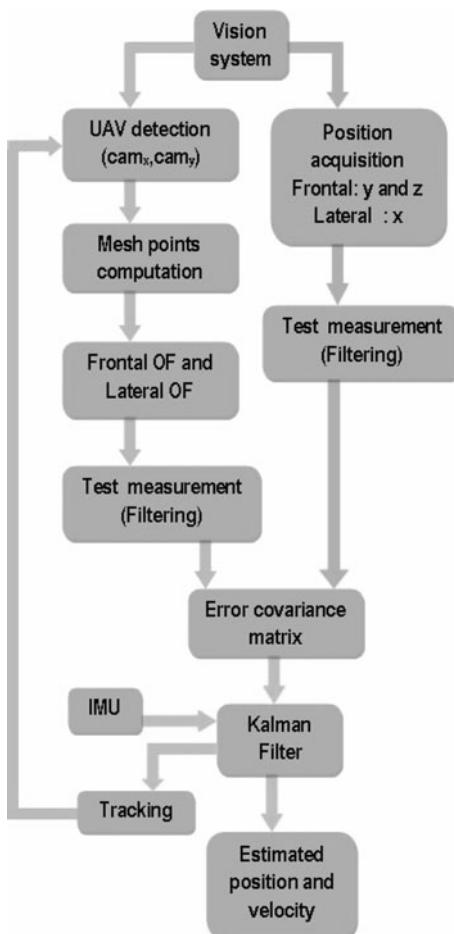
- VTOL prototype:  
The vehicle's fuselage is built of foam and carbon fiber. The UAV's propulsion relies on two brushless motors with two counter rotating blades, while, the control surfaces (elevator and ailerons) are controlled through analog servomotors.
- Onboard system
  - Microcontroller:  
A Rabbit 3400 manages the position and attitude feedback signals. It processes the control algorithm to send the PWM signals to the power interface. The RC-PPM (Pulse Position Modulated) signal from the radio, is captured and decoded so that the user can introduce an external input. Indeed, the position feedback is triggered by an external RC signal.
  - Inertial sensors:  
A Microstrain 3DM-GXI is employed to sense the attitude of the vehicle (angular rate and angular position).
- Outboard system:  
The visual system is implemented outboard on a PC. The vision algorithms provide the centroid detection (position) and optical flow (velocity), which are sent to the aircraft via modem Xbee at 38400 bauds.

Figure 6 shows the overall position and velocity estimation strategy.

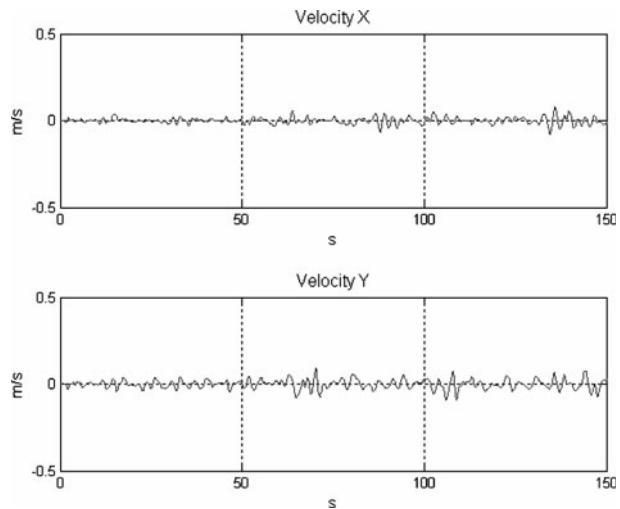
### 5.1 Experimental Results

The systems was tested in real time flight. In the experiments the vision algorithm was running at 14 Hz, which is sufficient to achieve position stabilisation. Figures 7 and 8 show the performance of the visual system and the embedded control law. In Fig. 8, the reference frame is translated to the chosen point where the rotorcraft is stabilized.

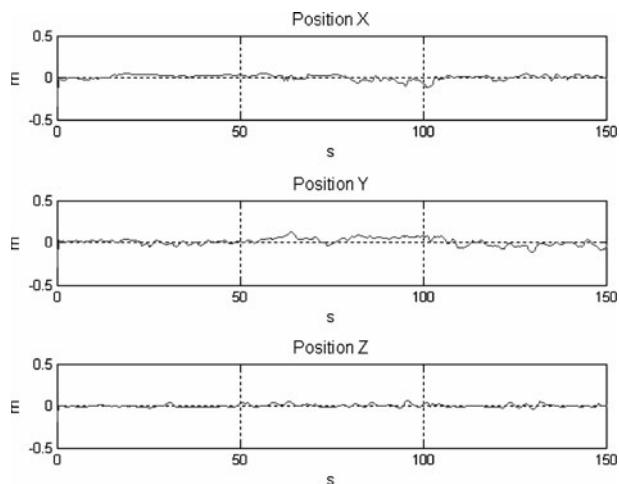
**Fig. 6** Flow diagram of the position and velocity estimation



**Fig. 7** Experimental results: velocity of the aircraft



**Fig. 8** Experimental results: position of the aircraft



## 6 Concluding Remarks

A strategy for autonomous hovering of a mini UAV has been presented using the combination of visual and inertial information. An embedded IMU has been used to obtain the orientation and angular velocity of the vehicle. The aircraft position has been computed using two external cameras arranged orthogonally to provide frontal and lateral views of the vehicle. The Viola-Jones object detection algorithm was used to compute the centroid of a pattern of points on the surface of the vehicle. This detection method is also promising for embedded applications (strapdown cameras) since the algorithm is tuned before the flight using image samples of the surrounding environment.

The position and velocity estimation problem has been set in the Kalman filter framework allowing the fusion of inertial measurements and visual information. A control strategy based on saturation functions has been proposed to stabilize the aircraft at hovering flight. The proposed methodology has been implemented in a real platform composed of a two-rotor VTOL and two external cameras. The vehicle has performed a successful fully autonomous flight during the experiments. The experimental tests validated the proposed estimation algorithm based on the fusion of inertial and vision information as well as the embedded control law.

## References

1. Barron, J., Fleet, D., Beauchemin, S.: Performance of optical flow techniques. *Int. J. Comput. Vis.* **12**(1), 43–77 (1994)
2. Ruffier, F., Franceschini, N.: Optical flow regulation: the key to aircraft automatic guidance. *Robot. Auton. Syst.*, Elsevier **50**, 177–194 (2005)
3. Zufferey, J.C., Floreano, D.: Toward 30-gram autonomous indoor aircraft: vision-based obstacle avoidance and altitude control. In: IEEE International Conference on Robotics and Automation. Barcelona, Spain (2005)
4. Serres, J., Ruffier, F., Viollet, S., Franceschini, N.: Toward optical flow regulation for wall-following and centering behaviours. *Int. J. Adv. Robot. Syst.* **3**(2), 147–154 (2006)

5. Muratet, L., Doncieux, S., Brière, Y., Meyer, J.A.: A contribution to vision-based autonomous helicopter flight in urban environments. *Robot. Auton. Syst.* **50**(4), 195–209 (2005)
6. Romero, H., Salazar, S., Sanchez, A., Lozano, R.: A new configuration having eight rotors: dynamical model and real time control. In: IEEE International Conference on Decision and Control. New Orleans, USA (2007)
7. Bhanu, B., Roberts, B., Ming, J.: Inertial navigation sensor integrated motion analysis for obstacle detection. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 954–959 (1990)
8. Mukai, T., Ohnishi, N.: The recovery of object shape and camera motion using a sensing system with a video camera and a gyro sensor. In: Internation Conference of Computer Vision, pp. 411–417. Kerkyra, Crece (1999)
9. Tian, T.Y., Tomasi, C., Heeger, D.J.: Comparison of approaches to egomotion computation. In: Computer Vision and Pattern Recognition, pp. 315–320 (1996)
10. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, pp. 511–518 (2001)
11. Bouguet, J.Y.: Pyramidal implementation of the Lucas Kanade feature tracker. Technical report, Intel Corporation, Microprocessor Research Lab. (1999)
12. Waxman, A.M., Duncan, J.: Binocular image flows. In: Proceedings of Workshop on Motion: Representation and Analysis, pp. 31–38 (1986)
13. Kalman, R.: Contributions to the theory of optimal control. *Bol. Soc. Mat. Mex.* **5**(1), 102–119 (1960)
14. Lozano, R., Castillo, P., Dzul, A.: Stabilization of the PVTOL: real-time application to a mini-aircraft. *Int. J. Control* **77**(8), 735–740 (2004)

# A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance

C. Goerzen · Z. Kong · B. Mettler

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 17 November 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** A fundamental aspect of autonomous vehicle guidance is planning trajectories. Historically, two fields have contributed to trajectory or motion planning methods: robotics and dynamics and control. The former typically have a stronger focus on computational issues and real-time robot control, while the latter emphasize the dynamic behavior and more specific aspects of trajectory performance. Guidance for Unmanned Aerial Vehicles (UAVs), including fixed- and rotary-wing aircraft, involves significant differences from most traditionally defined mobile and manipulator robots. Qualities characteristic to UAVs include non-trivial dynamics, three-dimensional environments, disturbed operating conditions, and high levels of uncertainty in state knowledge. Otherwise, UAV guidance shares qualities with typical robotic motion planning problems, including partial knowledge of the environment and tasks that can range from basic goal interception, which can be precisely specified, to more general tasks like surveillance and reconnaissance, which are harder to specify. These basic planning problems involve continual interaction with the environment. The purpose of this paper is to provide an overview of existing motion planning algorithms while adding perspectives and practical examples from UAV guidance approaches.

**Keywords** Autonomous · UAV · Guidance · Trajectory · Motion planning · Optimization · Heuristics · Complexity · Algorithm

---

This research was completed under grants NNX08A134A (San Jose State University Research Foundation) and NNX07AN31A (University of Minnesota) as part of the US Army Aeroflightdynamics Directorate (RDECOM) flight control program.

C. Goerzen  
NASA Ames Research Center, San Jose State University Research Foundation,  
Moffett Field, CA 94035, USA  
e-mail: chad.goerzen@us.army.mil

Z. Kong · B. Mettler (✉)  
Department of Aerospace Engineering and Mechanics,  
University of Minnesota, Minneapolis, USA  
e-mail: mettler@aem.umn.edu

## 1 Introduction

In the last decade, significant changes have occurred in the field of vehicle motion planning, and for UAVs in particular. UAV motion planning is especially difficult due to several complexities not considered by earlier planning strategies: the increased importance of differential constraints, atmospheric turbulence which makes it impossible to follow a pre-computed plan precisely, uncertainty in the vehicle state, and limited knowledge about the environment due to limited sensor capabilities. These differences have motivated the increased use of feedback and other control engineering techniques for motion planning. The lack of exact algorithms for these problems and difficulty inherent in characterizing approximation algorithms makes it impractical to determine algorithm time complexity, completeness, and even soundness. This gap has not yet been addressed by statistical characterization of experimental performance of algorithms and benchmarking. Because of this overall lack of knowledge, it is difficult to design a guidance system, let alone choose the algorithm.

Throughout this paper we keep in mind some of the general characteristics and requirements pertaining to UAVs. A UAV is typically modeled as having velocity and acceleration constraints (and potentially the higher-order differential constraints associated with the equations of motion), and the objective is to guide the vehicle towards a goal through an obstacle field. A UAV guidance problem is typically characterized by a three-dimensional problem space, limited information about the environment, on-board sensors with limited range, speed and acceleration constraints, and uncertainty in vehicle state and sensor data.

In order to be able to compare algorithms on a rigorous basis it is necessary to establish their degree of soundness, completeness, optimality, precision, and computational complexity. Once accomplished, this work will help in choosing an algorithm based on the requirements called for by a particular application. This brings up the dual of the problem of understanding the algorithm, which is that of understanding the requirements of a particular application. In the end, these are two sides of one larger problem. Before understanding the larger problem of UAV guidance we start with a comprehensive perspective of the field and a general understanding of the UAV guidance problem. This review covers recent developments in the robotics and aerospace guidance and control fields. It summarizes as comprehensively as possible what has been claimed or proven regarding these algorithms. It does not attempt to fill in characteristics for algorithms that have not been published. There is a great deal of work remaining to be done in order to characterize existing algorithms, in particular those capable of handling differential constraints.

The paper is organized as follows. The next section gives an overview of the fundamental issues in vehicle motion planning, including key terminology, definitions and concepts, problem types, problem metrics, and algorithm performance criteria. Section 3 provides a survey of recent work focusing on path planning without differential constraints. These include: roadmap methods, exact cell decomposition, approximate cell decomposition, potential field methods, probabilistic approaches, and weighted region problems. Section 4 provides a survey of trajectory planning with differential constraints. These include: state-space sampling methods, minimum distance discrete path followed by trajectory forming, mathematical programming, potential field methods, and solutions given uncertainty. We conclude the paper with a brief discussion based on the broad perspective offered by the survey.

## 2 Background and Overview

The main focus of this article is to survey algorithms that solve the problem of trajectory planning of a dynamics-constrained vehicle through an environment with obstacles. For many UAV applications, a point vehicle representation usually suffices as a slightly conservative assumption that vastly simplifies the problem. In addition, the problem of finding a trajectory that minimizes some cost functional is of interest. Given that an exact solution of trajectory planning in an obstacle field is a variational calculus problem with analytic solutions that are computable only for some of the simplest cases, all the algorithms used to solve this problem in three-dimensional space are approximation algorithms. Many of the algorithms designed to solve the dynamics-constrained problem rely on a decomposition approach, first solving a path planning problem, applying smoothing constraints, forming a trajectory that conforms to the path, and using a control loop to follow this trajectory [70]. Since any of the traditional dynamics-unconstrained algorithms found in the field of robotics may be used for the path planning stage in this approach, these are covered briefly in this review.

### 2.1 Overall Problem Description

Vehicle motion planning is a special case of the general motion planning problem, which in general is very difficult to solve, especially as the number of degrees of freedom increases. In a typical UAV application, the vehicle operates in three-dimensional space, has two to four degrees of freedom, and has differential constraints, including limited speed and maximum acceleration. The resulting problem space has at least five to 12 dimensions, associated with the equations of motion and involving constraints on states and input variables. There does not exist an algorithm that provides an exact analytic solution to such a problem. Indeed, even state-of-the-art approximation algorithms operating on a three-dimensional subspace of this problem space are difficult to compute in real time. Furthermore, several simplifications and sub-cases of the general problem have been proven to be unsolvable in polynomial time [18]. Approximation algorithms are possible, and often rely on exact solutions to simplified sub-problems.

### 2.2 Previous Surveys

There are several important works covering the field of motion planning, most of them in the form of textbooks. The book *Complexity of Robot Motion Planning* [18] proves bounds on several motion planning problems that are still relevant. The book by Schwartz and Yap [98] contains another review. The classic textbook *Robot Motion Planning* [66] gives detailed explanations of the algorithms used until 1990, and remains one of the best sources of information on algorithms for solving path planning problems on polygonal obstacle field representations. Another textbook, [39] titled *Motion Planning in Dynamic Environments*, systematically covers the dynamic environment problem and describes computational bounds for several problems. Hwang and Ahuja [48] is a comprehensive survey of the field of motion planning from the same year.

More recently, Barto et al. [4] includes a comprehensive overview of optimal control type approaches, especially dynamic programming, in its first 10 pages. Tarabanis [64] discusses closely related problems of sensor based planning, and [109] reviews probabilistic approaches. The most significant recent work is [70], a textbook titled *Planning Algorithms*. Although not designed to be a comprehensive survey, this book provides broad coverage of the field of motion planning algorithms, and has a strong focus on vehicle motion planning. Chapters 8 and 14 of this book are of particular interest to problems covered here, since they present material not found in the above-mentioned survey works. Chapter 8: Feedback Motion Planning, covers practical planning algorithms that provide feedback for all possible configurations of the vehicle, generally using navigation functions. Chapter 14: Sampling-Based Planning Under Differential Constraints, discusses approximate solutions for vehicles where differential constraints are important. Another important recent work is [33], which covers the heuristic-based algorithms (such as A\* and D\*) with a focus on dynamic problems, and contain concise summaries of the algorithms and many important references.

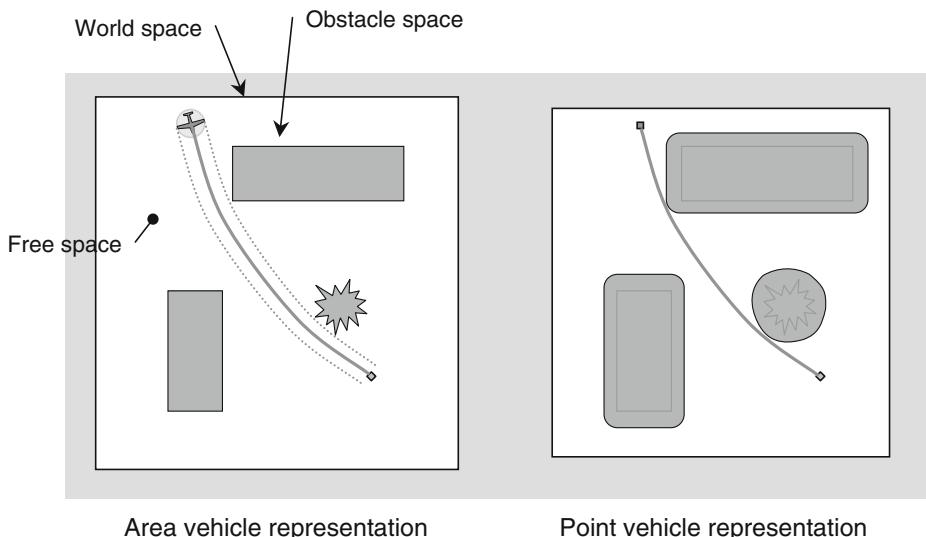
The following does not attempt to duplicate the efforts in these previous surveys, but rather to provide a summary of developments in the field since 1992, with a focus on the problem of trajectory planning for the point vehicle with differential constraints. As a result, there are many references below to the more important of these previous surveys. Especially in the review of path planning literature, for many algorithms a reference to one of these surveys is given as an only source.

## 2.3 Definitions and Terminology

One difficulty when reviewing and studying motion planning or trajectory planning comes from the varied background of the work. We start with an introduction of the key definitions and concepts. These will then be used consistently throughout this paper. Some terms are defined anew here but most are consistent with terminology used in Steven LaValle [70] and Hwang and Ahuja [48].

### 2.3.1 Problem Space

A *vehicle* is an object that is capable of motion. Generally, vehicles are represented by a position vector in two- or three-dimensional space and an orientation vector, along with a geometric model of the vehicle. A vehicle is defined in a similar way to Hwang's *robot*, but is simpler in that it does not contain a manipulator. A *world space* (Fig. 1) is the physical space in which a vehicle exists. For example, a helicopter may exist in three-dimensional Euclidean space, while a ground vehicle may be thought to exist on a two-dimensional surface embedded in three-dimensional space. A *configuration* is a vector of parameters that define the shape of the vehicle—most vehicles can be considered to be rigid bodies in three-dimensional space, and thus defined uniquely by six numbers: three position coordinates and three orientation coordinates. A robot with a manipulator will generally have a much larger number of parameters, because each degree of freedom of the manipulator adds a parameter to the configuration space. The set of all possible configurations of a vehicle is called the *configuration space* or *C-space*. It is often necessary, especially in the case of aircraft, to include a *state*, which consists of the configuration coupled with rates of change of the configuration. In our aircraft example, the state is given by three



**Fig. 1** Illustration of terms. Two representations of a simple two-dimensional example case are shown in order to illustrate terms. On the left the vehicle is represented by a disk, and on the right the vehicle is represented by a point in the expanded terrain representation: the vehicle is a UAV, initial state is marked by a square, and goal state is marked by a diamond; in both cases the configuration space reduces to the world space; the state space is not shown, but velocity and acceleration constraints are implied. Only the path is shown as a partial representation of the trajectory, without timing information

positions coordinates, three velocity coordinates, three orientation angles, and three orientation rate angles, for a total of 12 variables. The set of all possible states is known as the *state space*. The dynamic characteristics of the vehicle determine the dimension of the system, and many systems may use a reduced set of variables that adequately describe the physical state of the vehicle. It is common to consider smaller state spaces with coupled states, or to extend the state space to include higher-order derivatives. The number of coordinates needed to uniquely represent each configuration or state is known as the number of *degrees of freedom*, and is equivalent to the dimension of the configuration or state space. The world, configuration, or state space is divided into *free space* and *obstacle space*; these are disjoint subsets of the space. Free space is the subset of points in which the vehicle can move without contacting obstacles, and obstacle space is the subset of points representing a collision (in the case of state space, this includes the *region of inevitable collision*) between the vehicle and an obstacle. A *path* is a curve traced by the vehicle in the configuration space, and a *trajectory* is a path that includes the time along the path. A trajectory is typically associated with an equation of motion since the differential equation describes the coupling between the spatial and temporal evolution of the system states. Motion planning refers to either path or trajectory planning, and produces a path or trajectory from an *initial state* or configuration to a *goal state* or configuration. More generally, this can be considered as set of paths from a set of initial states to a set of goal states.

### 2.3.2 Motion Planning Terminology

A motion planning algorithm is considered to be *complete* if and only if it finds a path when one exists, and returns a variable stating no path exists when none exists. It is considered to be *optimal* when it returns the optimal path with respect to some criterion. Note that any optimal planner is also complete. Two weaker forms of completeness and optimality are also commonly used: *resolution completeness/optimality* and *probabilistic completeness/optimality*. Resolution completeness/optimality is related to the discretization of the solution space, and means that as the resolution of the discretization increases, an exact solution is achieved as the discretization approaches the continuum limit. Probabilistic completeness/optimality means that as computing time approaches infinity, the probability of finding an exact solution approaches one. Anthony Lazanas [3] discusses motion planning with uncertainty, and makes use of additional definitions. A *sound* planner is one that always guarantees the vehicle will enter the goal region and stop there without hitting any obstacle despite uncertainty in sensing and control. This definition implies that the uncertainties are bounded. Soundness is perhaps the most crucial property for UAVs, because the results of a collision are so catastrophic.

## 2.4 Algorithm Characteristics

### 2.4.1 Algorithmic Complexity

Standard algorithmic asymptotic complexity analysis methods are used in this paper. The variable  $N$  (or  $n$ ) denotes the complexities of describing the obstacle space (i.e. the number of obstacles), and represent the number of bits needed to define this space. Whenever a continuous space is approximated by a finite set of variables,  $M$  denotes the number of variables used to approximate this space, or the level of discretization. In robotics literature,  $M$  (or  $m$ ) is sometimes used to denote complexity of describing the robot shape—however, in this article  $M$  is used exclusively for the level of discretization. The number of dimensions of the configuration or state-space is denoted by  $D$ . Asymptotic notation is used for complexity analysis whenever possible:  $O()$  means the algorithm is bounded from above (within a constant factor),  $\Omega()$  denotes an algorithm bounded from below, and  $\Theta()$  denotes an algorithm bounded both above and below. Unless otherwise stated, complexity is based on time rather than on memory. Additionally, the standard notation of P, NP, PSPACE, EXPTIME, and EXPSPACE are used to characterize algorithms. The relation:

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$$

holds (with the condition that it is an open problem whether these spaces are proper or improper subsets of each other). Practically speaking, only algorithms in P are solvable in real time. However, approximation algorithms often exist for more difficult problems which admit an approximate solution with lower complexity than the exact solution.

### 2.4.2 Performance of Approximation Algorithms

Standard notation is also used for approximation bounds. A *relative performance guarantee* is denoted by the quantity  $\rho$ : a  $\rho$ -approximation algorithm is accurate within a factor  $\rho$  of the true solution. An *absolute performance guarantee* is denoted by the quantity  $\varepsilon$ , and means the approximation algorithm is accurate within a constant value  $\varepsilon$  of the true solution.

## 2.5 Problem Types

There are a variety of problem types defined in the literature. A problem is considered *static* if there is perfect knowledge of the environment, and *dynamic* if knowledge of the environment is imperfect or changes as the task unfolds. When the obstacles are fixed in space, the problem is called *time-invariant*, and when they are allowed to move, the problem is called *time-variant*. The term *differentially constrained* (or *kinodynamic*) means that vehicle's equations of motion act as constraints on the path, i.e., the path must be a trajectory of the dynamic system (for *differentially unconstrained* problems, the vehicle may use infinite accelerations to achieve a path).

It is possible to further categorize problems based on the assumed vehicle shape, environment type and behavior. The common problem types used in literature are described below.

### 2.5.1 Point Vehicle (*Point Robot*)

In this problem, the vehicle is modeled as point within the world space. Thus the configuration space is the same as the world space. Often, a vehicle is modeled by fitting it inside a bounding ball (in two-dimensional Euclidean space this is a circle and in three-dimensional Euclidean space a sphere), and the configuration space is simply the world space with the obstacles expanded by the radius of the vehicle's bounding ball. Thus the ball-shaped-vehicle problem is the same as the point vehicle problem. This is a conservative approximation to and simplification of the mover's problem (described below).

This is perhaps the simplest problem, and optimality is defined in terms of the distance between initial and goal points, i.e., the minimum-length path is the optimal path.

### 2.5.2 Point Vehicle with Differential Constraints

In problems with differential constraints, time and states have to satisfy the equations of motion of the vehicle (associated with Newton's second law). Typically the states are constrained by hard limits on velocity and acceleration, and sometimes also on higher-order derivatives of position. For many UAVs, this more realistic model is needed for stable control of the vehicle. In the limit of a vehicle constrained only by velocity bounds (a vehicle capable of infinite acceleration), this problem becomes simplified to a point vehicle problem without differential constraints.

Optimality may be defined as minimizing the flight time between initial and goal points, or by minimizing some other attributes of the trajectory, such as energy consumed.

### 2.5.3 Jogger's Problem

This problem, named in Shkel and Lumelsky [101], deals with the dynamic problem of a jogger with a limited field of view attempting to reach a goal position. The jogger's problem is representative of a vehicle with differential constraints, operating in a dynamic and possibly time-variant environment, with limited sensory range.

Optimality is defined the in the same manner as in other differentially constrained problems.

### 2.5.4 Bug Problem

This is actually a special case of the jogger's problem, in the limit when the field of view goes to zero. A complete algorithm for this problem is described in [118]. In this case, the vehicle must touch the obstacle (or come in very close proximity) in order to sense it.

### 2.5.5 Weighted Region Problem

When some regions of the world, configuration, or state-space are known to be more desirable than others, the problem is better modeled as a weighted-region problem. An example of this problem is the case of an off-road ground vehicle that drives more quickly over smooth terrain and more slowly over rough terrain. The terrain is represented by a function close to 0 for smooth terrain, and with an infinite value representing un-navigable terrain. This type of problem may be formulated either as an unconstrained or a dynamics-constrained vehicle.

In this problem, an optimum planner minimizes the integral of the weight over the path.

### 2.5.6 Mover's Problem

This is the problem of moving an object through an obstacle field to a goal state. The vehicle is usually modeled as a rigid body, thus the configuration space has a larger dimension than the world space. A classical problem of this case is the famous *piano mover's problem*. For this kind of problem, it is usually assumed that the object has no dynamic constraints. Mover's problems measure complexity of the vehicle in addition to that of the obstacle field, and call this number  $m$  or  $M$ . A more general case is to allow dynamic constraints, and is most commonly used in manufacturing. The differentially-constrained mover's problem is of very high dimension: for example, a three-dimensional differentially-constrained mover's problem has 12 dimensions. Given this difficulty, this problem is generally solved in configuration space, and many algorithms are merely complete and ignore the optimization problem. Some algorithms maximize closest approach between the vehicle and obstacle field.

### 2.5.7 General Vehicle with Differential Constraints

This is the most sophisticated problem type that is investigated. The differential constraints typically arise in two forms: one is on kinematics, and this kind of problem are usually called nonholonomic problem. Another one is on dynamics, this means that it involves second-order or higher differential constraints. The difference between this problem and the point vehicle problem is that now it is insufficient to model the vehicle with only a point in the world space, since we now need six

variables to indicate the position of the vehicle in a three dimensional Euclidean space. But for a space with no obstacles, the location can be converted to a point in a higher dimension configuration space. For most cases, the configuration space is not a simple Euclidean space. To make things worse, when obstacles are involved, the configuration space itself is not adequate to represent the obstacle avoidance requirements, a higher order phase space has to be employed.

### 2.5.8 Time-Varying Environments

In this problem, the vehicle has to avoid obstacles that are moving in time. Fujimura and Tunii [39] discusses this problem in depth, and determines some upper and lower bounds for algorithm complexity.

Optimal planners for time-varying environments generally attempt to minimize path length or time.

### 2.5.9 Multiple Movers Problem (*Multimovers*)

This is the case where there are multiple vehicles moving in the same space. It can be seen as a combination of the mover's problem and the time varying environment case. An example is a factory floor where many robots move between stations. As the dimension of the configuration space increases for each robot added, algorithms scale poorly with the number of robots.

One possible way of measuring optimality is to minimize the sum of the distance, time, or some other quantity covered by all vehicles in the problem space.

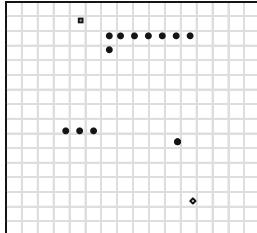
## 2.6 Problem Metrics

The choice of algorithm to use depends on the type of problem to be solved. For example, if the problem is one of plotting a course for a truck to travel across a continent, there is no need to use a dynamics-constrained planner. The most commonly-used metric is obstacle complexity, or the amount of information used store a computer model of the environment. It is generally measured in terms of number of obstacles, obstacle edges, or obstacle vertices, and is called  $N$ . Other metrics are the fill ratio (percentage of the configuration space occupied by obstacles), along with higher-order characteristics, such as mean passage width or degree of clustering of obstacles. A useful and thorough discussion on metrics useful for vehicle motion planning is available in Rhinehart [90].

## 2.7 Algorithm Performance Criteria

One of the purposes of this paper is to list the performance of various motion planning algorithms published in literature. Criteria that are important for a successful practical implementation include operational and computational aspects. Consistently keeping a safe distance from obstacles, and producing smooth paths that exhibit desirable properties (e.g. duration, energy usage) are typical requirements. A reliable, real-time computation without unexpected lags is critical since it can impede maintaining the operational requirements. Low computational complexity is therefore generally an important goal for an algorithm. A faster algorithm can allow a more rapid update of the solution. The main algorithm characteristics

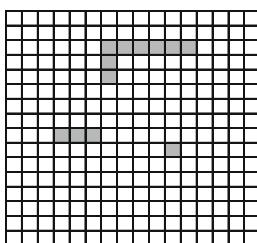
used for describing algorithms include computational complexity (often described in asymptotic big-O notation), and degree of approximation to the optimal solution (generally described by a maximum error factor or bound). Given the often uncertain



#### Sensor model:

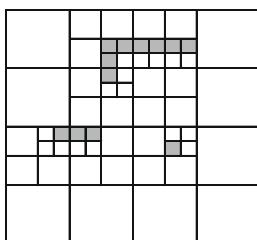
Sensor data representation:

- Black points represent sensed obstacle points
- Initial position marked by a square, goal position marked by a diamond
- The grid is not actually part of the representation



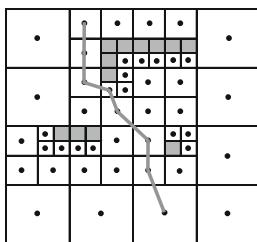
#### Terrain representation:

Obstacle space represented in this example by grey pixels and free space by white pixels



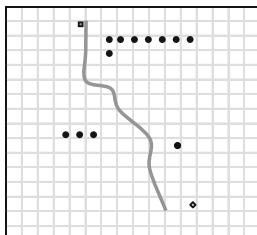
#### Roadmap:

Many types of roadmap are possible – shown here is a quadtree spatial decomposition map



#### Graph search:

The roadmap step produces a graph, and graph search algorithms such as Dijkstra's method or A\* are used to find a sequence of waypoints



#### Trajectory generator:

This starts with waypoints produced by the graph search, and is smoothed into a flyable trajectory. This may be divided into two steps: smoothing and speed control. Only smoothing is shown in this image.

**Fig. 2** Stages in a typical example multi-level decoupled control type planner

nature of real-world problems, the ability to deal with various types of uncertainty and system errors is also crucial for algorithms in practical applications. Another way to deal with uncertainty is to have adaptive algorithms which do not require re-computing.

Asymptotic polynomial time computational complexity for an exact algorithm has been demonstrated for some simple problems (especially in two-dimensional problem spaces). For most problems considered for UAV tasks, it has been proven that no algorithm exists that can find exact solutions in polynomial time. In the absence of a proven computational bound, it is important to have methods that allow precise benchmarking in order to characterize the algorithm, and demonstrate experimental performance. Ultimately, an algorithm's performance is judged by the operators of the actual vehicle in the field; this practical perspective needs to be kept in mind when analyzing computational complexity.

## 2.8 Problem Definition for a UAV as an Illustrative Example

Many UAVs can be modeled as a point vehicle, as described above in Section 2.2. This is the simplest model that is sufficiently detailed to produce a practically meaningful result: Since UAVs don't have to fit into tight spaces while flying, the simplification of bounding the UAV by a rotationally-symmetrical solid has little effect on the trajectory generated by the algorithm. The mover's problem therefore has more complexity than what is actually needed for UAV tasks.

In many UAV problems, the differential constraints are significant since not accounting for the equations of motion may produce conservative results and also precludes explicitly taking into account for criteria like duration or energy.

The jogger's problem, weighted region problem, and multiple vehicle problem are also useful to consider. Poor GPS telemetry, air turbulence, or obstacle detection and registration are common issues in UAV applications. Properly dealing with such sources of uncertainties has not yet been adequately posed or studied for this class of planning problems. In fact, the traditional problem statement of motion planning does not admit solution that is sound or complete, whenever uncertainty is characterized by an infinite-tailed distribution (such as the normal distribution). In other words, the motion planning is ill-posed for most practical problems involving uncertainty.

Figure 2 shows stages in an example multi-stage algorithm, and illustrates the close interaction between sensor data processing and motion planning. Many algorithms are able to combine two or more of these stages.

## 3 Recent Work in Path Planning Without Differential Constraints

Path planning without differential constraints is included in this article because it is often used as a basis for algorithms with differential constraints. Most of the algorithms are well covered by other surveys and textbooks, so comprehensive references are not provided for this class of problems unless significant advancements have been published. The reader is advised to refer to the references given in Section 2.2: Previous surveys. Table 1 provides a direct side-to-side comparison of all the algorithms presented in this section.

**Table 1** Comparison of algorithms for path planning without differential constraints

Algorithm name	Type of problem solved	Number of dimensions	Completeness	Optimality	Proven time complexity
<b>Roadmap</b>					
Visibility graph	Point vehicle	2	Complete	Optimal	$O(N^2)$
Edge-sampled visibility graph	Point vehicle	3	Complete	Resolution optimal	Exact solution is NP hard, approximation polynomial in N and 1/eta
<b>Exact cell decomposition</b>					
Voronoi roadmap	Point vehicle	2	Complete	Non-optimal	$O(N \log N)$
Freeway method	Point vehicle	2	Incomplete	Non-optimal	$O(\text{Exp}[D, N])$
Silhouette method	Point vehicle	Arbitrary	Complete	Non-optimal	$O(\text{Exp}[D, N])$
<b>Approximate cell decomposition</b>					
Trapezoidal decomposition	Point vehicle	2	Complete	Non-optimal	$O(N \log N)$
Critical curves and non-critical regions	Rigid vehicle	2	Complete	Non-optimal	$O(N^2 \log N)$
Cylindrical algebraic decomposition	Rigid vehicle	Arbitrary	Complete	Bon-optimal	$O(\text{Exp}(\text{Exp}(N)), \text{poly}(P))$
Rectanguloid	Point vehicle	2	Resolution complete	Non-optimal	
Quadtree, octree, or 2 <sup>m</sup> tree decomposition	Point vehicle	Arbitrary	Resolution complete	Non-optimal	
Approximate and decompose	Point vehicle	2	Resolution complete	Non-optimal	

<b>Potential-based methods</b>						
Potential field	All types	Arbitrary	Incomplete	Non-optimal		
Potential guided	All types	Arbitrary	Complete	Non-optimal		
Harmonic potential functions	Point vehicle	Arbitrary	Complete	Non-optimal		
Wavefront expansion	Regid vehicle	Arbitrary	Resolution complete	Resolution	$O(M \log M)$	$O(M \log M)$
Wavefront expansion with a skeleton (NF2)	Regid vehicle	Arbitrary	Resolution complete	optimal	$O(M \log M)$	$O(M \log M)$
Continuous Dijkstra	Regid vehicle	2	Complete	Optimal	$O(N^2 \log N)$	
<b>Probabilistic approaches</b>						
Randomized potential fields	All types	Arbitrary	Probabilistically	Non-optimal		
Global learning	All types	Arbitrary	complete when complete search used	Non-optimal		
<b>Weighted region problem</b>						
Exact algorithm	Polygonal weighted regions	2	Complete	Optimal	$O(N^8 \log N)$	
Approximation algorithm	Polygonal weighted regions	2	Complete	Resolution optimal	$O(N^* M \log N^* M)$	
Geodesic distance on a grid	Weighted grid	Arbitrary	Resolution complete	Resolution optimal	$O(M \log M)$	

### 3.1 Roadmap Methods

These methods reduce the problem to that of a graph search by fitting a graph (ie. a roadmap) to the space. In order for the algorithm based on a given roadmap to be complete, it must follow certain topographical properties [66].

A couple of more recent roadmap methods have been published. Dechter and Pearl [29] verifies the optimality of A\*, in comparison with generalized search strategies.

#### 3.1.1 Visibility Graph

This is an exact solution to the point vehicle problem, is both complete and optimal, and runs in  $O(N^2)$  time. However, it is computable in only two dimensions. In a C-space of more than two dimensions, an exact solution is proven to be NP-hard. This approach uses the knowledge that the shortest path grazes polygonal obstacles at their vertices, and builds a roadmap of lines connecting each vertex with all vertices visible from its position. Since the minimum-length path comes arbitrarily close to obstacles many times in a typical path, this approach offers no safety buffer to prevent collisions in the case of systems with uncertainty in their position. One way researchers have attempted to avoid this problem is by expanding the obstacle space by a ball larger than the vehicle's longest radius.

#### 3.1.2 Edge-Sampled Visibility Graph

This algorithm approximately solves the three-dimensional path length minimization point vehicle problem, and runs in complexity that is polynomial in both error and in computation time. This algorithm assigns multiple vertices along edges of polyhedral obstacles so that there is a minimum edge length  $\eta$ , and builds a visibility graph from this expanded set of vertices. It is complete and resolution-optimal.

#### 3.1.3 Voronoi Roadmap

Given the difficulty in controlling vehicles precisely enough to follow the minimum-distance path without risk of colliding with obstacles, many skeleton-based roadmap approaches have been taken. The Voronoi approach builds a skeleton that is maximally distant from the obstacles, and finds the minimum distance path that follows this skeleton. This computationally efficient algorithm runs in  $O(N \log N)$  time. Although this algorithm is a two-dimensional algorithm, there have been several efforts to convert it to 3 dimensions. It is complete, but not optimal.

More recently, Choset and Burdick [21] proposed a hierarchical Voronoi graph generalized to multiple dimensions, and in [22] the algorithm is updated to allow incremental construction. Howlett et al. [45] discusses use of Voronoi roadmap methods for practical unmanned helicopter operation.

#### 3.1.4 Freeway Method

This method, like the Voronoi roadmap, builds a skeleton that is distant from the obstacles, by fitting free space with generalized cylinders. It is not limited to two dimensions, but it is incomplete and non-optimal.

### 3.1.5 Silhouette Method

While not used in practical applications, this method is useful for proving bounds on complexity. This is the only algorithm that is proven to be complete for an arbitrary number of dimensions with arbitrary obstacle geometry. Canny [18], who designed the algorithm, proved that it can solve the problem in double exponential time. The algorithm is complete but not optimal.

## 3.2 Exact Cell Decomposition

These methods decompose the free configuration space into smaller convex polygons, which are then connected by a graph and searched using a graph search.

### 3.2.1 Trapezoidal (Vertical) Decomposition

This approach divides the free space into trapezoidal regions by dividing it with vertical lines from each of the obstacle vertices. The vertical lines are trimmed so that they do not bisect the obstacles themselves. A roadmap is then formed by connecting the midpoints of adjacent trapezoids, and searched using a graph searching algorithm. This approach is complete but not optimal, and runs in  $O(N \log N)$  time.

### 3.2.2 Critical-Curve Based Decomposition

While the trapezoidal decomposition is useful for point vehicle path planning, rigid vehicles with freedom to rotate require a more complex approach. In this algorithm, free space is divided into critical and non-critical regions. The boundaries of these regions are piecewise polynomial curves. The various regions formed by the decomposition process are connected by a graph and this graph is searched for a path. The algorithm is for two-dimensional problems, is complete but not optimal, and runs in  $O(N^2 \log N)$  time.

### 3.2.3 Cylindrical Algebraic Decomposition

This more complex decomposition extends the critical-curve decomposition to three-dimensional problems. It bisects parts of the free space using critical surfaces. It is complete but not optimal, and runs in double exponential time.

### 3.2.4 Connected Balls in Free Space

This approach is designed to deal with un-structured obstacle fields, and operates by filling free space with overlapping balls (for instance, spheres are balls in three-dimensional Euclidian space) that are totally in free space.

Vandapel et al. [117] introduces unions of free-space balls as a roadmap in multi-dimensional space.

### 3.3 Approximate Cell Decomposition

#### 3.3.1 Rectanguloid Cell Decomposition

This divides the entire configuration space into rectanguloid regions, and labels each rectanguloid as being completely filled (black), partially filled (grey), or completely empty (white). It is proven to be resolution-complete.

The most common example is that of the A\* or D\* search over a square or cubic grid of occupied or unoccupied cells. Ferguson et al. [33] reviews this type of approach, with a focus on dynamic problems.

#### 3.3.2 $2^m$ Tree Decomposition (Quadtree or Octree Decomposition)

This decomposition is designed to reduce the number of points needed to represent obstacles as compared to a full grid representation.

This type of representation is becoming increasingly more common, and several new papers using tree decompositions have been published. Sinopoli et al. [102] uses wavelet transform processing for path planning purposes. Behnke [5] proposes a quadtree algorithm with weights put in to avoid obstacles by a longer distance. Soucy and Payeur [105] compares a fixed resolution vs. quadtree characterization for similar problems. Tsenkov et al. [113] describes a real-time implementation of planning over a quadtree representation of obstacles, demonstrated on an unmanned helicopter.

### 3.4 Approximate and Decompose

This decomposition is similar to the trapezoidal decomposition, but replaces the triangular end regions with rectangular mixed regions. This approach reduces the proportion of mixed area in comparison with a grid decomposition with mixed cells.

### 3.5 Potential Field Methods

Potential field methods are based on the idea of assigning a potential function to the free space, and simulating the vehicle as a particle reacting to forces due to the potential field. The goal point has the lowest potential, and attracts the vehicle, while obstacles repel the vehicle. Since their initial publication Khatib and Mampey [59], potential field methods have been generally known for being of low computational complexity but incomplete. However, a potential field which has the properties of a navigation function [92] makes a complete path planner. There are two classes of potential fields known to satisfy properties of a navigation function: those based on harmonic functions [27] and those based on solving the optimal distance-to-go function [42]. These methods require, however, discretizing the configuration space into a grid with  $M$  points, and this discretization scales as  $O(M^D)$  with the dimension  $D$  of the configuration space. The added advantage of a navigation function is that it can be used to provide direct feedback control, rather than relying on feed-forward control, as traditional trajectory planners do. A single navigation function produces a trajectory for every possible starting point in the configuration space.

### 3.5.1 Potential Field with Gradient Descent (Virtual Force Field—VFF)

This is the original potential field approach, and is designed to run quickly. It assigns a decaying function to the goal point with a negative minimum value, and a decaying function to each of the obstacles with a positive maximum value, and sums the functions from the goal and all obstacles to get the total potential.

Since the VFF algorithm is sometimes used directly for trajectory generation and is valid for an arbitrary number of dimensions, some references are included here. Khatib and Mampey [59] is cited as the first use of the potential field method, and for many years the term “potential field” applied strictly to this algorithm. Borenstein and Koren [12] and Borenstein and Koren [13] couple sensing with planning in an evidence grid, using the VFF method to provide motion planning inputs for ground robots. A reactive method, based heavily on localized sensor input and immediate response, uses a grid with VFF. Yoram Koren and Johann Borenstein [63] discusses inherent limitations of potential methods, but the contents of this article apply only to the VFF algorithm. The limitations mentioned in this article, trap situations due to local minima and no passage between closely spaced obstacles, do not apply to potential fields that are navigation functions in the sense of Rimon-Koditschek. The oscillations cited here result from poor control system design, and can be eliminated by applying classical control theory. Ahuja and Chuang [2] publish a generalized potential field based on summed Poisson solutions of simple cases.

### 3.5.2 Potential Field Guided Search (Depth-First, Best-First, Variational Planning—Arbitrary Potential Field)

This approach is designed for potential fields that have local minima. Rather than use gradient descent, which is easily trapped in local minima, a search that is complete in the resolution or probabilistic sense is used. This can be considered as being similar to an A\* search with the simple heuristic replaced by a potential field, although this approach is somewhat more general since it admits depth-first searches as well. The variational planning approach uses the potential as a cost functional, and attempts to find a path to the goal point that minimizes this cost.

### 3.5.3 Harmonic Potential Functions

This class of functions is based on solving a partial differential equation with a Laplacian term. These equations include Laplace’s equation, Poisson’s equation, the conduction heat flow equation, approximations to Navier–Stoke’s equation, and other partial differential equations of this type. While not producing an optimal path, these equations generate functions that are true navigation functions in the sense of Rimon and Koditschek, meaning they are smooth, have only one local minimum that occurs at the goal point, potential obtaining a constant maximal value at the boundaries of obstacles, and has a non-degenerate Hessian at each critical point of the function. This implies that these functions may be used to produce a complete planner using a simple gradient descent search. This type of function needs to be solved by a numerical method with global information, and is generally solved on a grid.

The first publications of this method are in Akishita et al. [93] and Connolly et al. [27]. Connolly et al. [26] provides further theory into the dynamic aspects of the approach, including a Hamiltonian framework to model inertial effects, and allowing orbits around a goal point. Zelek [124] makes use of the harmonic potential field, and [107] use solutions to the diffusion equation as potential fields. Kazhdan et al. [83] describes Poisson surface reconstruction: while not a path planning paper, it describes numerical methods useful for solving the three-dimensional partial differential equation. Scherer et al. [94] describes the results of this method actually flown on an unmanned rotorcraft, and combines a harmonic potential planner with reactive planner.

### *3.5.4 Continuous Dijkstra (Optimal Navigation Function Using Visibility Polygons)*

The exact optimal navigation function can be generated using this method for two-dimensional problems, and runs in  $O(N^{5/3} \log N)$  time. This method divides the space into visibility polygons, each encoding “wavelets” rooted at a polygon vertex, and searches them. However, it cannot be extended to higher-dimensional problems.

### *3.5.5 Wavefront Expansion (Dynamic Programming)*

This is the grid-sampled version of the Continuous Dijkstra method, and can be used in multiple dimensions, running in  $O(M \log M)$  time. It is also closely related to the complete search over a grid using dynamic programming. It is able to produce a path that is complete and optimal in the resolution sense.

### *3.5.6 Wavefront Expansion with a Skeleton (NF2)*

This is an obstacle avoiding navigation function that plans paths along the medial axis rather than nearly optimal paths, and also runs in  $O(M \log M)$  time. The advantage of this is that the paths do not graze obstacle, but rather avoid the nearest obstacles by a maximal amount. Thus it is more suitable for practical applications, where the position of the vehicle or obstacles may not be known with perfect certainty.

## 3.6 Probabilistic Approaches

### *3.6.1 Randomized Search in a Potential Field to Avoid Being Trapped in Local Minima*

This method is designed to help a gradient descent search routine escape from local minima that are found in the original potential based method. There are various approaches to randomizing the search, such as adding random components to subsequent steps or changing the values of obstacle potentials at random. Completeness of these algorithms has been difficult to prove.

Caselli et al. [20] uses a heuristic stochastic search (instead of a random walk) to escape local minima.

### *3.6.2 Potential Field with a Global Optimization Search or Learning Algorithm*

The field of global optimization has several algorithms that are able to solve an optimization problem with local minima. Any of these may be applied to a potential field with local minima in order to find the true solution.

### 3.6.3 Probabilistic Roadmap (PRM)

This approach is important because it admits a solution to problems of arbitrary complexity and dimension that is convergent in the probabilistic sense. It is generally used in manipulator problems, where the configuration space is typically of high dimension and complex. However, the rate of convergence is slow, and the paths it produces are not optimal. In the case of two- and three-dimensional configuration space planning, the resultant paths are required to be smoothed significantly before a trajectory can be formed. It has particular difficulty in converging in problems that have long passageways.

Kavraki et al. [57] initially describes this algorithm. Mazer and Ahuactzin [76] combines random landmarks with a local search. Kavraki et al. [56] discusses theoretical properties of PRM algorithm. Bohlin and Kavraki [11] outlines the Lazy PRM, a modified method that overcomes some of the simple PRM's limitations.

## 3.7 Weighted Region Problem

In this problem, the configuration space has an associated weight function. The path planner has to minimize to weight integrated over a path.

### 3.7.1 Exact Algorithm for Polygonal Weighted Regions

This algorithm uses results of variational calculus used to derive Snell's law of refraction. It is capable of finding the exact path over weighted polygonal regions. While operating in polynomial time, it is expensive, requiring  $O(N^8 \log N)$  operations [82].

### 3.7.2 Approximation Algorithm for Polygonal Weighted Regions

The approximation algorithm for this problem is resolution optimal, and can run in  $O(MN \log MN)$  operations [88]. It divides the edges of the polygons into shorter segments, and finds a path connecting these vertices.

### 3.7.3 Exact Algorithm for Weighted Grids

This is a simpler problem than general polygonal weighted regions. It is simply a discrete search over a grid, and can be accomplished in  $O(M \log M)$  time. It is complete and optimal in the resolution sense.

Recently, Ikonen and Toivanen [51] discuss grey-level distance transforms for shortest paths on curved surfaces and their approximation using dynamic programming, and Ikonen [50] discusses the priority queue implementation and compares computational complexity with a previous iterative approach.

## 4 Trajectory Planning with Differential Constraints

Most trajectory planning problems relevant to today's UAV applications have to be considered as dynamics-constrained problems. The behavior of aerial vehicles is often not sufficiently well approximated by their kinematics (as is more often the case in ground vehicles). Taking into account the equations of motion is directly relevant to guaranteeing the soundness of the planner, since approximating the

dynamics solely through a kinematic model with constraints will lead to overly conservative models. The equations of motion are also relevant in details of the vehicle maneuvering affecting energy or duration of the trajectory. For example Kong and Mettler [62] show that performance criteria have a significant effect on the resulting trajectories and heuristic methods (e.g. minimizing distance) are not able meet specific performance requirements.

This class of planning problems is substantially more difficult to solve due to the dependency between time and the state-space introduced by the differential constraints. Even in the trivial case of connecting two states in a configuration space without obstacles, an exact solution is generally not possible. An exact solution is available for two-dimensional problems only, solvable in exponential time and polynomial space [54]. However, this approach cannot be extended to three-dimensional problems. For applications requiring a vehicle to navigate among obstacles or complex terrain, algorithms that exploit some form of approximation or heuristic are necessary, not only for the merit of finding a feasible or sub-optimal trajectory but also for the need to negotiate with hardware capacity. Solutions to this class of problem represent a newer research area where very few approximation bounds or benchmarking results have been proposed.

A tabulated overview of these algorithms is given in Table 2. Blank table cells correspond to properties of algorithms that cannot be determined from the literature. For a more detailed breakdown of potential-based methods, refer to Table 1.

## 4.1 Sampling-Based Trajectory Planning

### 4.1.1 Grid-Based State Space Search

This method played an important role because it establish proof for the completeness and optimality (in the resolution sense) for the case of a polynomial approximation to the problem with dynamic constraints, which are represented in the form of velocity and acceleration bounds. Furthermore, it defines an arbitrary speed-varying safety corridor, making this particular algorithm one of very few trajectory planning algorithms with a proven explicit safety guarantee. The way this algorithm works is that it discretizes the entire state space of the vehicle onto a lattice, and searches the lattice for the time-optimal path that satisfies the safety corridor. Although the algorithm converges as a polynomial of the number of obstacles, it is a high-order polynomial that is exponential with the number of dimensions, making practical real-time implementation difficult due to high dimensionality of the state space. It also has difficulty in solving planning problems for the case of under-actuated vehicles, which are quite common in application.

Donald [31] is the first publication of this algorithm. Donald and Xavier [32] gives a proof of a lower complexity bound for the same algorithm. Reif et al. [89] uses nonuniform discretization to reduce the algorithm's complexity.

### 4.1.2 State-Space Navigation Function with Interpolation

Much like the grid-based state space search, this method approximates the time-optimal path to a goal. Instead of returning only a single trajectory, it returns a navigation function over the state space. This navigation function can be computed by either value iteration or control policy iteration, although value iteration is more

**Table 2** Comparison of algorithms for trajectory planning with differential constraints

Algorithm name	Completeness	Optimality	Soundness	Proven time complexity
<b>Canonical two-dimensional solution</b>				
State-space sampling	Complete	Optimal	Sound	$O(\exp(N))$
State space lattice search	Resolution complete	Resolution optimal	Resolution sound	$O(C^d N^{err(-6d)})$
Dynamic programming with interpolation	Resolution complete	Resolution optimal	Resolution sound	$O(C^d N^{err(-3d)})$
Rapidly-expanding random tree (RRT)	Probabilistically complete	Non-optimal	Resolution sound	
Reachability graph, rapidly-expanding dense tree (RDT)	Resolution complete	Resolution optimal	Resolution sound	
Pruned reachability graph	Resolution complete	Resolution optimal	Resolution sound	
<b>Minimum distance path followed by trajectory forming</b>				
Linked boundary point solvers	Resolution complete	Resolution sound if constraints are met	Resolution sound if constraints are met	$O(\text{polynomial}(N))$
Maneuver automation	Resolution complete when complete search used	Non-optimal	Resolution sound	
Decoupled approach	Resolution complete when complete search used	Non-optimal	Resolution sound	
Plane slicing	Resolution complete when complete search used	Non-optimal	Resolution sound	
<b>Mathematical programming</b>				
Path-constrained trajectory planning	Resolution complete when complete search used	Non-optimal	Resolution sound	$O(\exp(M))$
Gradient-based trajectory optimization	Resolution complete when complete search used	Resolution optimal when complete search used	Resolution sound (for infinite horizon)	
Model predictive control (receding horizon)	Resolution complete when complete search used	Non-optimal	Resolution sound if constraints are met	
<b>Potential-based</b>				
Path-constrained trajectory planning	Resolution complete when complete navigation function is used	Non-optimal	Resolution sound if space varying speed limit constraints are met	$O(M \log M)$
Randomized potential fields		Non-optimal		

popular. Bertsekas [6] gives details in a more general sense. For any given state, performing gradient descent on this navigation function will produce an approximately time-optimal trajectory to the goal. Interpolation between lattice points allows a continuous function which can be used for feedback. The algorithm takes on the same order of complexity as the grid-based state space search. A recent paper which employs this method can be found in LaValle and Konkimalla [68].

#### 4.1.3 Rapidly-Expanding Random Tree (RRT)

This works by using a stochastic search over the body-centered frame of reference, and expanding a tree through a random sampling of the configuration space. This algorithm is proven to be complete in the probabilistic sense, and to produce a trajectory that is feasible given the dynamic constraints of the vehicle. However, there is no proof of the convergence rate or of optimality. In fact, it is known that certain types of problems, such as those with long passageways, may have very slow convergence rates. However, in these cases gradient based optimization methods can be applied to reach locally optimal solutions.

LaValle [67] is the first publication describing the RRT, followed by a more detailed report [71]. In these two papers, the vehicle is considered holonomic; neither dynamics nor kinematic constraints are considered. Frazzoli et al. [36] extends the RRT to systems with dynamics by reducing the whole system to finite states possible using a finite automaton. LaValle [69] compares dynamic programming and RRT algorithms. Frazzoli et al. [37] gives a more extensive description to the RRT method, and Redding et al. [87] provides an application of the RRT: this approach uses the RRT in combination with a Dijkstra search based path refinement step.

#### 4.1.4 Reachability Graph

This approach also uses a body-centered frame of reference: for each state, the tree explores variety of states including the maximum deflection states. The combinatorial complexity of such a process is often prohibitive (in fact, exponential with  $M$ ), and the tree quickly fills the space close to the initialization point. The basic approach has been employed for curvature-constrained path problem in two dimensions [70].

Another way to make this approach tractable is to use cell-based pruning: the configuration space is divided into cells, and the reachability graph is set up to be a tree that has no more than one leaf ending in each cell [70].

### 4.2 Decoupled Trajectory Planning

#### 4.2.1 Minimum Distance Discrete Path Followed by Trajectory Forming (Two-Step Approach)

The algorithms in this section follow the same general approach: first a discrete path through the configuration space is found by one of the algorithms, and then the resulting path is used as the basis for the generation of a trajectory that is feasible for the dynamics-constrained vehicle. For the first stage, a well-known algorithm such as A\* is applied over a grid, the PRM, or the Voronoi approach is typically used. Complexity in this approach is typically dominated by the path planning phase computations.

This decomposition-based approach allows efficient computation of approximate solutions, but makes proofs of completeness, optimality, or even of soundness, difficult. There is no safety corridor built in and soundness requires checking whether the trajectory intersects with obstacle space, and re-computing or rejecting trajectories that do not pass. Generally, practical implementations of these algorithms will use a conservative safety corridor to prevent collisions.

#### *4.2.2 Discrete C-Space Search Connected by Two-Point Free-Space Boundary Value Solver*

In this approach, a set of waypoints is first selected (generally by a grid-based search), a velocity is assigned to each one, and a boundary-value problem connecting each waypoint to the next point is solved. Although these boundary-value problems don't have to deal with obstacles, a general simple solution is not possible, so the solution is generally approximated using numerical methods. As with the previous method, an explicit collision check on the trajectory is needed to ensure soundness, possibly at the expense of completeness.

#### *4.2.3 Hierarchical Decoupled Planning and Control*

This is a general strategy that is used in most practical applications. Since feedback control is typically required for reliable operation of air vehicles and most traditional algorithms provide only feed-forward solutions, the hierarchical decoupled planning and control approach provides a straightforward way to integrate a waypoint planning algorithm and the vehicle control system. The overall hierarchic system involves open- and closed-loop controllers operating at a variety of rates, linked together from top to bottom. The outer, open loop consists of a discrete search that produces a set of waypoints leading to the goal while avoiding obstacles. The second open loop level smoothes this set so that the waypoints are feasible given the vehicle's velocity and acceleration limits. The third open loop level generates a timing function along the trajectory, and creates a set point (or "rabbit") that moves through space, and last, the inner loop is a closed-loop tracking controller that attempts to minimize the error between the vehicle and the rabbit.

This approach has been popular for UAVs because it is relatively easy to implement since it requires no sophisticated solvers and is made of a hierarchy of modules with well defined functions. Since many unmanned aircraft are often already equipped with an inner-loop tracking controller it can be easily adapted to a variety of vehicles. In addition, the majority of UAVs currently in production already have a waypoint following system (without obstacle avoidance) as the primary means of control—this makes the multi-level decoupled control approach easiest to integrate with the existing control scheme. However, there is no proof demonstrating in general that this method is sound, complete, nor that it produces near-optimal paths. Evidence of a sound planner needs to be produced on a case-to-case basis for safe use of this approach.

Boyle and Charnitoff [14] covers the closed-loop section of the planner, or the maneuver tracking controller. Judd and McLain [55] describes a Voronoi-based planner followed by spline smoothing for trajectory formation. Yang and Zhao [122] is based on A\* coupled with higher order heuristics. Suzuki et al. [108] uses A\*, followed by direct optimization of the trajectory, using an RTABU search. Scherer

et al. [94] uses an evidence grid with a Laplacian-based potential method as the outer loop, a reactive planner (dodger) to enforce soundness, a speed controller to convert the path into a trajectory, and an inner loop flight controller. Kim and Bhattacharya [60] is based on a modified visibility graph roadmap method that is followed by finite horizon optimal control. Takahashi et al. [110] covers the design and characterization of the inner-loop control law used in such a multi-level decoupled controller for an unmanned rotorcraft based on two types of path planners (quasi-3d implementations of an A\* and a Voronoi-based planner). Howlett et al. [44] describes the implementation of the two path planner modules.

#### *4.2.4 Discrete C-Space Search Interpolated with Polynomial Arcs*

In this approach, an ordered set of waypoints produced by a discrete planner is fitted with a spline made up of polynomial arcs. This spline is set up so that the vehicle can follow it without violating acceleration constraints, and typically consists of circular arc segments with a minimum radius and straight segments. Other types of spline, such as the Pythagorean hodograph, have been proposed for the same purpose.

Yakimenko [120] uses optimization on a family of polynomials to approximate the two-point boundary value problem solution, which can be used for interpolating between states. Shanmugavel et al. [99] describe Pythagorean hodograph interpolation. Both approaches use optimization to determine the spline that gives best performance based on minimal time or minimal control energy.

#### *4.2.5 Curvature-Constrained Paths with Speed-Control Planning*

This is coupled with the previous approach: once a path is produced that is curvature constrained, it is possible to optimize the speed so that the vehicle will follow this path in minimal time. This path-constrained trajectory planning problem requires solving only a two-dimensional path-constrained state space (with time and velocity axes), and can be accomplished efficiently.

Slater [104] describes an approach for helicopter short trajectories based on guidelines for piloted flight. Yang and Kapila [121] considers curvature constrained paths based on the Canny's exact algorithm for two-dimensional problems.

#### *4.2.6 2-D Voronoi Solutions from Multiple Body-Based Planes*

In this approach, several planes containing both the initial and goal points are extracted as subsets of the three-dimensional configuration space. These planes that are discriminated from each other by a single angle—this angle is discretized so there are a finite number of planes (in experiments, it was found that four to six planes suffices). These planes are then searched with a two-dimensional path planner, such as a Voronoi roadmap planner, and then rated against each other according to optimality criteria. The best of the set is chosen.

Howlett et al. [45] provides a basis by describing the 2-d Voronoi approach, and Whalley et al. [119] describes the Voronoi-based plane slicing method with multi-level control for an unmanned rotorcraft with multiple-target surveillance ability.

### 4.3 Finite-State Motion Model: The Maneuver Automaton (MA)

The general idea of finite state models is to reduce the optimization or search problem from an infinite-dimensional space to a finite one. It helps to significantly reduce the computational complexity of a trajectory optimization.

There are two primary type of finite-state models for dynamics systems: the first is a discrete-time model with quantized states (quantization); another choice is to relax the restrictions on control and time and instead use operations over discrete stages with fixed start and end states. These stages, which are feasible time-parameterized curves in state space, are called motion primitives.

In the context of vehicle trajectory planning, this model is called a maneuver automaton (MA). The concept of MA for vehicle is based in part on the observation that human pilots achieve agile control using a combination of trim trajectories and maneuvers (non-equilibrium transitions between trims).

While Agarwal and Wang [1] is not exactly a MA, but is based on Canny's exact algorithm with fixed-radius segments, it includes some ideas similar to the MA. Yakimenko [120] stores a set of solutions to the two-endpoint boundary value problem as a motion primitive set (but does not deal with obstacles). Piedmonte and Feron [85] and Gavrilets et al. [40] investigate the concept of maneuver automaton for human piloted acrobatic flight. Frazzoli et al. [35] provides a rigorous definition of the concept of MA within the context of autonomous guidance. In the basic MA form the set of trim and maneuvers are used to pre-compute a cost-to-go map. This map can be used online with a greedy guidance policy. States falling between the pre-computed values are obtained via interpolation. Mettler et al. [78] provide a simulation example of this MA based guidance policy for online rotorcraft guidance. In this form of guidance policy, the vehicle behavior is constrained to the set of primitives in the MA. Hence, for agile vehicles, it can be an issue to achieve a sufficiently expressive MA due to the “curse of dimensionality”. To relax the vehicle behavior and provide flexibility in an obstacle-rich environment, Schouwenaars et al. [96] use the concept of the MA within a receding horizon optimization framework. Instead of fixed trim trajectories, the trims are replaced by controllable linear modes. This model is a more faithful emulation of the human control strategy. The maneuvers are still open-loop trajectories used to transition between modes or for a tactical purpose (e.g. split-S maneuver can be used to reverse direction of flight without lateral displacement). Dever et al. [30] extends this framework to allow interpolation between maneuver boundary conditions within a class of maneuver. This provides additional flexibility for the initiation and completion of maneuvers. In the meantime, the MA has also been used to generate a state-dependent cost-to-go map for a receding horizon planning [81]. This will be discussed in Section 4.5.

Parallel to the maneuver automaton concept, a similar idea called “control quanta” is introduced for driftless systems with a symmetry property [72]. For this special class of systems, by employing control quanta, the reachable set can be restricted to a lattice. And by choosing a suitable set of control quanta, the reachable set can be everywhere dense in the limit when  $M$  approaches infinity. The difference is that for the control quanta method, the control policy is chosen from a collection of control library policies, while for motion primitive method the trajectory is chosen from a library of maneuvers that can result from a various control strategies.

## 4.4 Mathematical Programming

Mathematical programming methods treat the trajectory planning problem as a numerical optimization problem. Some popular methods include Mixed Integer Linear Programming (MILP), nonlinear programming, and other constrained optimization approaches. These methods are also known as trajectory optimization methods, since they find a trajectory to a goal point that is optimal in the resolution sense. However, the cost functions typically have a number of local minima, thus finding the global solution strongly depends on the initial guess (the general formulation is NP-hard, although given an initial guess sufficiently close to the global solution, the optimization converges in polynomial time).

For this type of problem, one standard strategy is to enforce the equations of motion as constraints. An earlier review of this method can be found in Betts [10]. Another strategy is to discretize the variational principles underlying the systems dynamics, such as Hamilton's principle or Lagrange–D'Alembert principles, and then these discrete equations can serve as constraints. This kind of strategy is called Discrete Mechanics and Optimal Control (DMOC) and can be found at Marsden and West [75] and Junge et al. [53]. Kobilarov et al. [61] extends the framework to deal with obstacles. Several approaches have been used to break this into simpler problems.

### 4.4.1 Initialization of Mathematical Programming Methods (Infinite Horizon Control)

An initial trajectory for the mathematical programming methods, such as a constant-speed trajectory, can be formed using a discrete search over the configuration space. These waypoints are then used as an initial point in the mathematical programming search. If this initial point falls within the basin of attraction of the global solution, then the mathematical programming approach can find the optimal solution in polynomial time. However, unless care is taken in finding proper initial points, the solution could fall into a local minimum, and general global optimization approaches guaranteed to find the global minimum are prohibitively expensive.

Toussaint et al. [112] describes this approach and Richards and How [91] covers both single and multiple-vehicle planning in two-dimensional cases. Milam [74] produces trajectories for a constrained ducted fan setup with two degrees of freedom and provides flight test results. Carlyle and Wood [19] describes the Lagrangian relaxation solution. Menon et al. [77] give an application of direct optimization used in aerospace: they produce fuel-optimal periodic cruise trajectories involving high Mach numbers and allowing periodic large changes of altitude. Carlyle and Wood [19] describes again the Lagrangian relaxation approach, with the addition of risk avoidance (UAVs are required to avoid SAM sites). Keith et al. [58] describes how a trajectory planning problem within complex terrain can be converted into a MILP problem by representing terrain as a piece-wise affine function.

## 4.5 Receding Horizon Control (Model Predictive Control)

Receding horizon control (RHC) or model predictive control (MPC) solves the numerical optimization problem over a reduced time horizon. In this approach, an open-loop control policy is designed to control the vehicle until the end of

the time horizon. Optimization over a finite horizon requires reduced computation time, however, it will not converge to a globally optimal solution without using an appropriate cost-to-go function to capture the discarded portion of the trajectory. Except in trivial cases, optimality and completeness are difficult to prove. In UAV guidance applications, this approach has often been used with a MILP solution, however, there is no reason to restrict the receding horizon control method with this type of numerical solver.

Jadbabaie [52] introduced receding horizon control to solve trajectory planning problems for general nonlinear systems in an obstacle-free environments. The paper also provides necessary conditions for the stability of the RH scheme by employing concept of control Lyapunov function. The RHC-based approach is used in Singh and Fuller [103] to navigate a vehicle with nonlinear dynamics through a toy urban environment with a vector of known way-points in a decoupled manner: first convert the problem into a convex optimization problem by linearizing the vehicle model and obtain a nominal trajectory, then generate a series of overlapping convex regions around the trajectory and finally within these feasible convex regions open-loop trajectory is updated by RHC as time evolves.

The features of RHC makes it a suitable trajectory planning technique for many UAV applications. Sensory information can be incorporated into on-line computation thus it can deal with uncertainty; at the same time, only local information is integrated thus it can reduce computational effort. However, properly designed terminal cost function needs to be provided to the on-line planner to guarantee completeness and near-optimality. For instance, Schouwenaars et al. [97] uses a cost function based on a visibility graph and [7] estimates the cost function by searching a graph representation of the environment with Dijkstra's algorithm. Schouwenaars et al. [97] also investigates the effect of the length of planning horizon on the computation time and optimality. Schouwenaars et al. [96] and Shim and Sastry [100] investigate hardware and software implementation details and also provide experimental flight-test results for a fixed-wing aircraft and a rotorcraft vehicle in various 2D guidance missions. Mettler and Bachelder [79] describe a receding horizon scheme in 3D with visibility constraints. Frew [38] describes the way to apply the Fisher information matrix to integrate passive, non-cooperative sensory information into the RHC framework. Prazenica et al. [86] estimates the obstacle map from local visual data by using an adaptive learning algorithm so as to avoid unknown obstacles in an urban environment and uses RHC to generate a trajectory and control strategy.

The value function captures the relationship between the vehicle dynamics (state), the environment and the cost.

Without a cost-to-go function which provides an sufficiently good approximation of the value function associated with the trajectory optimization, important aspects pertaining to performance can be lost. This may be acceptable for vehicle with simpler dynamics but will cause a gap in performance for highly agile vehicles. For example, techniques like the visibility graph do not take the vehicle state into account and therefore cannot capture the spatio-dynamic relationship. Toupet and Mettler [80] approximate the state-dependency using a multi-scale environment decomposition. Toupet and Mettler [111] implement and flight test the receding horizon planner with an improved CTG on a surrogate A-160 rotorcraft. Mettler and Kong [81] use the concept of MA to compute state-dependent cost-to-go map for a receding horizon planning. Dadkhah et al. [28] describes the implementation

and experimental evaluation of this planner and highlight the shortcomings of approximate CTG functions that do not take into account state information.

## 4.6 Other Methods and Issues

### 4.6.1 Ad Hoc Receding Horizon Planning

The advent of more agile robotic platforms has highlighted some limitations of classic robot motion planning techniques. In the Dynamic Window Approach (DWA) [34], a robot's translational and rotational velocity is computed periodically by optimizing a measure of distance to the goal. The velocities are selected from a finite set of admissible velocities, which is determined based on the proximity of immediate obstacles.

The concept was extended to prevent local minima [15] by combining the DWA with a global path planner based on an occupancy grid. Further extensions were implemented to guarantee the stability of the system [84]. Finally, in Hwangbo et al. [49], a similar technique was applied to aerial vehicle motion planning. These latest techniques are conceptually similar to receding horizon optimization but are ad-hoc in their formulation and implementation. These examples can be viewed as a testimony of the convergence of the robotics planning and control concepts.

### 4.6.2 Potential Field Methods and the Navigation Function

Just as in solving a problem with unconstrained dynamics, the potential field can be used to serve as a controller. The same properties apply here: the function is generally simple to compute, but may result in an incomplete planner, and is non-optimal in general. If the potential field is designed properly, it may be used directly as part of a feedback controller. However, if used in that way, care needs to be taken so that the feedback controller is stable. Furthermore, to use such a method, there usually exist certain constraints on the form of the vehicle dynamics.

Such a method is first proposed by Conner et al. [23]: the free configuration space is first decomposed into convex cells and then local control policies are designed for each cell to respect dynamic constraints. The convergence is proved for a double integrator. Belta et al. [9] uses the same idea to solve planning problem of a vehicle whose dynamics can be represented as an affine system within a polyhedral environment. Such configuration space division techniques also enable a marriage of control method and powerful logic-based AI methods as shown in Belta et al. [8]. Conner et al. [24] further extend the idea to convex-bodied nonholonomic wheeled vehicles.

### 4.6.3 Planning in the Presence of Uncertainties

The general problem of planning with uncertainty can be phrased as follows: Given a vehicle with uncertain position information, uncertain environment knowledge (e.g. obstacle locations), and having limited precision in tracking commands, find the best path to the goal.

While the idea of uncertainty in planning has been around for a long time, it has traditionally been considered in the domain of compliant control, where the robot is allowed, or even required, to touch obstacles. The type of situations considered are relevant to problems involving manipulators and gripping, but are not of much use to

vehicle motion planning, where contact with an obstacle is to be avoided at all costs (with the exception of landing or perching). In most of the real world UAV planning problems, the issue of uncertainty in sensing and control is unavoidable.

Connolly [25] presents the Laplacian potential field as a solution to the problem of minimizing collisions of a random walk with an obstacle field. Lazanas [3] solves the problem exactly given regions where there is no uncertainty. Schouwenaars et al. [95] describes a robust Maneuver Automaton which guides the vehicle by taking explicitly into account the uncertainty in the maneuver outcome. Zengin and Dogan [125], which solves the problem of approaching a goal while avoiding SAM sites, uses a state-space search. The RHC framework described earlier can also be employed to deal with uncertainties: the off-line, pre-planned trajectory or approximate cost-to-go function accounts for global convergence based on known knowledge and the online RHC can be used to negotiate with mid-flight uncertainties. For instance, in Kuwata et al. [65], the RHC is used to generate trajectories for a vehicle operating in an environment with atmospheric turbulence.

The field of Simultaneous Localization and Mapping (SLAM) is important in problems of planning with uncertainty. Although the SLAM problem is closely linked with the problem of motion planning with uncertainty, it doesn't immediately address the motion planning issue, so it is not covered here in any detail. Thrun [114] gives a survey of SLAM techniques [115], describes a SLAM example for helicopter flight, and Thrun et al. [116] shows what part SLAM played in their DARPA Grand Challenge entry.

#### 4.6.4 Reactive Planning

The term “reactive planning” refers in general to a broad class of algorithms that use only local knowledge of the obstacle field to plan the trajectory. Reactive algorithms are important in dealing with uncertainty, and run very quickly since no elaborate couplings are involved. In the case where a global obstacle map is not available and obstacle positions are known only within a small radius, a reactive algorithm prevents last-minute collisions by stopping or swerving the vehicle when an obstacle is known to be in the trajectory, which has been planned by a different algorithm. This type of approach is important in many existing practical implementations in order to “patch” an unsound algorithm to ensure that it is sound, as well as to deal with obstacle fields that may change suddenly. However, reactive planners, due to their inability to take the global planning problem into consideration, are seldom used as the sole trajectory generation process. In other words, if only the reactive planner is used, the vehicle may never find a trajectory that will lead to the goal, let alone an optimal one.

The Motion Description Language (MDL), described in Brockett [16], and its extension MDLe [73] can be used to define and design the reactive algorithms. In this framework, a sensor based interrupt (i.e. obstacle detected) will cause the vehicle to switch to another behavior. Hristu-Varsakelis et al. [46] proves that MDLe is formal language.

Zavlangas et al. [123] uses fuzzy logic as a basis for a reactive algorithm. Hui-Zhong et al. [47] describes a different approach, and [43] describes an algorithm that uses learning from the examples of human operators to improve the reactive planner. Call et al. [17] describes visual flow based reactive planning, and Spenko et al. [106] uses a planner based on a high fidelity ground vehicle dynamic model. Geyer and Johnson [41] uses a reactive algorithm in based on laser scanner data in

local coordinates. There are numerous ongoing works on this topic, especially bio-inspired reactive planning algorithms. This can be explained due to the deficiencies of existing planning algorithms to involve sensory information in a principled way so that a complete planning framework results.

## 5 Discussion

In this paper we provided a survey of published motion planning techniques. This survey emphasizes practical methods and provides a general perspective on the particular problems arising with UAVs. UAVs belong to a class of vehicles for which velocity and acceleration constraints are both significant. More agile UAVs even require taking into account the higher order differential constraints associated with the equations of motion, or accounting for aerodynamic effects.

While efficient algorithms exist that are well-characterized for simpler subproblems, such as the problem of motion planning for vehicles not bound by dynamic constraints, an exact solution to a typical UAV problem is often not trivial. Algorithms that can solve these planning problems are frequently too expensive to be used in real time, and when they are tractable, they are not proven to be complete, sound, or optimal.

Given these difficulties, one conclusion of this survey is that a solutions must be chosen to specifically fit the characteristics of the particular planning problem. In some cases, aspects that at first may seem challenging like partial knowledge of the environment or disturbances, can in reality provide opportunities, or at least rationales, for certain forms of approximations.

From the reviewed literature, we find that the issue of uncertainties and robustness in general has not been studied much. Therefore understanding these effects represents a fundamental aspect of determining practical algorithms that are simultaneously computationally efficient, optimal and robust. If we consider feedback control systems, we get a general ideas on how uncertainties or partial knowledge can be addressed by an algorithm. We also see that often relaxing the complexity of the controller can contribute to robustness.

When reviewing the different papers, we find that the majority of the methods surveyed here do not include much discussion about practical implementation. When they do, they often only provide simulation results based on idealized vehicles and operational conditions. So it is inevitable that theoretical work often leaves out important issues. In particular, some of the more expensive, complete algorithms have, to date, never been implemented in code. Among the papers surveyed, most of the practical implementations of UAV guidance have been of the hierachic decoupled control type, or in some cases focus only on the reactive planning needed to avoid obstacles. With the advent of smaller and faster microcontrollers, more sophisticated planners are beginning to be implemented for real-time guidance of vehicles.

The reality is that it is often going to be impossible to generate provably complete or optimal algorithms for problems with differential constraints, therefore more work needs to take place on the development of benchmarks needed to compare algorithms and provide some design standards. Benchmarking will give researchers a way of determining the ability of an algorithm to approximate an optimal solution as well

as an evaluation of its computational complexity. They also provide opportunities to develop more knowledge about the characteristics of the planning problem itself, which ultimately will be most important in the optimality and complexity and the choice of method.

Finally, not all tasks are simple goal-directed guidance problems, but the overall guidance behavior results from the interaction of the vehicle with the environment. In many cases, the information gathered by the UAV about the environment (and the mission) will be useful to improve the trajectory. In robotics these interactive problems are most often found in the context of simultaneous localization and mapping (SLAM). For UAVs there is a stronger emphasis on the system dynamics, thus simultaneous mapping and planning (SMAP) may be more appropriate. Sensors and perception may play as much of a role as the control system or the dynamics in dictating the behavior and performance of the autonomous guidance system. Therefore, future work will have to encompass all of these aspects to achieve a realistic and comprehensive understanding of the algorithms and the development of design principles that will achieve reliable performance and robustness.

**Acknowledgements** This work owes its existence to the funding from the Army Aeroflightdynamics Directorate, and collaboration with the ongoing Autonomous Rotorcraft Project.

## References

1. Agarwal, P.K., Wang, H.: Approximation algorithms for curvature-constrained shortest paths. *SIAM J. Comput.* **30**(6), 1739–1772 (1996)
2. Ahuja, N., Chuang, J.: Shape representation using a generalized potential field model. *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 169–176 (1997)
3. Lazanas, A., Latombe, J.C.: Motion planning with uncertainty: a landmark approach. *Artif. Intell.* **76**(1&2), 287–317 (1995)
4. Barto, A.G., Bradtko, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. *Artif. Intell.* **72**(1), 81–138 (1995)
5. Behnke, S.: Local multiresolution path planning. In: Proceedings of 7th RoboCup International Symposium, pp. 332–343 (2004)
6. Bertsekas, D.: *Dynamic Programming and Optimal Control*, vols. I and II. Athena Scientific, Nashua, NH (2007)
7. Bellingham, J., Richards, A., How, J.P.: Receding horizon control of autonomous aerial vehicles. In: Proceedings of the American Controls Conference, pp. 3741–3746 (2002)
8. Belta, C., Bicci, A., Egerstedt, M., Frazzoli, E., Klavins, E., Pappas, G.: Symbolic control and planning of robotic motion. *Proc. IEEE Robot. Autom. Mag.* **14**, 61–70 (2007)
9. Belta, C., Isler, V., Pappas, G.: Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Trans. Robot.* **21**(5), 864–874 (2005)
10. Betts, J.T.: Survey of numerical methods for trajectory optimization. *J. Guid. Control Dyn.* **21**, 193–207 (1998)
11. Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. *IEEE Int. Conf. Robot. Autom.* **1**, 521–528 (2000)
12. Borenstein, J., Koren, Y.: Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.* **19**(5), 1179–1187 (1989)
13. Borenstein, J., Koren, Y.: The vector field histogram—fast obstacle avoidance for mobile robots. *IEEE J. Robot. Autom.* **7**, 278–288 (1991)
14. Boyle, D.P., Charnitoff, G.E.: Autonomous maneuver tracking for self-piloted vehicles. *J. Guid. Control Dyn.* **22**(1), 58–67 (1999)
15. Brock, O., Khatib, O.: High-speed navigation using the global dynamic window approach. In: *IEEE International Conference on Robotics and Automation*, pp. 341–346 (1999)
16. Brockett, R.W.: Formal languages for motion description and map making. *Proc. Symp. Appl. Math.* **41**, 181–193 (1990)

17. Call, B., Beard, R., Taylor, C., Barber, B.: Obstacle avoidance for unmanned air vehicles using image feature tracking. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO (2006)
18. Canny, J.F.: The Complexity of Robot Motion Planning. MIT, Cambridge (1988)
19. Carlyle, W.M., Wood, R.K.: Lagrangian relaxation and enumeration for solving constrained shortest-path problems. In: Proceedings of the 38th Annual ORSNZ Conference (2007)
20. Caselli, S., Reggiani, M., Rocchi, R.: Heuristic methods for randomized path planning in potential fields. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 426–431 (2001)
21. Choset, H., Burdick, J.: Sensor-based exploration: the hierarchical generalized Voronoi graph. *Int. J. Rob. Res.* **19**(2), 96–125 (2000)
22. Choset, H., Walker, S., Eiamsa-Ard, K., Burdick, J.: Sensor-based exploration: incremental construction of the hierarchical generalized Voronoi graph. *Int. J. Rob. Res.* **19**(2), 126–148 (2000)
23. Conner, D.C., Rizzi, A.A., Choset, H.: Composition of local potential functions for global robot control and navigation. In: Proceedings of 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 3546–3551 (2003)
24. Conner, D.C., Choset, H., Rizzi, A.A.: Integrated planning and control for convex-bodied nonholonomic systems using local feedback control policies. In: Robotics: Science and Systems II, Philadelphia, PA (2006)
25. Connolly, C.I.: Harmonic functions and collision probabilities. *Int. J. Rob. Res.* **16**(4), 497 (1997)
26. Connolly, C.I., Souccar, K., Grupen, R.A.: A Hamiltonian framework for kinodynamic planning and control. *Proc. IEEE Conf. Robot. Autom.* **3**, 2746–2752 (1995)
27. Connolly, C.I., Burns, J.B., Weiss, R.: Path planning using Laplace's equation. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2102–2106 (1990)
28. Dadkhah, N., Kong, Z., Korukanti, V., Mettler, B.: Experimental demonstration of an online trajectory optimization scheme using approximate spatial value functions. In: Proceedings of the Conference on Decision and Control, Shanghai, China (2009)
29. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A\*. *J. ACM* **32**, 505–536 (1985)
30. Dever, C., Mettler, B., Feron, E., Papovic, J., McConley, M.: Trajectory interpolation for parametrized maneuvering and flexible motion planning of autonomous vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, RI (2004)
31. Donald, B., Xavier, P.: Kinodynamic motion planning. *J. ACM* **40**, 1048–1066 (1993)
32. Donald, B.R., Xavier, P.G.: Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open chain manipulators. *Algorithmica* **14**, 958–963 (1995)
33. Ferguson, D., Likhachev, M., Stentz, A.: A Guide to heuristic-based path planning. In: Proceedings of ICAPS Workshop on Planning under Uncertainty for Autonomous Systems, AAAI (2005)
34. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **4**(1), 23–33 (1997)
35. Frazzoli, E., Dahleh, M.A., Feron, E.: A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In: Proceedings of the Conference on Decision and Control, Phoenix, AZ (1999)
36. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Denver, CO (2000)
37. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. *J. Guid. Control Dyn.* **25**(1), 116–129 (2002)
38. Frew, E.: Receding horizon control using random search for UAV navigation with passive, non-cooperative sensing. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, CA (2005)
39. Fujimura, K., Tunii, T.L.: Motion Planning in Dynamic Environments. Springer, Secaucus (1992)
40. Gavrilets, V., Frazzoli, E., Mettler, B., Piedmonte, M., Feron, E.: Aggressive maneuvering of small autonomous helicopters: a human-centered approach. *Int. J. Rob. Res.* **20**(10), 795 (2001)
41. Geyer, M.S., Johnson, E.N.: 3D obstacle avoidance in adversarial environments for unmanned aerial vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO (2006)

42. Guillemin, V., Pollack, A.: Differential Topology. Prentice-Hall, New York (1974)
43. Hamner, B., Singh, S., Scherer, S.: Learning obstacle avoidance parameters from operator behavior. In: Special Issue on Machine Learning Based Robotics in Unstructured Environments, vol. 23, pp. 1037–1058. Wiley InterScience (2006)
44. Howlett, J., Tsenkov, P., Whalley, M., Schulein, G., Takahashi, M.: Flight evaluation of a system for unmanned rotorcraft reactive navigation in uncertain urban environments. Presented at the 63rd Annual Forum of the American Helicopter Society, Virginia Beach, VA (2008)
45. Howlett, J.K., Schulein, G., Mansur, M.H.: A practical approach to obstacle field route planning for unmanned rotorcraft. In: American Helicopter Society 60th Annual Forum Proceedings, Baltimore, MD (2004)
46. Hristu-Varsakelis, D., Egerstedt, M., Krishnaprasad, P.S.: On the structural complexity of the motion description language MDL. In: Proceedings of the 42nd IEEE Conference on Decision and Control, pp. 3360–3365 (2003)
47. Hui-Zhong Zhuang, D.S., Wu, T.: Real-time path planning for mobile robots. In: Proceedings of International Conference on Machine Learning and Cybernetics, vol. 1, pp. 526–531 (2005)
48. Hwang, Y.K., Ahuja, N.: Gross motion planning—a survey. ACM Comput. Surv. **24**, 219–291 (1992)
49. Hwangbo, M., Kuffner, J., Kanade, T.: Efficient two-phase 3D motion planning for small fixed-wing UAVs. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1035–1041 (2007)
50. Ikonen, L.: Pixel queue algorithm for geodesic distance transforms. In: Discrete Geometry for Computer Imagery, pp. 228–239 (2005)
51. Ikonen, L., Toivanen, P.: Shortest routes on varying height surfaces using gray-level distance transforms. Image Vis. Comput. **23**, 133–140 (2005)
52. Jadabaie, A.: Receding horizon control of nonlinear systems: a control Lyapunov function approach. PhD, California Institute of Technology (2000)
53. Junge, O., Marsden, J.E., Ober-blöbaum, S.: Discrete mechanics and optimal control. In: The 16th IFAC World Congress, Prague, Czech (2005)
54. John Canny, A.R.: An exact algorithm for kinodynamic planning in the plane. In: Proceedings of the Sixth Annual Symposium on Computational Geometry, pp. 271–280 (1990)
55. Judd, K.B., McLain, T.W.: Spline based path planning for unmanned air vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Montreal, Canada (2000)
56. Kavraki, L., Nolountzakis, M., Latombe, J.: Analysis of probabilistic roadmaps for path planning. IEEE Trans. Robot. Autom. **14**(1), 166–171 (1998)
57. Kavraki, L., Svestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**, 566–580 (1996)
58. Keith, G., Tait, J., Richards, A.G.: Efficient path optimization with terrain avoidance. In: AIAA Navigation, Guidance and Control Conference, Hilton Head, SC (2007)
59. Khatib, O., Mampey, L.M.: Fonction Decision-Commande d'un Robot Manipulateur, Rep. 2/7156. DERA/CERT, Toulouse, France (1978)
60. Kim, S.H., Bhattacharya, R.: Multi-layer approach for motion planning in obstacle rich environments. In: AIAA Navigation, Guidance and Control Conference, Hilton Head, SC (2007)
61. Kobilarov, M., Desbrun, M., Marsden, J., Sukhatme, G.S.: A discrete geometric optimal control framework for systems with symmetries. In: Robotics: Science and Systems, vol. 3, pp. 1–8 (2007)
62. Kong, Z., Korukanti, V., Mettler B.: Mapping 3D guidance performance using approximate optimal cost-to-go function. In: AIAA Navigation, Guidance and Control Conference, Chicago, IL (2009)
63. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Proceedings of the IEEE Conference on Robotics and Automation, vol. 2, pp. 1398–1404 (1991)
64. Tarabanis, K., Allen, P.K., Tsai, R.Y.: A survey of sensor planning in computer vision. IEEE Trans. Robot. Autom. **11**, 86–104 (1995)
65. Kuwata, Y., Schouwenaars, T., Richards, A., How J.: Robust constrained receding horizon control for trajectory planning. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, CA (2005)
66. Latombe, J.: Robot Motion Planning. Kluwer Academic, Boston (1991)
67. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Tech. Rep (TR 98-11), Computer Science Dept, Iowa State University (1998)

68. LaValle, S.M., Konkimalla, P.: Algorithms for computing numerical optimal feedback motion strategies. *Int. J. Rob. Res.* **20**, 729–752 (2001)
69. LaValle, S.M.: From dynamic programming to RRTs: algorithmic design of feasible trajectories. In: Control Problems in Robotics, pp. 19–37 (2002)
70. LaValle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
71. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 20, no. 5, p. 378 (1999)
72. Marigo, A., Bicchi, A.: Steering driftless nonholonomic systems by control Quanta. In: Proceedings of IEEE International Conference on Decision and Control, pp. 4164–4169 (1998)
73. Manikonda, V., Krishnaprasad, P.S., Hendler, J.: Languages, behaviors, hybrid architectures and motion control. In: Mathematical Control Theory, pp. 199–226 (1998)
74. Milam, M.B., Franz, R., Murray R.M.: Real-time constrained trajectory generation applied to a flight control experiment. In: Proceedings of the IFAC World Congress (2002)
75. Marsden, J.E., West, M.: Discrete mechanics and variational integrators. *Acta Numer.* **10**, 357–514 (2001)
76. Mazer, E., Ahuactzin, J.M.: The Ariadne's clew algorithm. *J. Artif. Intell. Res.* **9**, 1–32 (1998)
77. Menon, P.K., Sweriduk, G.D., Bowers, A.H.: Study of near-optimal endurance-maximizing periodic cruise trajectories. *J. Aircr.* **44**(2), 393–398 (2005)
78. Mettler, B., Valenti, M., Schouwenaars, T., Frazzoli, E., Feron, E.: Rotorcraft motion planning for Agile maneuvering using a maneuver automaton. In: 58th Forum of the American Helicopter Society, Montreal, Canada (2002)
79. Mettler, B., Bachelder, E.: Combining on-and offline optimization techniques for efficient autonomous vehicle's trajectory planning. In: AIAA Guidance, Navigation, and Control Conference, San Francisco, CA (2005)
80. Mettler, B., Toupet, O.: Receding horizon trajectory planning with an environment based cost-to-go function. In: Joint ECC-CDC Conference, Seville, Spain (2005)
81. Mettler, B., Kong, Z.: Receding horizon trajectory optimization with a finite-state value function approximation. In: American Control Conference (ACC), Seattle, WA (2008)
82. Mitchell, J., Papadimitriou, C.: The weighted region problem: finding shortest paths through a weighted planar subdivision. *J. ACM* **38**, 18–73 (1991)
83. Kazhdan, M., Bolitho, M., Hoppe H.: Poisson surface reconstruction. In: Proceedings of Eurographics Symposium on Geometry, pp. 61–70 (2006)
84. Ogren, P., Leonard, N.E.: A convergent dynamic window approach to obstacle avoidance. *IEEE Trans. Robot.* **21**, 188–195 (2005)
85. Piedmonte, M., Feron, E.: Aggressive maneuvering of autonomous aerial vehicles: a human-centered approach. In: Proceedings of the International Symposium on Robotics Research, vol. 9, pp. 413–420 (1999)
86. Prazenica, R.J., Kurdila, A.J., Sharpley, R.C., Evers, J.: Multiresolution and adaptive path planning for maneuver of Micro-Air-Vehicles in urban environments. In: AIAA Navigation, Guidance and Control Conference, San Francisco, CA (2005)
87. Redding, J., Amin, J.N., Boskovic, J.D., Kang, Y., Hedrick, K., Howlett, J., Poll, S.: A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters. In: AIAA Navigation, Guidance and Control Conference, Hilton Head, SC (2007)
88. Reif, J., Sun, Z.: An efficient approximation algorithm for weighted region shortest path problem. In: Proceedings of the 4th Workshop on Algorithmic Foundations of Robotics, p. 191 (2000)
89. Reif, J.H., Wang, H.: Nonuniform discretization for kinodynamic motion planning and its applications. *SIAM J. Comput.* **30**, 161–190 (2000)
90. Rhinehart, M.: Monte Carlo testing of 2- and 3-dimensional route planners for autonomous UAV navigation in urban environments. PhD thesis, University of Minnesota (2008)
91. Richards, A., How, J.: Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: American Control Conference, Anchorage, AK (2002)
92. Rimon, E., Koditschek, D.E.: Exact robot navigation using cost functions: the case of distinct spherical boundaries in  $E^n$ . *Proc. IEEE Int. Conf. Robot. Autom.* **3**, 1791–1796 (1988)
93. Akishita, S., Kawamura, S., Hayashi, K.: Laplace potential for moving obstacle avoidance and approach of a mobile robot. In: Proceedings of the Japan–USA Symposium on Flexible Automation (a Pacific Rim Conference), pp. 139–142 (1990)
94. Scherer, S., Singh, S., Chamberlain, L.J., Saripalli, S.: Flying fast and low among obstacles. In: Proceedings International Conference on Robotics and Automation, pp. 2023–2029 (2007)

95. Schouwenaars, T., Mettler, B., Feron, E., How, J.: Robust motion planning using a maneuver automaton with built-in uncertainties. *Proc. Am. Control Conf.* **3**, 2211–2216 (2003)
96. Schouwenaars, T., Mettler, B., Feron, E., How, J.: Hybrid model for trajectory planning of agile autonomous vehicles. *J. Aero. Comput. Inform. Commun.* **1**(12), 466–478 (2004)
97. Schouwenaars, T., Moor, B.D., Feron, E., How, J.: Mixed integer programming for multi-vehicle path planning. In: European Control Conference (2001)
98. Schwartz, J.T., Yap, C.K.: Advances in Robotics: Algorithmic and Geometric Aspects of Robotics, vol. 1. Lawrence Erlbaum, Philadelphia (1986)
99. Shanmugavel, M., Tsourdos, A., Zbikowski, R., White, B.A.: 3d path planning for multiple UAVs using Pythagorean Hodograph curves. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Hilton Head, South Carolina (2007)
100. Shim, D., Sastry, S.: A situation-aware flight control system design using real-time model predictive control for unmanned autonomous helicopters. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO (2006)
101. Shkel, A., Lumelsky, V.: The jogger's problem: accounting for body dynamics in real-time motion planning. In: Intelligent Robots and Systems, vol. 95, pp. 441–447 (1995)
102. Sinopoli, B., Micheli, M., Donato, G., Koo, T.J.: Vision based navigation for an unmanned aerial vehicle. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1757–1765 (2001)
103. Singh, L., Fuller, J.: Trajectory generation for a UAV in urban terrain, using non-linear MPC. In: Proceedings of American Control Conference, pp. 2301–2308 (2001)
104. Slater, G.L.: Guidance on maneuvering flight paths for rotary wing aircraft. In: AIAA Guidance, Navigation and Control Conference, vol. 1, pp. 793–803. Monterey, CA (1987)
105. Soucy, M., Payeur, P.: Robot path planning with multiresolution probabilistic representations: a comparative study. In: Proceedings of the Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Canada (2004)
106. Spenko, K., Overholt, J., Iagnemma, K.: High speed hazard avoidance for unmanned ground vehicles in emergency situations. In: 25th Army Science Conference, Orlando, FL (2006)
107. Stopp, A., Riethmüller, T.: Fast reactive path planning by 2D and 3D multi-layer spatial grids for mobile robot navigation. In: Proceedings of the 1995 IEEE International Symposium on Intelligent Control, pp. 545–550 (1995)
108. Suzuki, S., Komatsu, Y., Yonezawa, S., Masui, K., Tomita, H.: Online four-dimensional flight trajectory search and its flight testing. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, CA (2005)
109. Švestka, P., Overmars, M.H.: Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In: Proceedings of the IEEE Conference on Robotics and Automation, Nagoya Japan (1995)
110. Takahashi, M.D., Schulein, G., Whalley, M.: Flight Control Law Design and Development for an Autonomous Rotorcraft. American Helicopter Society, Virginia Beach, VA (2008)
111. Toupet, O., Mettler, B.: Design and flight test evaluation of guidance system for autonomous rotorcraft. In: Proceedings of the AIAA Guidance, Navigation and Control Conference, Keystone, CO (2006)
112. Toussaint, G.J., Basar, T., Bullo, F.: Motion planning for nonlinear underactuated vehicles using H-infinity techniques. In: Proceedings of the 2001 American Control Conference, vol. 5, pp. 4097–4102 (2001)
113. Tsenkov, P., Howlett, J.K., Whalley, M., Schulein, G., Takahashi, M., Rhinehart, M.H., Mettler, B.: A system for 3d autonomous rotorcraft navigation in urban environments. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI (2008)
114. Thrun, S.: Robotic mapping: a survey. In: Exploring Artificial Intelligence in the New Millennium. Morgan Kaufmann, San Francisco (2002)
115. Thrun, S., Diel, M., Hahnel, D.: Scan alignment and 3-D surface modeling with a helicopter platform. In: International Conference on Field and Service Robotics, Japan (2003)
116. Thrun, S., Montemerlo, M.: The graph slam algorithm with applications to large-scale mapping of urban structures. *Int. J. Rob. Res.* **25**(5–6), 403 (2006)
117. Vandapel, N., Kuffner, J., Amidi, O.: Planning 3-D path networks in unstructured environments. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 4624–4629 (2005)
118. Wei, S., Zefran, M.: Smooth path planning and control for mobile robots. In: Proceedings of IEEE International Conference on Networking, Sensing and Control, pp. 894–899 (2005)

119. Whalley, M., Freed, M., Harris, R., Takahashi, M., Schulein, G., Howlett, J.K.: Design, integration, and flight test results for an autonomous surveillance helicopter. In: Proceedings of the AHS International Specialists Meeting on Unmanned Rotorcraft, Candler, AZ (2005)
120. Yakimenko, O.A.: Direct method for rapid prototyping of near-optimal aircraft trajectories. *J. Guid. Control Dyn.* **23**(5), 865 (2000)
121. Yang, G., Kapila, V.: Optimal path planning for unmanned air vehicles with kinematic and tactical constraints. In: IEEE Conference on Decision and Control, Las Vegas, NV (2002)
122. Yang, H.I., Zhao, Y.J.: Trajectory planning for autonomous aerospace vehicles amid known obstacles and conflicts. *J. Guid. Control Dyn.* **27**(6), 997–1008 (2004)
123. Zavlangas, P.G., Tzafestas, P.S.G., Althoefer, D.K.: Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots. In: European Symposium on Intelligent Techniques, Aachen, Germany (2000)
124. Zelek, J.S.: Dynamic path planning. In: IEEE International Conference on Systems, Man and Cybernetics, Vancouver, Canada (1995)
125. Zengin, U., Dogan, A.: Probabilistic trajectory planning for UAVs in dynamic environments. In: AIAA 3rd “Unmanned Unlimited” Technical Conference, Workshop and Exhibit, Chicago, IL (2004)

# An Efficient Path Planning and Control Algorithm for RUAV's in Unknown and Cluttered Environments

Kwangjin Yang · Seng Keat Gan · Salah Sukkarieh

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 27 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper presents an efficient planning and execution algorithm for the navigation of an autonomous rotary wing UAV (RUAV) manoeuvering in an unknown and cluttered environment. A Rapidly-exploring Random Tree (RRT) variant is used for the generation of a collision free path and linear Model Predictive Control(MPC) is applied to follow this path. The guidance errors are mapped to the states of the linear MPC structure by using the nonlinear kinematic equations. The proposed path planning algorithm considers the run time of the planning stage explicitly and generates a continuous curvature path whenever replanning occurs. Simulation results show that the RUAV with the proposed methodology successfully achieves autonomous navigation regardless of its lack of prior information about the environment.

**Keywords** Dynamic path planning · Model predictive control · Rapidly-exploring random trees · Small-scale helicopter

---

This work is supported in part by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

K. Yang (✉) · S. K. Gan · S. Sukkarieh  
Australian Centre for Field Robotics, University of Sydney,  
Sydney, New South Wales 2006, Australia  
e-mail: k.yang@acfr.usyd.edu.au

S. K. Gan  
e-mail: s.gan@acfr.usyd.edu.au

S. Sukkarieh  
e-mail: salah@acfr.usyd.edu.au

## 1 Introduction

Navigation through an unknown environment has been of great interest in field robotics [1–6]. To complete the mission successfully in such environments, functionalities such as obstacle map building, dynamic path planning, and path following need to be implemented. The path planning problem for UAVs is difficult since these vehicles have fast and complicated dynamics and are compounded by the issues of real time navigation in 3D space. There are several considerations for an ideal path planner including: optimality; completeness; and computational complexity. There is a natural trade off between these elements [7]. For UAV applications computational complexity is the most important requirement. If previously unknown obstacles are detected, the UAV has to replan the path in real time to avoid these obstacles. The computational time of deterministic and complete algorithms grows exponentially with the dimension of the configuration space, and hence these algorithms do not provide an adequate solution for real-time UAV path planning in unknown natural environments [8]. Recently, sample-based motion planning has gained much interest as an alternative to complete motion planning methods. Among them Rapidly-exploring Random Trees (RRT) has been used successfully to demonstrate efficient autonomous navigation in unknown environments [4, 6, 9, 11]. In this research RRT is used for the generation of a collision free piecewise linear path and a path smoothing algorithm is applied which satisfies curvature continuity and non-holonomic constraints.

Accurate path following control is important, especially when the RUAV flies in an environment cluttered with obstacles. In addition, it is also important to guarantee the smooth transition between the previous path and the new path when replanning occurs. However, it is difficult to design an accurate path following controller of a helicopter because it is a poorly damped system and its dynamics are highly nonlinear and unstable. A cascaded type controller is favored for helicopter control [12, 13]. The rule of thumb of this controller is that the inner loop control system must have sufficiently large bandwidth to track the commands that are generated from the outer loop guidance system. However, since the two systems are effectively coupled, stability and adequate performance of the combined systems can not be guaranteed [14].

Instead of this bandwidth separation scheme, an integrated guidance and control strategy such as Model Predictive Control (MPC) can achieve greater performance and stability. MPC employs an explicit model of the plant to predict the future output behavior by minimizing tracking error over a future horizon [15]. However standard linear MPC is not applicable for trajectory tracking since the position states are not included in the state space model. In this paper a novel explicit linear MPC path following control algorithm is used. The main advantage of this approach is that an explicit state feedback law avoids the need for on-line optimization [16]. Due to its low complexity and high reliability this method provides a good alternative for UAV control. To solve this path following control problem in a linear MPC framework we map the guidance errors to the states of the linear MPC using nonlinear kinematic equations. With this technique the guidance errors are converted to the reference values of the states which are controllable in a linear MPC formulation.

If new obstacles are detected and if they are located within the safety zone then the RRT path planner must replan the path to generate a collision free path. In this

research the run time of the path planning algorithm is considered explicitly when replanning occurs. Furthermore the replanned path we propose also preserves the continuous curvature property which smoothes the transition of the UAV motion from the old path to the new path.

The paper is organized as follows: Section 2 describes the path planning algorithm, and the MPC path following control is presented in Section 3. Section 4 illustrates dynamic path replanning and simulation results are given in Section 5. Conclusion is presented in Section 6.

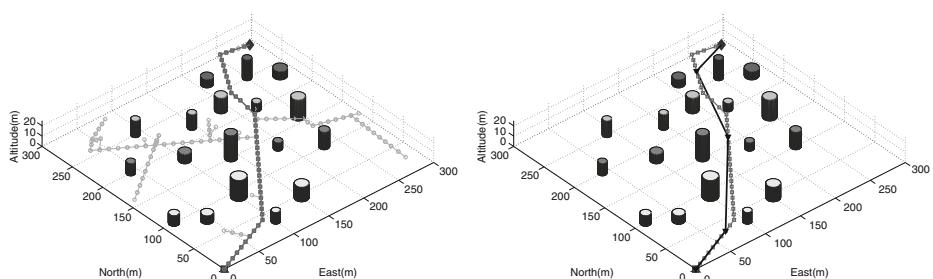
## 2 Path Planning

### 2.1 RRT Path Planning

RRT was first suggested in [17] as an alternative to complete path planning in high degree of freedom situations. The RRT algorithm operates as follows. First, a position  $x_{rand}$  is chosen at random from within the workspace, and this point is compared with existing tree nodes to find the closest point in the tree,  $x_{near}$ . A line is drawn connecting  $x_{near}$  to  $x_{rand}$ , and a new point  $x_{new}$  is generated along this ray at a fixed distance  $d$  from  $x_{near}$ . If there are no collisions in the interval between  $x_{near}$  and  $x_{new}$  then the latter is added to the tree.

Figure 1 (Left) shows the path planning result of the RRT Algorithm. The thin lines are all trees generated by the algorithm and the thick line is the shortest path which connects the starting point and the target point. Even though RRT is an effective and computationally efficient tool for complex online motion planning, the solution produced is simply a random exploration of the space. We extend upon the standard approach to find a path that eliminates most extraneous nodes produced by the standard RRT algorithm.

Let the original path of nodes from start to goal point be denoted  $\{x_1, \dots, x_N\}$ , such that  $x_N$  is the goal location. Let the pruned path be initially an empty set, and let  $j = N$ . The pruning operation is as follows. First add  $x_j$  to the pruned path. Then for each  $i \in [1..j-1]$ , check the line between  $(x_i, x_j)$  for a collision, stopping on the first  $x_i$  without collision. Let  $j = i$ , add  $x_j$  to the pruned path, and repeat the process



**Fig. 1** (Left) Path planning using the RRT algorithm. (Right) Path pruning algorithm: the path consists of 43 nodes but after removing redundant waypoints, the number of nodes is reduced to only 3

until a complete path is generated. This method will remove unnecessary waypoints very quickly. Figure 1 (Right) shows the result of the path pruning algorithm applied to the initial RRT path. The path has 43 nodes between the starting and end points initially but the number of nodes is reduced to only 3 after the redundant waypoints are pruned.

## 2.2 Path Smoothing

The path in Fig. 1 (Right) is piecewise linear and not suitable for a RUAV with kinematic and dynamic constraints. To achieve curvature continuity the position, heading, and the curvature values of connecting linear paths must be the same at the joints.

Let the degree  $n$  Bézier curve with  $n + 1$  control points  $(P_0, P_1, \dots, P_n)$  be defined as [18]

$$P(s) = \sum_{i=0}^n P_i B_{n,i}(s) \quad (1)$$

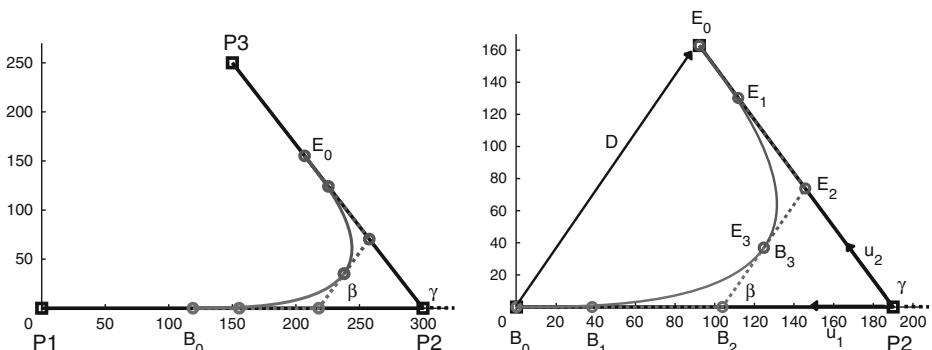
where the coefficients  $B_{i,n}(s)$  are named Berstein polynomials and are defined as follows:

$$B_{n,i}(s) = \binom{n}{i} s^i (1-s)^{n-i} \quad (2)$$

The continuous curvature curve which satisfies the maximum curvature constraint can be obtained using the two spiral curves as shown in Fig. 2.

The first curve consists of following the four control points:

$$\begin{aligned} B_0 &= P_2 + d \cdot u_1, & B_1 &= B_0 - g_b \cdot u_1, \\ B_2 &= B_1 - h_b \cdot u_1, & B_3 &= B_2 + k_b \cdot u_d \end{aligned} \quad (3)$$



**Fig. 2** (Left) G2CBS path smoothing result between two line segments. (Right) Curvature continuity condition to connecting two lines

The second curve consists of the following four control points:

$$\begin{aligned} E_0 &= P_2 + d \cdot u_2, & E_1 &= E_0 - g_e \cdot u_2, \\ E_2 &= E_1 - h_e \cdot u_2, & E_3 &= E_2 - k_e \cdot u_d \end{aligned} \quad (4)$$

where

$$\begin{aligned} h_b &= h_e = c_3 \cdot d, \\ g_b &= g_e = c_1 c_3 \cdot d, \\ k_b &= k_e = \frac{6c_3 \cos \beta}{c_1 + 4} \cdot d \end{aligned} \quad (5)$$

Here  $u_1$  is a unit vector between  $P_2$  and  $P_1$ ,  $u_2$  is that of  $P_3$  and  $P_2$  and  $u_d$  is a unit vector between  $B_2$  and  $E_2$ , and  $d$  is a length between  $B_0$  and  $P_2$ , and  $\beta = \frac{\gamma}{2}$ , and  $c_1 = \frac{2}{5}(\sqrt{6} - 1)$ , and  $c_2 = 7.2364$ , and  $c_3 = \frac{c_1+4}{c_2+6}$ .

The only design variable to generate a continuous curvature path is  $d$  as can be seen in Eqs. 3–5. If  $d$  is selected as Eq. 6 [19], then the path generates a  $G^2$  path which satisfies the maximum curvature constraint.

$$d = \left( \frac{(c_1 + 4)^2}{54c_3} \right) \cdot \frac{\sin \beta}{\kappa_{max} \cdot \cos^2 \beta} \quad (6)$$

where  $\kappa_{max}$  is a maximum curvature value.

Figure 3 shows smoothed path result (Left) and the curvature of the path (Right).

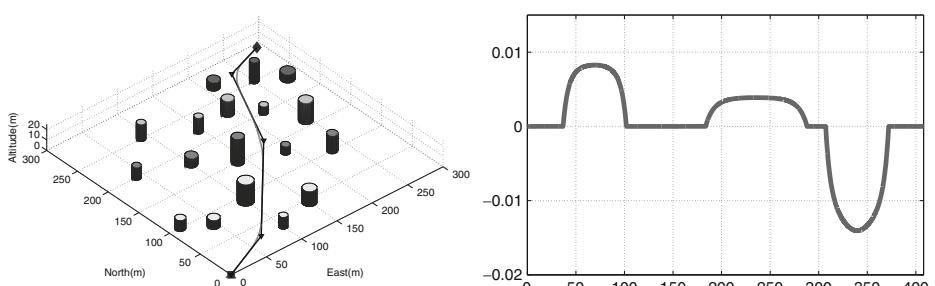
### 3 Path Following Control

#### 3.1 Helicopter Model

In this research a six DOF rigid body model suggested by Mettler et al. [20] was used. It expresses a small-size unmanned helicopter with system equations of first order. This model captures the nonlinear helicopter dynamics with very high accuracy and has been applied successfully in real applications [21].

The system equation is given as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (7)$$



**Fig. 3** (Left) Continuous curvature path smoothing. (Right) Curvature of the path

This equation can be separated into the dynamics and kinematics of the helicopter.

$$\begin{bmatrix} \dot{\mathbf{x}}^D \\ \dot{\mathbf{x}}^A \\ \dot{\mathbf{x}}^S \end{bmatrix} = \begin{bmatrix} \mathbf{Ax}^D + \mathbf{Bu} \\ C_B^S \omega \\ T_B^S \dot{\mathbf{x}}^B \end{bmatrix} \quad (8)$$

where

$$\begin{aligned} \mathbf{x}^D &= [u, v, p, q, a, b, w, r, r_{fb}]^T \\ \mathbf{x}^A &= [\phi, \theta, \psi]^T, \quad \mathbf{x}^S = [x^S, y^S, z^S]^T \\ \omega &= [p, q, r]^T, \quad \mathbf{u} = [u_{lat}, u_{lon}, u_{col}, u_{tr}] \\ C_B^S &= \begin{bmatrix} 1, \sin \phi \tan \theta, \cos \phi \tan \theta \\ 0, \cos \phi, -\sin \phi \\ 0, \sin \phi \sec \theta, \cos \phi \sec \theta \end{bmatrix} \\ T_B^S &= \begin{bmatrix} c\theta c\psi, s\phi s\theta c\psi - c\phi s\psi, c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi, s\phi s\theta c\psi + c\phi c\psi, c\phi s\theta s\psi - s\phi c\psi \\ -s\theta, s\phi c\theta, c\phi c\theta \end{bmatrix} \end{aligned}$$

Here  $\mathbf{x}^D$  represents the dynamic states and  $\mathbf{x}^A$ ,  $\mathbf{x}^S$  represents the kinematic states of the helicopter dynamics. The superscript  $S$  and  $B$  denote spatial and body coordinates,  $(u, v, w)$  are translational body velocities,  $(p, q, r)$  are body rotational roll, pitch, yaw rates,  $(\phi, \theta, \psi)$  are roll, pitch, yaw angles,  $(a, b)$  are the longitudinal, and lateral flapping angles of the main rotor,  $r_{fb}$  is a yaw rate gyro feedback state, and  $c\theta, s\theta$  denotes  $\cos \theta$  and  $\sin \theta$ .

There are four control inputs.  $u_{lon}$  and  $u_{lat}$  are cyclic pitch controls which are used to change the direction of the thrust vector controlling the pitch and roll, and  $x$ ,  $y$  position and velocity.  $u_{col}$  is used for the generation of the thrust controlling the heave velocity and position.  $u_{tr}$  is used to counteract the torque effect of the main rotor and heading control.

### 3.2 PID Control

To judge the performance of the MPC implementation a PID control method is developed based on the work in [21]. The first step of designing the controller is to stabilize the attitude dynamics. Small scale helicopters are usually fitted with a stabilizer bar and an active yaw damping system. These two systems act as dynamic stability augmentations and have major effects on the vehicle response. The stabilizer bar acts as a lagged rate feedback in the pitch and roll axes and the angular rate gyro acts as a negative feedback of the helicopter heading rate. Therefore we can assume that small scale helicopters are naturally equipped with roll-pitch-yaw rate feedback controllers. Because of this property the attitude dynamics can be stabilized with only proportional feedback of the roll-pitch-yaw angle.

After stabilizing the attitude, the longitudinal and lateral velocity dynamics can be also stabilized by proportional control. The proportional control is sufficient for the heave velocity dynamics control because it is intrinsically stable.

Finally, position control also can be achieved by proportional and derivative control after stabilizing the velocity dynamics. To remove steady state error integral

controls are added to the position control. The PID control laws for path following control are shown in Eq. 9.

$$\begin{aligned} u_{lon} &= -K_\theta e_\theta - K_u e_u - K_x e_x - K_{Ix} \int e_x \\ u_{lat} &= -K_\phi e_\phi - K_v e_v - K_y e_y - K_{Iy} \int e_y \\ u_{col} &= -K_w e_w - K_z e_z - K_{Iz} \int e_z \\ u_{tr} &= -K_\psi e_\psi \end{aligned} \quad (9)$$

### 3.3 Model Predictive Control

To formulate the explicit MPC structure consider the following linear system:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du} \end{aligned} \quad (10)$$

The states of the system at time  $t$  is obtained by the following convolution integral,

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_{\tau=0}^t \Phi(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau \quad (11)$$

where

$$\Phi(t) = e^{At} \quad (12)$$

The states at  $n$  prediction time step later is,

$$\mathbf{x}(t_n) = \Phi(t_n)\mathbf{x}(0) + \sum_{k=1}^n \Phi(t_n - t_k) \times A^{-1} [\Phi(t_k - t_{k-1}) - I] \mathbf{B}\mathbf{u}(t_{k-1}) \quad (13)$$

By substituting this to the state space output equation in Eq. 10 yields,

$$\begin{aligned} \mathbf{y}(t_n) &= C\Phi(t_n)\mathbf{x}(0) + \sum_{k=1}^n C\Phi(t_n - t_k) \\ &\quad \times A^{-1} [\Phi(t_k - t_{k-1}) - I] \mathbf{B}\mathbf{u}(t_{k-1}) + D\mathbf{u}(t_n) \end{aligned} \quad (14)$$

This simplifies to,

$$\mathbf{y}(t_n) = \theta_n \mathbf{x}(0) + \sum_{k=1}^n K_{n,k} \mathbf{u}_{k-1} + D\mathbf{u}_n \quad (15)$$

where

$$\begin{aligned} \theta_n &= C\Phi(t_n) \\ K_{n,k} &= C\Phi(t_n - t_k) A^{-1} [\Phi(t_k - t_{k-1}) - I] B \end{aligned} \quad (16)$$

By combining  $K_{i,j}$  and  $D$ , we can obtain the following equation

$$\mathcal{Y} = \Theta \mathbf{x}(0) + \Psi \mathcal{U} \quad (17)$$

where  $\mathcal{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n]^T$  and  $\Theta = [\theta_1, \theta_2, \dots, \theta_n]^T$  and  $\mathcal{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_n]^T$  and  $\Psi$  is,

$$\begin{bmatrix} K_{1,1} & D & 0 & 0 & \cdots & \cdots & 0 \\ K_{2,1} & K_{2,2} & D & 0 & \cdots & \cdots & 0 \\ K_{3,1} & K_{3,2} & K_{3,3} & D & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K_{n,1} & K_{n,2} & K_{n,3} & \cdots & K_{n,n-1} & K_{n,n} & D \end{bmatrix}$$

For path following control, we define the following cost function,

$$J = \sum_{k=1}^n (\mathbf{y}_d(k)^T Q \mathbf{y}_d(k) + \mathbf{u}(k)^T R \mathbf{u}(k)) \quad (18)$$

where  $\mathbf{y}_d(k) = \mathbf{y}_r(k) - \mathbf{y}(k)$ . Here  $\mathbf{y}_d(k)$  represents an error between the reference states ( $\mathbf{y}_r$ ) and the current helicopter states ( $\mathbf{y}$ ).  $Q$  and  $R$  are positive definite diagonal weighting matrices. This cost function can be simplified as follows:

$$J = Y_d^T Q Y_d + U^T R U \quad (19)$$

The explicit form of the optimal control input which minimizes the above cost function can be obtained by the partial derivative of  $J$  with respect to control input,  $\frac{\partial J}{\partial U} = 0$ , which yields:

$$U = (\Psi^T Q \Psi + \mathcal{R})^{-1} \Psi^T Q E \quad (20)$$

where  $E$  is the error between system future response of the RUAV and the reference path.

$$E = Y_r - Q X \quad (21)$$

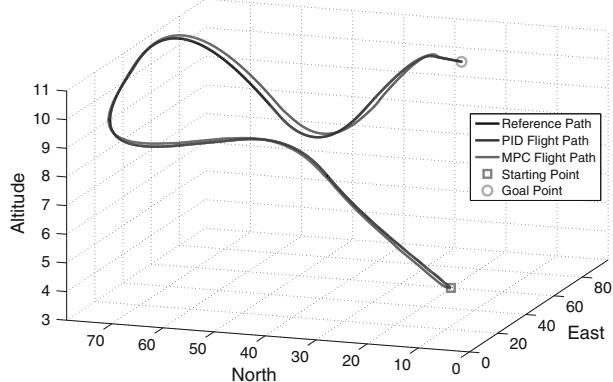
This MPC tracking controller minimizes the tracking errors and control inputs. However this linear MPC can not be directly applicable to path following control since the states of the MPC do not contain the position and heading states. Therefore to control the position and heading of the helicopter these guidance errors have to be appropriately mapped into the MPC controllable states. It is reasonable to select the  $u, v, w$  and  $r$  states to control  $x, y, z$  position and heading ( $\psi$ ). We convert the guidance errors ( $e_x, e_y, e_z, e_\psi$ ) to the reference states of the MPC ( $u_r, v_r, w_r, r_r$ ). Due to the mapping of these guidance errors to the reference states of the MPC the position and heading errors can be optimally controlled.

### 3.4 Path Following Simulations

In this section, we compare the path following performance between the MPC algorithm presented here and the multi-loop PID control presented above. Initially the helicopter is hovering at position (0,0,5). The mission is to follow the path with a constant forward velocity of 5m/s. In this simulation a 3 prediction horizon is used in the MPC path following control since it is sufficient to get a accurate tracking performance.

Figure 4 shows the path tracking simulation result of the PID and the MPC control and Fig. 5 shows the guidance error. As we can see in Fig. 5, the position and altitude

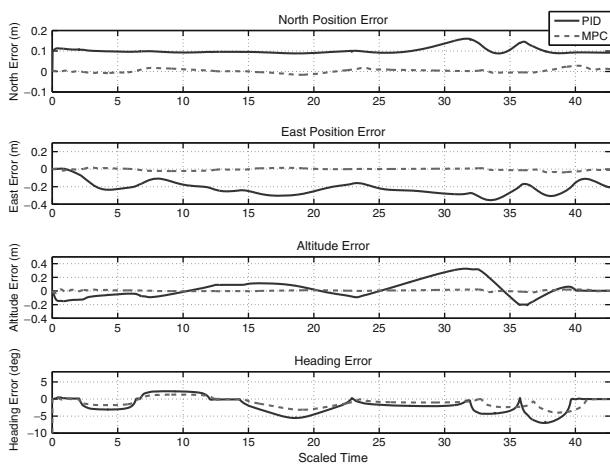
**Fig. 4** 3D path following results of the PID control and the MPC



errors of the MPC is almost zero but in the PID control case there exists relatively large errors. The heading error of MPC is also smaller than PID but the difference is not so big as the position errors. This is because the path has a smooth curvature profile and so the PID control can follow the reference heading well.

To compare the performance quantitatively the  $\mathcal{L}_2$  norm of the guidance errors is calculated and presented in Table 1. The position error  $\mathcal{L}_2$  norms of the PID control is more than 8 times larger than that of MPC with little difference in the heading error. Since the reference path is originally generated in a cluttered environment [19], a larger error increases the possibility of a collision. Therefore the maximum tracking error is also crucial in path following. Table 2 shows the  $\mathcal{L}_{\infty}$  norms of the tracking error of the MPC and the PID control. Similar to the error  $\mathcal{L}_2$  norms, the maximum error of PID control is more than 8 times larger than MPC except for the heading error.

**Fig. 5** Position and heading errors of the PID control and the MPC



**Table 1** Comparison of the  $\mathcal{L}_2$  norm path tracking errors

	North error(m)	East error(m)	Height error(m)	Heading error(rad)
PID control	4.79	10.59	5.81	2.41
MPC	0.60	0.81	0.68	1.86

## 4 Dynamic Path Planning

If new obstacles are detected within the safety zone the RRT path planner replans the path to generate a collision free path. Moreover, the replanned path preserves the continuous curvature property which makes for smooth transition of the UAV motion from the old path to the new path.

### 4.1 Decreasing Curvature Path Generation

The replanning can be triggered at an arbitrary point on the path whenever the helicopter detects new obstacles. Therefore the curvature of the path at that point may also have an arbitrary value. If the path planner does not consider the curvature at the replanning point then there will exist a curvature discontinuity between the old path and the new replanned path. This will cause a discontinuity in the centripetal acceleration which will result in rapid change of heading control and an increase in tracking error. To generate a continuous curvature path when replanning occurs a decreasing curvature path is proposed which monotonically increases or decreases the curvature to zero at the end of the path segment.

In [10] the condition proposed to generate a cubic Bézier spiral curve is as follows:

$$\frac{k}{h} \geq c_1, \quad \frac{g}{h} \leq \frac{6 \cos \theta}{\frac{g}{h} + 4} \quad \text{and} \quad 0 < \theta < \frac{\pi}{2} \quad (22)$$

where  $g, h, k$  are the lengths between two points and  $\theta$  is the angle between the  $\overline{P_0 P_1}$  line and the  $\overline{P_1 P_3}$  line as is shown in Fig. 6.

Also the conditions to generate a decreasing curvature curve is suggested as follows:

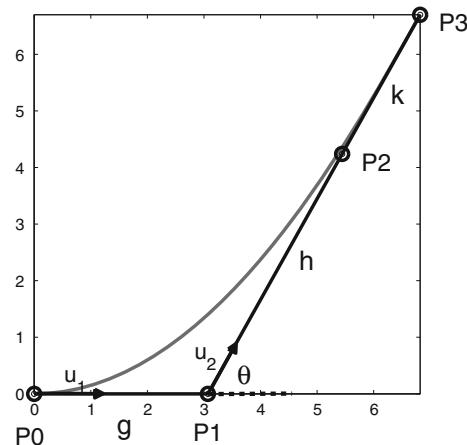
$$P_{3x} > 0, \quad P_{3y} \leq c_y \cdot \kappa_0 \cdot P_{3x}^2 \quad (23)$$

where  $c_y = 0.4869$ , and  $P_{3x}, P_{3y}$  are the  $x$  and  $y$  positions of  $P_3$  and  $\kappa_0$  is the curvature at  $P_0$ .

**Table 2** Comparison of the  $\mathcal{L}_{\infty}$  norm path tracking errors

	North error(m)	East error(m)	Height error(m)	Heading error(rad)
PID control	0.161	0.354	0.327	0.122
MPC	0.028	0.036	0.030	0.071

**Fig. 6** Decreasing curvature path generation



To decide on the four control points for generating the decreasing curvature path both Eqs. 22 and 23 must be satisfied. The curvature at  $P_0$  can be calculated using Eq. 24 then

$$\kappa_0 = \frac{|3\vec{g} \times (6\vec{h} - 6\vec{g})|}{|3\vec{g}|^3} = \frac{2h \sin \theta}{3g^2} \quad (24)$$

$h$  can be expressed by  $\kappa_0$ ,  $g$  and  $\theta$ .

$$h = \frac{3\kappa_0 g^2}{2 \sin \theta} \quad (25)$$

Therefore from the geometry of Fig. 6,  $\theta$ ,  $h$ ,  $k$  can be expressed as follows:

$$\theta = \text{atan} \left( \frac{P_{3y}}{P_{3x} - g} \right), \quad (26)$$

$$h = \frac{3\kappa_0 \cdot g^2 \sqrt{P_{3y}^2 + (P_{3x} - g)^2}}{2 \cdot P_{3y}} \quad (27)$$

$$k = \frac{(2P_{3y} - 3\kappa_0 \cdot g^2) \sqrt{P_{3y}^2 + (P_{3x} - g)^2}}{2 \cdot P_{3y}} \quad (28)$$

There are an infinite number of paths which satisfy Eqs. 22 and 23. however by making  $k = c_1 \cdot h$  then the shortest path is selected as seen in Eq. 22.

$$k = c_1 \cdot h \quad (29)$$

Since  $P_{3y}$  can be obtained using the geometry in Fig. 6,

$$P_{3y} = (h + k) \sin \theta = (c_1 + 1) \cdot h \sin \theta \quad (30)$$

$h$  and  $k$  can be expressed by  $P_{3y}$  as follows:

$$h = \frac{P_{3y}}{(c_1 + 1) \cdot \sin \theta}, \quad k = \frac{c_1 \cdot P_{3y}}{(c_1 + 1) \cdot \sin \theta} \quad (31)$$

If we substitute Eqs. 27 and 28 into Eq. 29,

$$\frac{k}{h} = c_1 = \frac{\frac{(2P_{3y} - 3\kappa_0 \cdot g^2)\sqrt{P_{3y}^2 + (P_{3x} - g)^2}}{2P_{3y}}}{\frac{3\kappa_0 \cdot g^2 \sqrt{P_{3y}^2 + (P_{3x} - g)^2}}{2P_{3y}}} = \frac{2P_{3y} - 3\kappa_0 \cdot g^2}{3\kappa_0 \cdot g^2} \quad (32)$$

$g$  can be obtained as follows:

$$g = \sqrt{\frac{2P_{3y}}{3\kappa_0(c_1 + 1)}} \quad (33)$$

Then the only variable remaining in  $g$ ,  $h$ , and  $k$  (Eqs. 31 and 33) is  $P_{3y}$ . We can chose any value smaller than  $c_y \cdot \kappa_0 \cdot P_{3x}^2$  as shown in Eq. 23 but it is better to use small  $P_{3y}$  to minimize the path length. However if  $P_{3y}$  is too small the curvature will change very sharply shown in Fig. 7. To avoid this situation we choose  $P_{3y}$  which is proportional to the curvature.

$$P_{3y} = c_4 \kappa_0 P_{3x}^2 = \left( \frac{\kappa_0}{\kappa_{max}} c_y \right) \kappa_0 P_{3x}^2 \quad (34)$$

where  $\kappa_{max}$  is the maximum allowable curvature of the path. If the curvature at  $P_0$  is large as in Fig. 6 then  $P_{3y}$  will have a large value which will tend to bend the path and if the curvature is small then  $P_{3y}$  will also be small which in turn generates an almost a straight line.

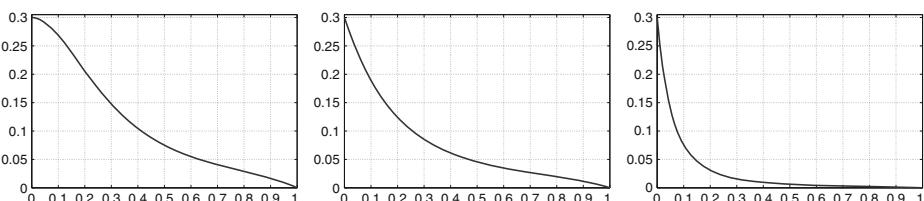
Finally, four control points generating a decreasing curvature path can be obtained as follows:

$$\begin{aligned} P_0 &= (0, 0), & P_1 &= P_0 + g \cdot u_1, \\ P_2 &= P_1 + h \cdot u_2, & P_3 &= P_2 + k \cdot u_2 \end{aligned} \quad (35)$$

where

$$g = \sqrt{\frac{2P_{3y}}{3\kappa_0(c_1 + 1)}}, \quad h = \frac{P_{3y}}{(c_1 + 1) \cdot \sin \theta}, \quad k = c_1 h, \quad \text{and} \quad P_{3y} = c_4 \kappa_0 P_{3x}^2 \quad (36)$$

Therefore if  $P_{3x}$  is determined then the other parameters can be determined based on  $P_{3x}$  as can be seen in Eqs. 35 and 36. If the computational time of the path



**Fig. 7** Curvature comparison: the curvature at  $P_0$  is 0.3 and  $c_4$  is chosen with different values;  $c_4 = 0.48$  (Left),  $c_4 = 0.20$  (Middle),  $c_4 = 0.05$  (Right)

replanning is specified as  $T_{rp}$  and the helicopter flies with  $V$ m/s then the distance which helicopter flies during the replanning time is

$$D_{rp} = V \cdot T_{rp} \quad (37)$$

Therefore  $P_{3x}$  must be decided which makes the length of the planned path  $D_{rp}$ . Generally, a numerical method is used to calculate the path length in Bézier curves [22] but Gravesen suggested an algorithm which can approximate the length of the Bézier curves in [23].

$$L_{bc} = \frac{2 \cdot L_c + (n - 1) \cdot L_p}{n + 1} \quad (38)$$

where  $L_c$  is the length between  $P_0$  and  $P_3$ ,  $L_p$  is the sum of all polygon lengths, and  $n$  is the order of the Bézier curve. In our application  $n = 3$  since the cubic Bézier curves is used and  $L_p = g + h + k$ . Therefore, Eq. 38 can be modified

$$L_{bc} = \frac{L_c + g + h + k}{2} \quad (39)$$

If we substitute the values obtained from Eqs. 36 into 39 then it can also be modified to:

$$L_{bc} = \frac{\sqrt{P_{3x}^2 + P_{3y}^2} + \sqrt{\frac{2P_{3y}}{3(c_1+1)\cdot\kappa_0}} + \frac{P_{3y}}{\sin\theta}}{2} \quad (40)$$

Since  $P_{3y} = c_4 \cdot \kappa_0 \cdot P_{3x}^2$  then

$$\tan\theta = \frac{P_{3y}}{P_{3x} - g} = \left( \frac{c_4}{1 - \sqrt{\frac{2c_4}{3(c_1+1)}}} \right) \kappa_0 P_{3x} = c_5 \kappa_0 P_{3x} \quad (41)$$

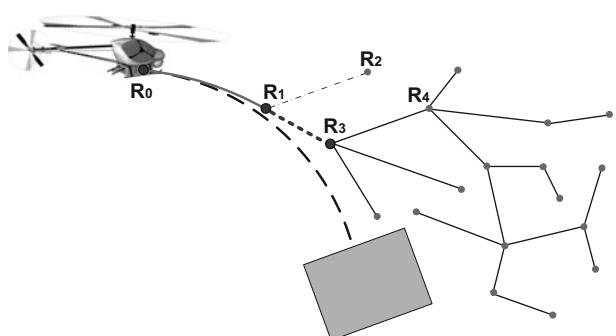
and therefore

$$\sin\theta = \frac{c_5 \kappa_0 P_{3x}}{\sqrt{1 + (c_5 \kappa_0 P_{3x})^2}} \quad (42)$$

Finally Eq. 40 becomes:

$$L_{bc} = \frac{\sqrt{1 + (c_4 \kappa_0 P_{3x})^2} + \sqrt{\frac{2c_4}{3(c_1+1)}} + \frac{c_4 \sqrt{1 + (c_5 \kappa_0 P_{3x})^2}}{c_5}}{2} P_{3x} \quad (43)$$

**Fig. 8** Path replanning: if the RRT path planner starts the replanning at  $R_1$ , it may cause a heading discontinuity at this point even though it satisfies the position and curvature continuity. However, if the heading at  $R_1$  is stored and another linear path is created which holds the same heading as  $R_1$ , the continuous curvature path can be generated



**Table 3** Flight time comparison

	Known E	Partially known E	Unknown E
Flight time (seconds)	36.8	38.0	38.2
Flight distance (m)	184.2	189.9	191.2

The goal is to decide on a value for  $P_{3x}$  which makes  $L_{bc} = D_{rp}$ . It is difficult to find the exact solution to Eq. 43. To solve this problem we assume that  $P_{3x}$  located in inside the square-root has a constant value  $D_p$ . Using this assumption Eq. 43 becomes

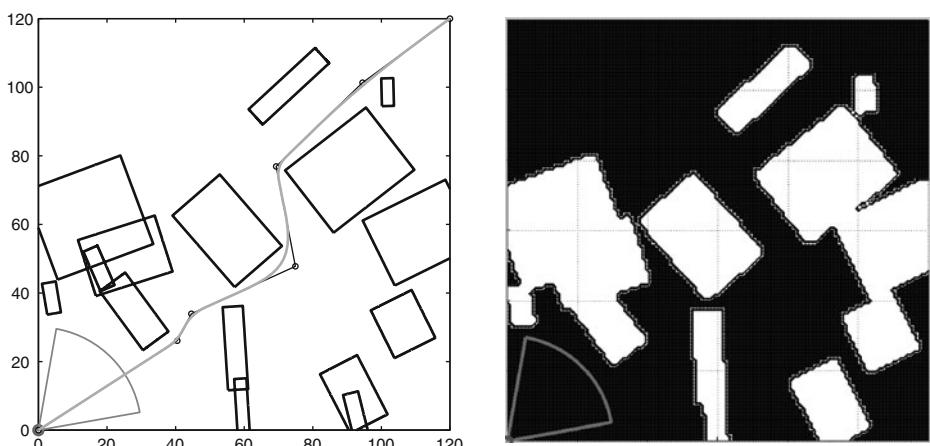
$$P_{3x} = f(V, T_{rp}, \kappa_0) \approx \frac{2D_{rp}}{\sqrt{1 + (c_4\kappa_0 D_p)^2} + \sqrt{\frac{2c_4}{3(c_1+1)}} + \frac{c_4\sqrt{1+(c_5\kappa_0 D_p)^2}}{c_5}} \quad (44)$$

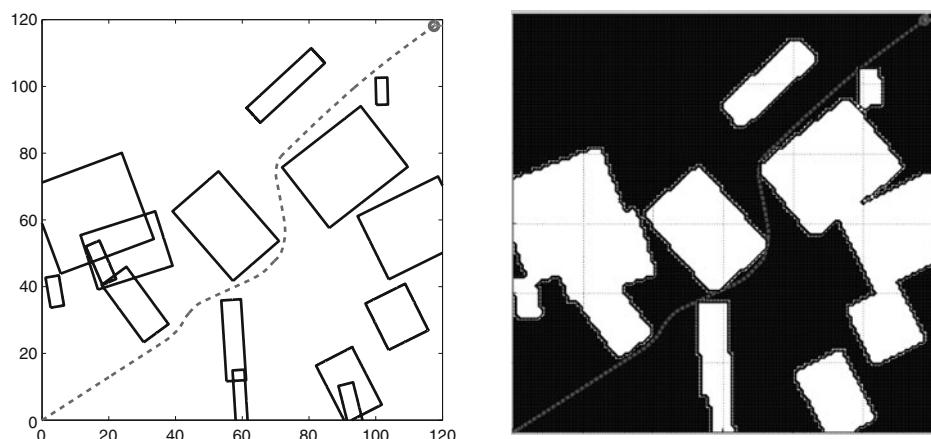
Since Eq. 43 is highly nonlinear the assumption in Eq. 44 is not valid. In our application the helicopter flies slower than 10m/s and the allowable maximum curvature of the helicopter is 0.25. The maximum run time for path planning is 1 second. Within these constraints the  $D_p$  function can be approximated as follows:

$$D_p = D_{rp} [e^{-\kappa_u} + (0.9 - 0.12 \cdot D_{rp}) \cdot \kappa_u] \quad (45)$$

where  $D_{rp} = V \cdot T_{rp}$  is the distance which the helicopter flies during the run time of path generation and  $\kappa_u$  is an unsigned curvature. Therefore  $D_p$  changes with respect to the helicopter velocity, the replanning run time, and the curvature at the point which the replanning is triggered. If  $P_{3x}$  is decided using Eq. 44 which considers the helicopter velocity, the replanning run time, and the curvature of the path, then the decreasing curvature path is generated from Eqs. 35 and 36.

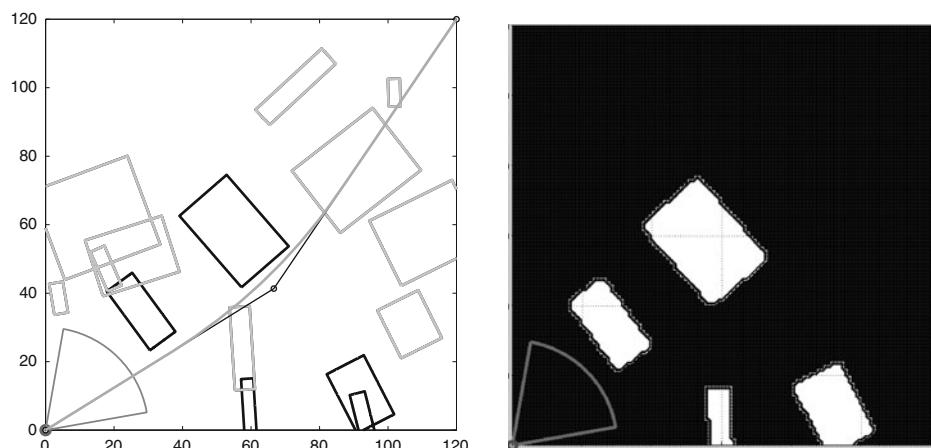
Figure 8 shows the proposed dynamic path replanning algorithm. When the helicopter flies at  $R_0$ , a new obstacle is detected located in the reference path(dashed line). The helicopter starts the replanning at  $R_0$  point. First the decreasing curvature

**Fig. 9** On-line dynamic planning and control in known environment: start

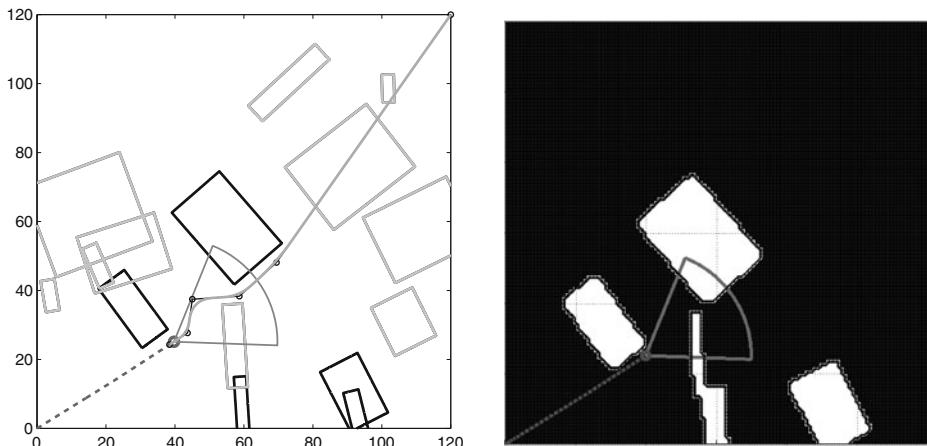


**Fig. 10** On-line dynamic planning and control in known environment: finish (36.84 s)

path is generated which ends at  $R_1$ . If the RRT path planner starts the path replanning at  $R_1$ , it may cause a heading discontinuity at this point even though it satisfies the position and curvature continuity (They share the same position value at  $R_1$  and the curvature of both path are zero at this point but the headings are different). To solve this problem, the heading at  $R_1$  is stored and another linear path is created which holds the same heading as  $R_1$ .  $R_3$  represents this linear line. The role of  $R_3$  is to offer an length for path smoothing which preserves the heading continuity. Therefore even though the RRT path planner starts the replanning at  $R_3$ , the new smooth path will start at  $R_1$ .



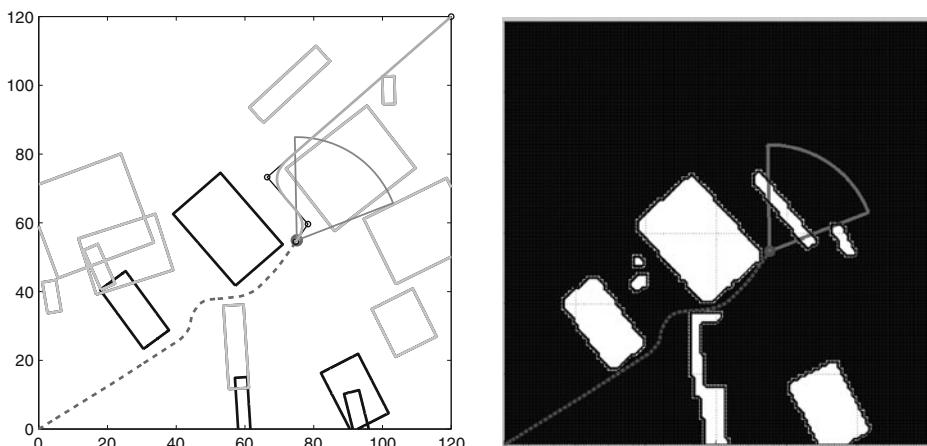
**Fig. 11** On-line dynamic planning and control in partially known environment: start



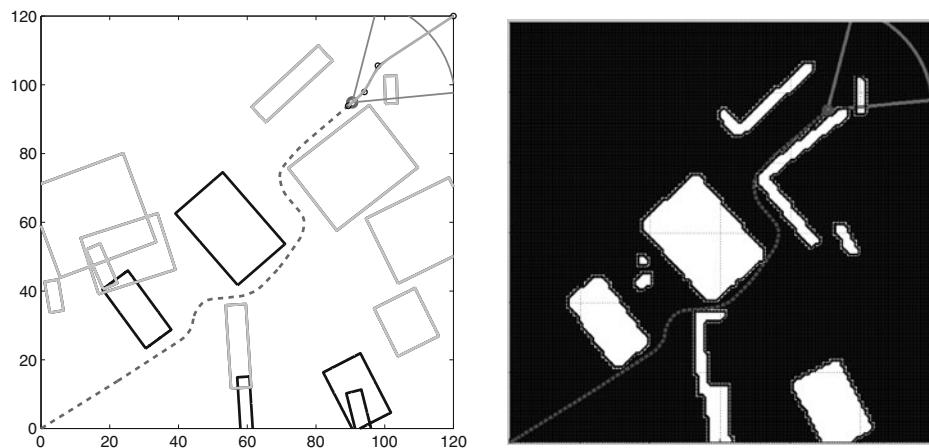
**Fig. 12** On-line dynamic planning and control in partially known environment: 11 s

## 5 Simulation Experiments

In this section we will demonstrate the effectiveness of our on-line path planning algorithm for three different scenarios: a fully known environment; partially known environment; and a totally unknown environment. The helicopter flies with 5m/s velocity and 15 obstacles are generated randomly. An obstacle map is built by using a laser scanner as in [24, 25]. The laser scanner has a 70° field of view and returns the range information within 30m. The obstacle size in the map is larger than the real size since a safety margin is included when generating the map. Table 3 shows the flight time and distance comparison among three different environments. It takes the shortest flight time and distance in the fully known environment case and it takes the longest time and distance in the totally unknown environment case.



**Fig. 13** On-line dynamic planning and control in partially known environment: 21 s

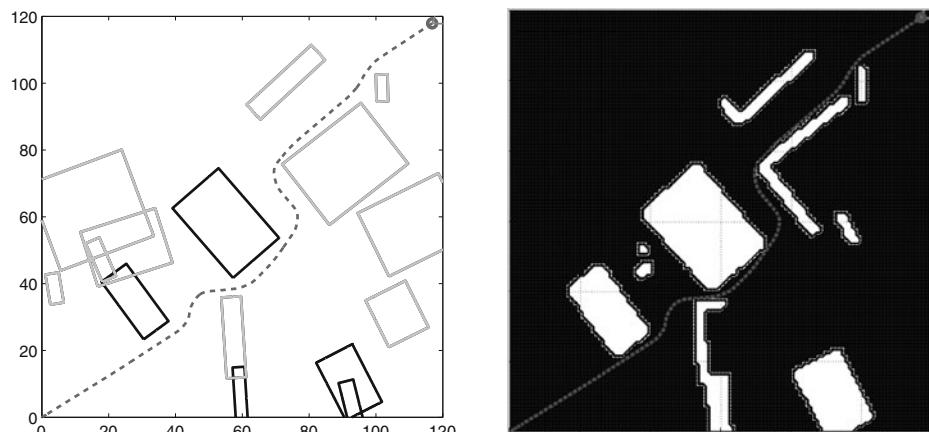


**Fig. 14** On-line dynamic planning and control in partially known environment: 31 s

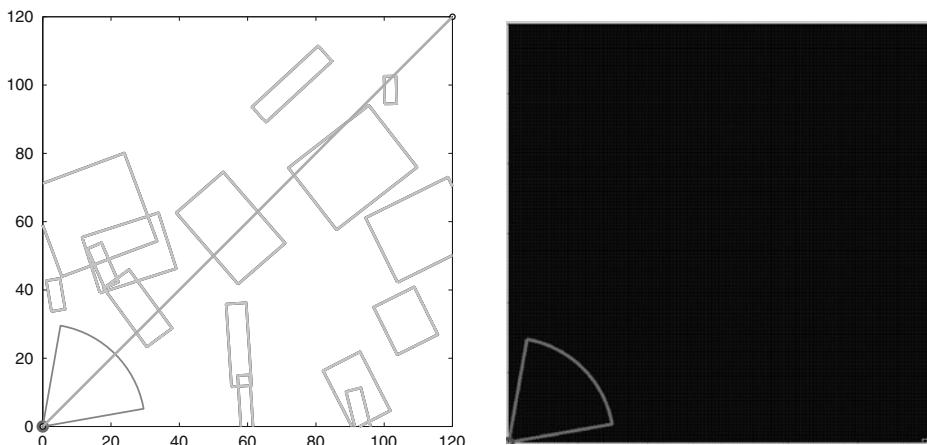
### 5.1 Fully Known Environment

In this scenario all obstacle locations are known before flight so the helicopter holds perfect information about the environment. Figure 9 (left) shows the path planning result: the black line represents a polygonal path and the green line represents a continuous curvature path. Figure 9 (right) shows an obstacle map of the environment. The obstacle size in the map is larger than the real obstacle as already mentioned based on the safety margin applied.

Figure 10 shows the flight path (dotted line) and the map. The helicopter navigates successfully in the cluttered environment without collision. The obstacle map (right) is not changed since the helicopter already has a perfect map of the environment before flight.



**Fig. 15** On-line dynamic planning and control in partially known environment: finish (37.98 s)



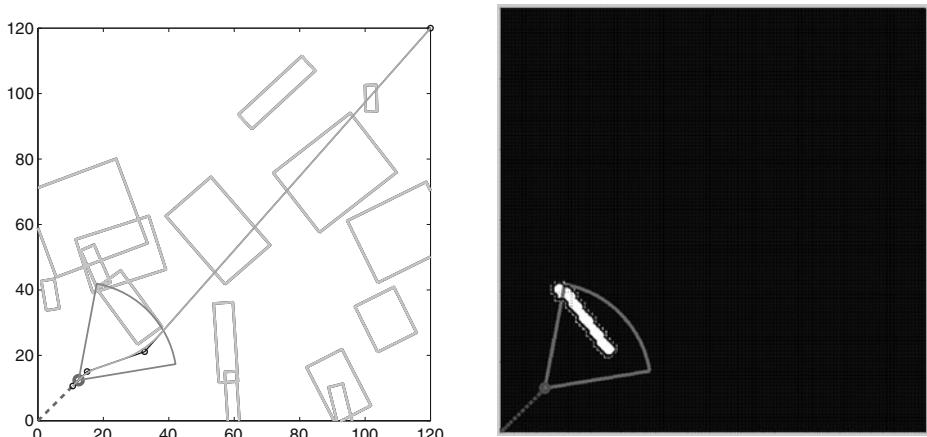
**Fig. 16** On-line dynamic planning and control in unknown environment: start

## 5.2 Partially Known Environment

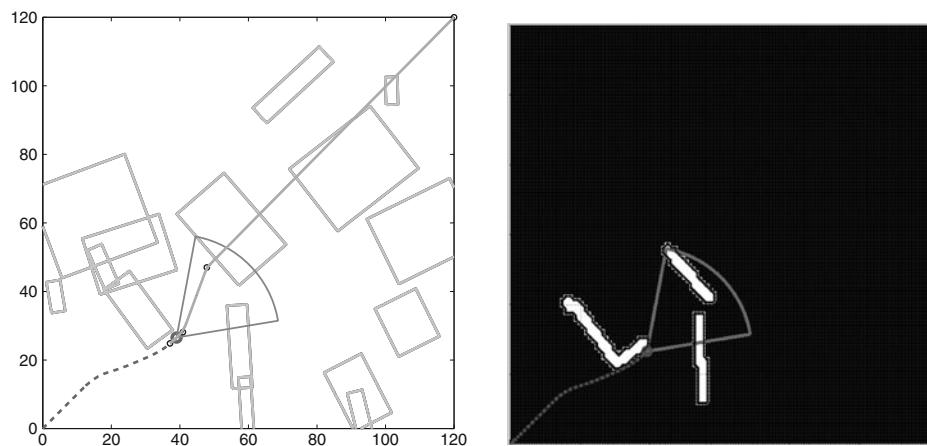
In this case the helicopter only has information about five obstacles (black color) and there are unknown ten obstacles (crayon color). Figure 11 shows the initial path (left) and the map (right). Since the helicopter generates a path based on partial information then collisions will occur.

Figure 12 shows the path and the obstacle map when the flight time is 11 seconds. A new obstacle located at (60,30) is detected and using the updated map the RRT path planner replans the path. It generates a path which passes through the narrow passage since the shortest length metric is used to generate a path.

Figure 13 shows the path and the obstacle map when the flight time is 21 seconds. Even though the obstacle located at (90,75) is large, the size of this obstacle in the



**Fig. 17** On-line dynamic planning and control in unknown environment: 5 s



**Fig. 18** On-line dynamic planning and control in unknown environment: 11 s

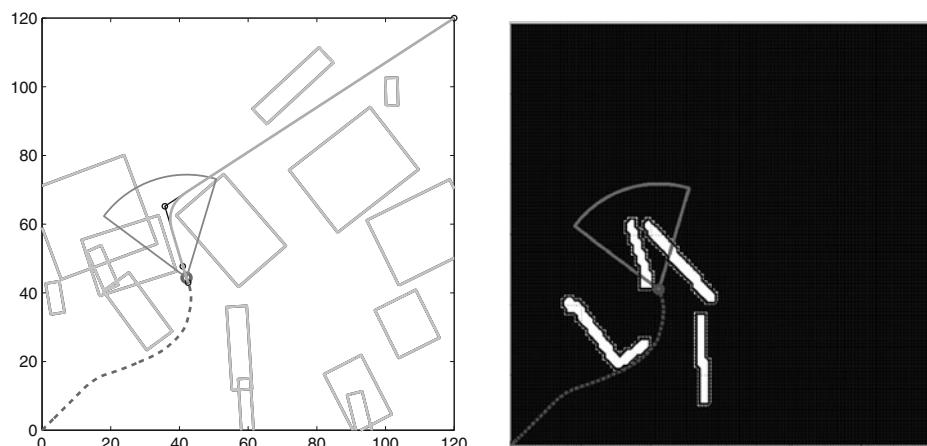
map is small due to the occlusion of the sensor. However the planner generates a collision free path successfully based on this map.

Figure 14 shows the path and the obstacle map at 31 seconds. The sensor detects the last obstacle at (105,95) and replans the path again to avoid the obstacle.

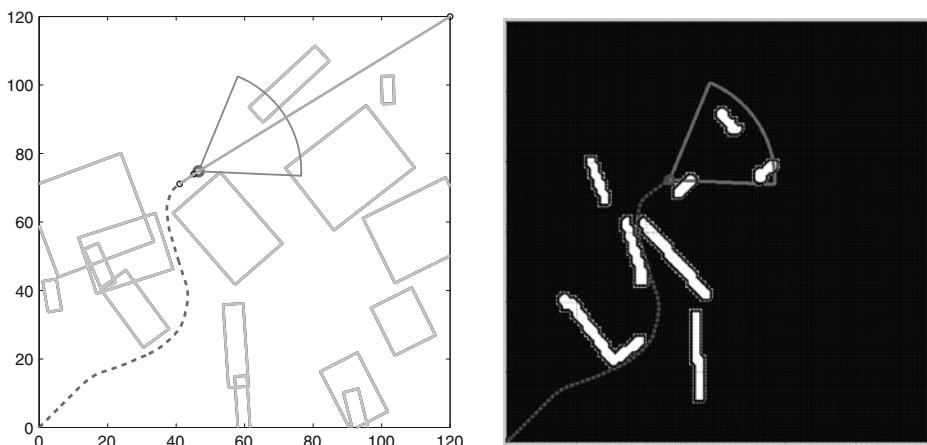
Figure 15 shows the path and the obstacle map when the helicopter arrives at the target point. The final path has a similar shape to that of the fully known environment except for a slight change near (50,35).

### 5.3 Totally Unknown Environment

In this case, the helicopter has no information about the obstacles when it starts its mission. Figure 16 shows the initial path and the obstacle map. Since the helicopter



**Fig. 19** On-line dynamic planning and control in unknown environment: 15 s

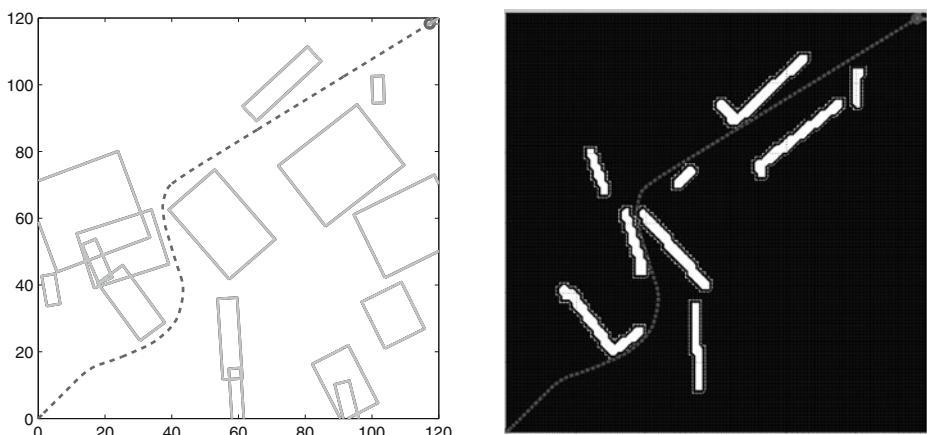


**Fig. 20** On-line dynamic planning and control in unknown environment: 22 s

assumes that there are no obstacles in the environment it initially generates a straight line path.

Figure 17 shows the path and the obstacle map when the flight time is 5 seconds. The previous two cases have the information about the obstacle located in (30,30) so the initial heading is about 30° but the initial heading of the unknown environment case is 45° which causes replanning at 5 seconds.

Figure 18 shows the path and the obstacle map when the flight time is 11 seconds. The previous two cases have the information about the obstacle located in (60,60) so they plan the path which pass the narrow passage since it is shortest path based on their map. However in this case the helicopter senses the obstacle located in (60,60) gradually when it approaches this obstacle. In this case the shortest path is to travel to the left side of the obstacle when the replanning occurs.



**Fig. 21** On-line dynamic planning and control in unknown environment: finish (38.24 s)

Figure 19 shows the path and the obstacle map when the flight time is 15 seconds. The flight route of the helicopter is different from the previous two cases. The planner selects the left side of the obstacle located in (60,60).

Figure 20 shows the path and the obstacle map when the flight time is 22 seconds. Even though the helicopter detects 3 obstacles in front, it does not change the path since they do not cause a collision.

Figure 21 shows the path and the obstacle map when the helicopter arrives at the target point. It shows that the RRT planner successfully generates a collision free path even though it has no a-priori information about the environment.

## 6 Conclusions

An on-line dynamic path planning algorithm for a UAV navigating in an unknown cluttered environment is presented. The RRT path planner generates a collision-free path and cubic Bézier spiral curves are used to smoothen the path guaranteeing the curvature continuity. This path can then be tracked accurately by the novel MPC path following controller presented in this paper. We also consider the run time of the path planning process explicitly when replanning occurs. The proposed method generates a curvature continuous path whenever the path replanning is triggered. The simulation results show that the proposed method can achieve autonomous navigation successfully regardless of the level of a-priori information about environment.

**Acknowledgements** This work is supported in part by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

## References

1. Stentz, A., Hebert, M.: A complete navigation system for goal acquisition in unknown environments. *Auton. Robots* **2**(2), 127–145 (1995)
2. Brock, O., Khatib, O.: High-speed navigation using the global dynamic window approach. In: IEEE International Conference on Robotics and Automation, Detroit, Michigan (1999)
3. Ersson, T., Hu, X.: Path planning and navigation of mobile robots in unknown environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii (2001)
4. Bruce, J., Veloso, M.: Real-time randomized path planning for Robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland (2002)
5. Philipsen, R., Kolski, S., Macek, K., Siegwart, R.: Path planning, replanning, and execution for autonomous driving in urban and offroad environments. In: Proceedings of the ICRA Workshop on Planning, Perception and Navigation for Intelligent Vehicles (2007)
6. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Maximizing miniature aerial vehicles. *IEEE Robot. Autom. Mag.* **13**, 34–43 (2006)
7. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT, Cambridge (2005)
8. Frazzoli, E., Dahleh, M., Feron, E.: Real-Time Motion Planning for Agile Autonomous Vehicles. American Control Conference, Arlington, Virginia (2001)
9. Espinoza, J., Sánchez, A., Osorio, M.: Exploring unknown environments with randomized strategies. In: MICAI 2006: Advances in Artificial Intelligence (2006)
10. Walton, D.J., Meek, D.S., Ali, J.M.: Planar G2 transition curves composed of cubic Bezier spiral segments. *J. Comput. Appl. Math.* **157**(2), 453–476 (2003)
11. Wzorek, M., Doherty, P.: Reconfigurable path planning for an autonomous unmanned aerial vehicle. In: IEEE International Conference on Hybrid Information Technology (2006)
12. Sanders, C.P., DeBitetto, P.A., Feron, E., Vuong, H.F., Leveson, N.: Hierarchical control of small autonomous helicopters. In: IEEE Conference on Decision and Control, Tempa, Florida (1998)

13. Muratet, L., Doncieux, S., Briere, Y., Meyer, J.A.: A contribution to vision-based autonomous helicopter flight in urban environments. *Robot. Auton. Syst.* **50**, 195–209 (2005)
14. Silvestre, C., Pascoal, A., Kaminer, I.: On the design of gain-scheduled trajectory tracking controllers. *Int. J. Robust Nonlinear Control* **12**(9), 797–839 (2002)
15. Kim, H., Shim, D., Sastry, S.: Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In: American Control Conference, Anchorage, AK (2002)
16. Granchiarova, A., Johansen, T.A., Kocjan, J.: Explicit model predictive control of gas-liquid separation plant via orthogonal search tree partitioning. *Comput. Chem. Eng.* **28**, 2481–2491 (2004)
17. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. TR 98-11, Computer Science Dept, Iowa State University (1998)
18. Bartels, R.H., Beatty, J.C., Barsky, B.A.: An Introduction to Splines for use in Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers, Los Altos, California (1986)
19. Yang, K., Sukkarieh, S.: 3D smooth path planning for a UAV in cluttered natural environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, 22–26 September 2008
20. Mettler, B., Tischler, M.B., Kanade, T.: System identification of small-size unmanned helicopter dynamics. In: American Helicopter Society 55th Forum, Montreal, Quebec (1999)
21. Shim, D.H., Kim, H.J., Sastry, S.: Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles. In: AIAA Guidance, Navigation and Control Conference, Denver (2000)
22. Hwang, J., Arkin, R., Kwon, D.: Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In: IEE/RSJ Int. Conference on Intelligent Robots and Systems, Las Vegas, Nevada (2003)
23. Gravesen, J.: Adaptive subdivision and the length and energy of Bezier curves. *Comput. Geom.* **8**(1), 13–31 (1997)
24. Shim, D., Chung, H., Sastry, S.: Conflict-free navigation in unknown urban environments. *IEEE Robot. Autom. Mag.* **13**, 27–33 (2006)
25. Scherer, S., Singh, S., Chamberlain, L., Elgersma, M.: Flying fast and low among obstacles: methodology and experiments. *Int. J. Rob. Res.* **27**(5), 549–574 (2008)

# On the Generation of Trajectories for Multiple UAVs in Environments with Obstacles

Armando Alves Neto · Douglas G. Macharet ·  
Mario F. M. Campos

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 28 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper presents a methodology based on a variation of the Rapidly-exploring Random Trees (RRTs) that generates feasible trajectories for a team of autonomous aerial vehicles with holonomic constraints in environments with obstacles. Our approach uses Pythagorean Hodograph (PH) curves to connect vertices of the tree, which makes it possible to generate paths for which the main kinematic constraints of the vehicle are not violated. These paths are converted into trajectories based on feasible speed profiles of the robot. The smoothness of the acceleration profile of the vehicle is indirectly guaranteed between two vertices of the RRT tree. The proposed algorithm provides fast convergence to the final trajectory. We still utilize the properties of the RRT to avoid collisions with static, environment bound obstacles and dynamic obstacles, such as other vehicles in the multi-vehicle planning scenario. We show results for a set of small unmanned aerial vehicles in environments with different configurations.

**Keywords** Multiple UAV trajectory planning · Rapidly-exploring Random Trees · Pythagorean Hodograph curves · UAV swarm

---

This work was developed with the support of CNPq, CAPES and FAPEMIG.

A. Alves Neto (✉) · D. G. Macharet · M. F. M. Campos  
Computer Vision and Robotic Laboratory (VeRlab), Computer Science Department,  
Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil  
e-mail: aaneto@dcc.ufmg.br

D. G. Macharet  
e-mail: doug@dcc.ufmg.br

M. F. M. Campos  
e-mail: mario@dcc.ufmg.br

## 1 Introduction

Several open issues still demand much research effort in order to endow a vehicle with full autonomy in a real environment. Among those, one well known problem is posed by one of mobile robotics basic questions related to navigation: “How do I get to a given goal?” Throughout the years, several efficient and relevant solutions have been proposed with varying degrees of success. Normally, given some knowledge of the environment, such as position of the obstacles and of others vehicles, it is possible to plan a path that allows the vehicle to safely achieve this goal.

A fundamental feature of a path planning algorithm is to ensure that the vehicle will be able to execute this path, which means that during the trajectory generation, the movements restrictions of the vehicle must be considered (i.e. nonholonomic constraints). For example, a typical car cannot move laterally, so the radius of curvature is one of the restrictions imposed on trajectories generated for cars.

It is important to remark, however, that trajectory planning and path planning are two different tasks. Path planning deals with the position of a continuous set of points that represents a collision free path through the environment. A trajectory is parameterized by time, somehow embedding the body dynamic constraints. Therefore, different vehicles can be at the same position, but in a different time.

Several strategies to obtain a path between two different positions can be found in the literature [1]. Among the simplest and most common are the visibility graph, cell decomposition and potential field path planners. However, one of the problems with those techniques is that they guarantee obstacle free path between two points with no consideration as to the vehicle’s capacity to execute follow the generated path (e.g. negotiate a curve between obstacles).

In the robotics literature, most of path planners were designed for manipulators and ground vehicles moving in a two dimensional environment. Nowadays, the kinds of mobile robots abound, each of them possessing different locomotion particularities as well as different types of motion constraints. The choice of the type of vehicle to be used is determined by the mission it carry out and the environment thereof.

The interest and research in Unmanned Aerial Vehicles (UAVs) has been increasing growing, specially due to the decrease in cost, weight, size and performance of sensors and processors. Clearly UAVs have their niche of applications, which is cannot be occupied by other types of mobile robots, as such as they are capable of covering a broad set of relevant applications. They are able to navigate over large areas obviously faster than land vehicles, with a privileged view from above, which is one of their main uses in monitoring and surveillance. As the availability increases, so does the possibility of having multiple such vehicles traversing a given volume of the space. Therefore, there is growing need to study and develop techniques for the generation of safe and feasible trajectories considering the specific constraints of different types of aerial robots.

UAVs can be divided into at least three classes: rotary-wing aircrafts (e.g. helicopters and quadrotors), aerostatic aircrafts (such as airships and hot air balloons) and fixed-wing aircrafts (airplanes). The technique described in this text will be instantiated for fixed-wing UAV, however, without loss of generality, it can be applied to other types of vehicles. Fixed-wing aircrafts present constraints on their mobility such as minimum curvature, maximum angle of climb or dive, and minimum speed.

The possibility of using multiples UAVs brings additional advantages, including the reduction of time needed to complete a task, as well as improving the robustness of the overall system. Thus, the generation of a trajectory must also take into account all the vehicles in that air space. Having this as one of our research objectives, in this text we present an improved approach to the trajectory generation problem for multiples UAVs flying in an environment with obstacles. Our approach simplifies the three dimensional case to UAVs modeled as vehicles moving in two dimensions with non zero speed and limited turning rate.

An efficient adaptation of the well-known probabilistic technique used for the motion planning problem, called Rapidly-exploring Random Tree (RRT) is described, where the possible link between new states is calculated based on the use of a special type of Bézier curve, known as the Pythagorean Hodograph curve (PH). We also show the advantages of using this approach in the trajectory planning of multiple robots in different environments.

This paper is organized as follows. Next section presents previous works found in the literature which deal with the problem of motion planning. Section 3 is on our methodological approach. We first formalize the problem, and then we detail the classic RRT algorithm followed by a description of PH curves. Finally, our adaptation using these techniques is carefully developed and discussed. The description and results of several experiments are presented in Section 4. Finally, Section 5 presents the conclusion and possible directions for future work.

## 2 Related Works

Motion-planning problem for autonomous vehicles is the subject of many investigations, and various works on this overall topic can be found in literature [1–3]. One possible taxonomy of the area can be based on the number of vehicles involved (one or several), and the presence or absence of obstacles in the environment. Even though the generation of feasible paths (or trajectories) for nonholonomic vehicles is in itself a great challenge, ignoring the possible presence of obstacles restricts a broader use of such techniques. Some approaches of single vehicle path planning in general environments can be found in literature [4, 5]. Voronoi diagram is a widely used technique to generate paths with such constraints [6, 7]. Rapidly-exploring Random Trees (RRTs) can also be used in this case, especially for solving the case of nonholonomic vehicles. Cheng et al. [8] presents trajectory planning for both an automobile and a spacecraft. In the later example, even though an obstacle free environment is concerned, the focus remains on the motion constraints that need to be satisfied for a safe entry of the spacecraft in the Earth's atmosphere. Other works like [9, 10], use this technique to generate nominal paths for miniature air vehicles. The authors present an algorithm for terrain avoidance for the air platform BYU/MAV, which allows, among other things, flying through canyons and urban environments.

As we have already mentioned, the use of multiple vehicles can provide a better and more efficient execution of a given task, but at a possible higher cost and increase in the complexity of the controller responsible for the generation of such motions. Therefore, some works only deal with the generation of safe trajectories for all vehicles assuming an obstacle free environment. It is important to note that vehicles

on a team may be regarded as obstacles, making cooperation almost an inevitable approach for the problem [11, 12].

The more general case of motion planners that can handle multiple vehicles in the presence of obstacles can also be found in the literature. A common technique used to deal with a large number of robots is to team them up, simplifying the planner's task by considering all the robots in a group as a single robot [13, 14]. The main problem with this approach is when different robots in the team need to execute different tasks. One of the most popular techniques for dealing with those cases is based on potential fields [15]. The planning stage, in this case consists basically in computing the attraction force to the goal letting the obstacle avoidance be completely reactive. A major problem with this approach is the possible occurrence of local minima, and some works have described ways to overcome this problem [16]. Finally, two other methods are the use an adaptation of the classic roadmap path planning, making it act dynamically, considering each robot as a moving obstacle [17], and those based on geometric solutions [18].

### 3 Methodology

#### 3.1 Problem Statement

Our technique assumes the existence of an environment with obstacles whose position and geometry are known. Also, others limitations for the robot navigation are imposed by its own kinematic constraints. Two configurations,  $\mathbf{P}_{init}$  and  $\mathbf{P}_{goal}$  respectively, describe the initial and final states and which define the position and the orientation of the robot in the extreme points of the path. These states can represent any pair of waypoints in a set, which in turn, is defined by the mission planning module at a higher level in the robot's architecture.

A trajectory may be defined, mathematically, as a parametric curve  $\vec{r}(t)$  in the two-dimensional space, where  $t$  is the time parameter that continuously varies in  $\mathbb{R}$ . In this manner, the trajectory planning problem for a single robot can formally be described as:

$$\begin{aligned}\mathbf{P}_{init}(x_{init}, y_{init}, \psi_{init}) &= \vec{r}(t_{init}), \\ \mathbf{P}_{goal}(x_{goal}, y_{goal}, \psi_{goal}) &= \vec{r}(t_{goal}),\end{aligned}\quad (1)$$

where  $t_{init}$  and  $t_{goal}$  are the initial and final time values, respectively, for the curve parameter  $t$ .

Each waypoint is described by two position ( $x, y$ ) and one orientation ( $\psi$ ) variable. The variable  $\psi$  (also called yaw) is an angle that describes the waypoint orientation parallel to the  $XY$  plane in relation to the  $\mathbf{X}$  axis of the space.

We consider both local and global constraints. The local constraints are related to the kinematic and dynamic behavior of the robot in the configuration space. The global constraints represent the composition of the obstacles in the environment. The RRT is a technique which generates paths that respect both constraints.

In order to be considered a feasible trajectory for the robot (in an environment with or without obstacles), the curve  $\vec{r}(t)$  must simultaneously fulfill kinematic and dynamic constraints and their maximum numerical values. The main kinematic constraint considered in this work is the maximum curvature  $\kappa_{max}$ , that corresponds

to the minimum curvature radius of the vehicle in space. It is possible to completely define a curve in  $\mathbb{R}^2$  only by means of its curvature function [19].

As far as the physics is concerned, the curvature may be defined as a quantity that is directly proportional to the lateral acceleration of the robot in the space. The value of  $\kappa_{max}$  is inversely proportional to the minimum curvature radius ( $\rho_{min}$ ) of the curve that the vehicle is able to execute, which is also proportional to the vehicle's maximum lateral acceleration. The curvature function of a curve in the  $n$ -dimensional space is given by the following equation:

$$\kappa(t) = \frac{|\dot{\vec{r}}(t) \times \ddot{\vec{r}}(t)|}{|\dot{\vec{r}}(t)|^3}. \quad (2)$$

We still consider the minimum velocity constraint, which represents the minimum translation velocity for the robot on a given trajectory. In case of fixed-wing airplanes, for example, this is an important requirement once those vehicles are not capable to hover. This will also be considered when we discuss the motion planning problem for multiple robots.

With regard to the dynamic, it is important to consider the form by which such constraints vary in the time. The continuity of the curvature and velocity functions is another fundamental characteristic in the trajectory planning for real vehicles. Discontinuities in the curvature function, for example, can induce infinite variations of lateral acceleration, which of course, are not realizable. The same reasoning is valid for the velocity profile. Finally, the curve produced by the trajectory planning algorithm is continuously derivable, should be second order differentiable, according to the Eq. 2.

As far as global constraints are concerned, there are two types of environments. The most common type is the static environment, which is specified only by the geometry, position and orientation of the obstacles in it. In the dynamic environment, the obstacles (which may be other agents, such as robots) move through space, and their trajectories may or may not be known. As we will show later this text, we will use the RRT algorithm to generate trajectories that avoid both static and dynamic obstacles.

Finally, a trajectory  $\vec{r}(t)$  is valid for a vehicle if, and only if, the magnitude of the curvature function is smaller than the vehicle's absolute maximum curvature value, and if the curve is entirely contained in a region of the environment free of obstacles, as described below:

$$\vec{r}(t) \rightarrow (|\kappa(t)| < \kappa_{max}) \wedge (\vec{r}(t) \in \mathbf{P}_{free}), \quad (3)$$

where  $\mathbf{P}_{free}$  represent the set of all states that satisfy the global constraints.

To calculate this trajectory, we first established paths between states produced by the RRT algorithm. These paths are produced by means of the Pythagorean Hodograph curve technique that simultaneously fulfills the kinematic and dynamic constraints of the robot. The composition of these paths will be considered a trajectory when the velocity profile of the robot in each of the PH curves is defined. This will be accomplished by the calculation of the parametric speed of each curve that composes the RRT tree. With this, we will be able to establish positions and orientations for the robot in time, which is essential to avoid collision.

### 3.2 Rapidly-Exploring Random Tree

There are some key factors of a path generating method that need to be considered, such as its efficiency, its ability to deal with obstacles in the environment, and the feasibility of the produced trajectory given a robot's kinematic and dynamic restrictions. A known sampling based planning algorithms used to solve this problem is a technique called Rapidly-exploring Random Tree (RRT) [20], which has been shown to be a good alternative, mainly due to the fact of its ability to quickly explore large areas, and to consider both environmental and the vehicle's nonholonomic constraints during its generation. Figure 1 shows the evolution of the RRT algorithm in an environment, which is faster than other techniques.

Some interesting features related to the RRTs are the fact that it is (probabilistically) complete under general conditions, since it always remains connected regardless of the amount of edges and that it is a relatively simple algorithm when compared to other techniques.

Algorithm 1 presents the basic steps for the generation of an RRT.

---

**Algorithm 1** GENERATE\_RRT( $\mathbf{P}_{init}, K$ )

---

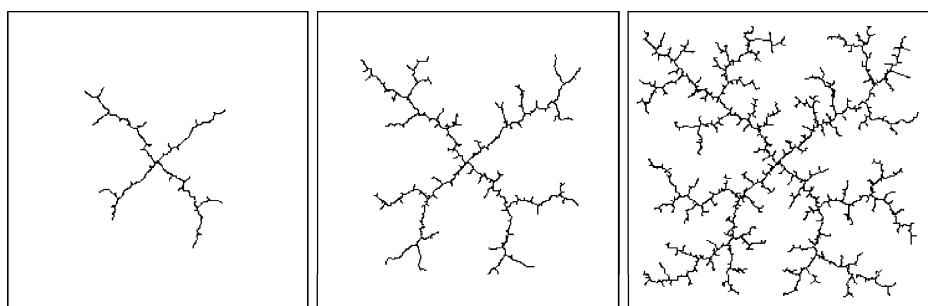
```

1:  $\mathcal{T}.\text{init}(\mathbf{P}_{init})$ 
2: for  $k = 1$  to  $K$  do
3:    $\mathbf{P}_{rand} \leftarrow \text{RANDOM\_STATE}()$ 
4:    $\mathbf{P}_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(\mathbf{P}_{rand}, \mathcal{T})$ 
5:    $u_{new} \leftarrow \text{SELECT\_INPUT}(\mathbf{P}_{rand}, \mathbf{P}_{near})$ 
6:    $\mathbf{P}_{new} \leftarrow \text{NEW\_STATE}(\mathbf{P}_{near}, u_{new})$ 
7:    $\mathcal{T}.\text{add\_vertex}(\mathbf{P}_{new})$ 
8:    $\mathcal{T}.\text{add\_edge}(\mathbf{P}_{near}, \mathbf{P}_{new}, u_{new})$ 
9: end for
10: return  $\mathcal{T}$ 

```

---

First of all, the starting point to the execution of the algorithm needs to be chosen. The starting point will represent a robot state ( $\mathbf{P}_{init} \in \mathbf{P}_{free}$ ), and the maximum number of iterations ( $K$ ) that the algorithm must perform before it stops must also be set. Then, a random position on the map ( $\mathbf{P}_{rand}$ ) is generated, and this position will be used to guide the growth of the tree. A search is executed through all nodes already in the tree to find the one that is the closest to the new random position



**Fig. 1** Rapidly-exploring random tree algorithm evolution [20]

( $\mathbf{P}_{near}$ ) according to some defined metric  $\rho$  (the Euclidean distance is a commonly used metric). The tree will expand from this node. A segment connecting  $\mathbf{P}_{near}$  to  $\mathbf{P}_{rand}$  must be inserted, however only a certain and fixed part of this segment will be actually used to expand the tree. A verification is made to check if it is possible to expand the tree, which means to verify if the segment doesn't intersect with any obstacles. If possible, a new state is generated and added to the tree ( $\mathbf{P}_{new}$ ). In this case, instead of Euclidean distance, our method uses a metric  $\rho$  for determining the  $\mathbf{P}_{near}$  that best suits the motion constraints of the robot. Then, we use the key ideas of a Pythagorean Hodograph curve to determine the value of  $\mathbf{P}_{new}$ , and describe a path between this and the nearest state.

### 3.3 Pythagorean Hodograph Curves

Pythagorean Hodograph (PH) curves are a special kind of parametric curves that present many computational advantages over polynomial parametric curves in general. They were introduced in [21], where, for the first time PH curves of fifth order for the two-dimensional case were presented. A Hermite Interpolation algorithm was proposed in [22], where the author demonstrate that there exist four possible solutions for the curve in  $\mathbb{R}^2$ . The chosen solution is the one that minimizes the cost function [23] (bending energy function) based on the integral of the magnitude of the curvature function.

A PH curve is represented, in general, as  $\vec{p}(\tau) = [x(\tau), y(\tau)]$  which are the derivatives in relation to its parameter (hodograph components). They exhibit a special algebraic property, which is that they satisfy the Pythagorean condition

$$\dot{x}(\tau)^2 + \dot{y}(\tau)^2 = h(\tau)^2, \quad (4)$$

for some polynomial  $h(\tau)$ , where the variable  $\tau$  is the curve parameter. This characteristic provides some properties for the PH curve that can be considered advantageous in the path planning problem: (a) uniform distribution of points along the curve, which contributes to the smoothness of the path; (b) elimination of numerical squaring in computing the path-length, which allows its determination by an exact form; (c) the parametric speed  $s(\tau)$  is a polynomial function of its parameter which means that the velocity profile of the curve is continuous; and finally (d) the curvatures and offset curves are in exact rational form, which enables PH curves to admit real-time interpolator algorithms for computer numerical control.

The length of the path,  $s$ , can be exactly calculated as

$$s = \int_{\tau_i}^{\tau_f} \sqrt{\dot{x}(\tau)^2 + \dot{y}(\tau)^2} d\tau = \int_0^1 |h(\tau)| d\tau, \quad (5)$$

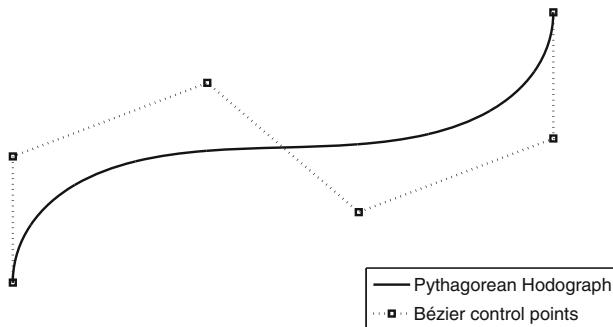
and the PH curves are still shaped as fifth order Bézier curves

$$\vec{p}(\tau) = \sum_{k=0}^5 \mathbf{p}_k \binom{5}{k} (1 - \tau)^{5-k} \tau^k; \quad \tau \in [0, 1], \quad (6)$$

where  $\mathbf{p}_k = [x_k, y_k]$  is the  $k$ -th control point of the Bézier curve. The Fig. 2 presents a example of PH curve of fifth order modeled as a Bézier function.

The path planning problem is then reduced to finding a solution to the Hermite Interpolation problem described in [21]. One important advantage of using this

**Fig. 2** Pythagorean Hodograph curve of fifth order and its Bézier control points for the two-dimensional case



model is that the resulting curve is infinitely continuous, so that the curvature function is always smooth.

Another advantage is related to the offset curve of a PH,  $\vec{p}_o(\tau)$ . This can be used to define a safety navigation region or still a sensor uncertainty for the position and orientation of the vehicle along the path. The offset curve can be computed as

$$\vec{p}_o(\tau) = \vec{p}(\tau) \pm d\mathbf{N}(\tau), \quad (7)$$

where  $\mathbf{N}(\tau)$  represents the unit normal vector of the robot acceleration,

$$\mathbf{N}(\tau) = \frac{\dot{\mathbf{T}}(\tau)}{h(\tau)\kappa(\tau)},$$

that is a function of the variation of the unit tangent vector  $\mathbf{T}(\tau)$  of the path, and  $d$  is the distance of the offset. As the offset curve self-intersects when the path is too convex or too concave,  $d$  must be chosen as a value less than the local radius of curvature to avoid this problem.

In the next section we show how to compute the Pythagorean Hodograph curve between two arbitrary states. This principle will be used when we generate a RRT tree whose vertices are connected by this Bézier curve. The total trajectory for a single robot will be a composition of PH curves, which can be stated as

$$\vec{r}(t) = [\vec{p}_1(t), \vec{p}_2(t), \dots, \vec{p}_V(t)], \quad (8)$$

where  $V$  is the number of vertex in the tree that belongs to the correct path, and  $\vec{p}_i(t)$  is represented as

$$\vec{p}_i(t) = \frac{s_i}{v_{max}} \left[ \frac{\vec{p}_i(\tau)}{|\dot{s}_i(\tau)|} \right],$$

where  $s_i$  is the path-length of the PH given by Eq. 5,  $\dot{s}_i(\tau)$  is the parametric speed and  $v_{max}$  is the maximum speed of the UAV.

### 3.4 Realizable Path Calculation

In this section we discuss how to generate paths that are attainable by a robot in a plane using the principles of PH curves. Those paths comply with the curvature constraints imposed by a given robot. Thus, assuming the model described by Eq. 6,

the path planning problem reduces to determining the six control points of the Bézier curve.

Four of these points can be calculated by the following set of equations (derived from [22]), which depends only the initial and final configuration of the robot:

$$\begin{aligned}\mathbf{p}_0 &= [x_i, y_i], \\ \mathbf{p}_1 &= \mathbf{p}_0 + \frac{k_0}{5} [\cos(\psi_i), \sin(\psi_i)], \\ \mathbf{p}_4 &= \mathbf{p}_5 - \frac{k_5}{5} [\cos(\psi_f), \sin(\psi_f)], \\ \mathbf{p}_5 &= [x_f, y_f],\end{aligned}\tag{9}$$

where  $k_0$  and  $k_5$  are gain factors that have unit value for PH with no constraints.

There are two problems to be resolved at this point. First, we must be able to calculate the remaining points,  $\mathbf{p}_2$  and  $\mathbf{p}_3$  of the Bézier curve. The determination of these points is closely related with the Pythagorean condition (Eq. 4). Second, we must choose the best values of  $k_0$  and  $k_5$  in order to comply with the curvature constraint of the robot.

To find the remaining points, we should resolve the following equation, derived from the Hermite Interpolation system presented in [22]:

$$\begin{aligned}\mathbf{p}_2 &= \mathbf{p}_1 + \frac{1}{5} \begin{bmatrix} u_0 u_1 - v_0 v_1 \\ u_0 v_1 + u_1 v_0 \end{bmatrix}^T, \\ \mathbf{p}_3 &= \mathbf{p}_2 + \frac{1}{15} \begin{bmatrix} 2u_1^2 - 2v_1^2 + u_0 u_2 - v_0 v_2 \\ 4u_1 v_1 + u_0 v_2 + u_2 v_0 \end{bmatrix}^T.\end{aligned}\tag{10}$$

The parameters  $[u_0, u_1, u_2]$  and  $[v_0, v_1, v_2]$  represent coefficients of the polynomial function  $u(t)$  and  $v(t)$  respectively, used in the fifth order PH project. They can be calculated as

$$\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \sqrt{\frac{5}{2}} \begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_0\| + \Delta x_0} \\ \text{sign}(\Delta y_0) \sqrt{\|\Delta\mathbf{p}_0\| - \Delta x_0} \end{bmatrix},\tag{11}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \pm \sqrt{\frac{5}{2}} \begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_4\| + \Delta x_4} \\ \text{sign}(\Delta y_4) \sqrt{\|\Delta\mathbf{p}_4\| - \Delta x_4} \end{bmatrix},\tag{12}$$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = -\frac{3}{4} \begin{bmatrix} u_0 + u_2 \\ v_0 + v_2 \end{bmatrix} \pm \sqrt{\frac{1}{2}} \begin{bmatrix} \sqrt{c+a} \\ \text{sign}(b) \sqrt{c-a} \end{bmatrix},\tag{13}$$

where

$$\begin{aligned}\Delta x_k &= x_{k+1} - x_k, \\ \Delta y_k &= y_{k+1} - y_k,\end{aligned}$$

and

$$\Delta\mathbf{p}_k = [\Delta x_k, \Delta y_k].$$

The  $a$ ,  $b$  and  $c$  parameters can be determinated by following:

$$\begin{aligned} a &= \frac{9}{16}(u_0^2 - v_0^2 + u_2^2 - v_2^2) + \frac{5}{8}(u_0 u_2 - v_0 v_2) + \frac{15}{2}(x_4 - x_1), \\ b &= \frac{9}{8}(u_0 v_0 + u_2 v_2) + \frac{5}{8}(u_0 v_2 + v_0 u_2) + \frac{15}{2}(y_4 - y_1), \\ c &= \sqrt{a^2 + b^2}. \end{aligned}$$

As stated before, there are four solutions for the PH calculation between any initial and final waypoints,  $\mathbf{P}_i$  and  $\mathbf{P}_f$ , as it can be readily seen in the ambiguity of Eqs. 12 and 13. They represent an underdetermined system, for which the best solution of the combinatorial arrangement is the one that minimizes the cost function of the path, or the bending elastic energy function [23],

$$\mathcal{E} = \int_0^1 \kappa(\tau)^2 |\dot{\vec{p}}(\tau)| d\tau. \quad (14)$$

Once the Bézier points are computed, we must guarantee that the PH does not violate the kinematic constraints of the vehicle. For this, we must determine the values of  $k_0$  and  $k_5$  in Eq. 9. As there is no closed form solution, these values are increased iteratively, until that  $\vec{p}(\tau)$  becomes realizable.

At this point we just monitor the maximum curvature and the minimum parametric speed of the PH. The global constraints will be obtained by a variation of the RRT algorithm described in the next section.

### 3.5 Single Trajectory Planning

Initially, the case of planning a trajectory for a single robot in an environment with obstacles will be considered. As stated earlier, the kinematic and dynamic constraints of the vehicle must be satisfied in order that the robot be able to execute the generated trajectory.

The first step in calculating the RRT consists of initializing the tree, which is accomplished by placing a node with the specifications of the initial state of the robot  $\mathbf{P}_{init}$ . This state not only describes the starting point of the tree but it also includes the initial orientation of the vehicle. The time of arrival at this node, and the initial speed of the UAVs, are considered to be null for the first node.

Then, a new random state  $\mathbf{P}_{rand}$  is chosen for the expansion of the tree. There are many ways to achieve this step, as shown in [8]. According to the authors, a good approach is to perform a biased random choice of this state so that the goal  $\mathbf{P}_{goal}$  is used some of the time. That helps in a more rapid conversion to the final result. In this work, the goal was chosen as a new position 50% of the time, with a completely random state in the rest of the time.

After choosing the random state, the next step is to identify which node already inserted in the tree is the closest to the generated one. Such proximity is measured by a metric  $\rho$ , determined for each specific issue dealt with by the RRT. The most common way to measure the proximity between two points in space is through the Euclidean distance  $D$ . However, this calculation does not take into account the orientation at these points in space.

When dealing with the navigation of nonholonomic vehicles (as those considered here), it is important to also consider the orientation, as the real distance between

two states can vary greatly between two positions. For this reason, a nonholonomic distance calculation is used, which is the same applied in the determination of the Dubins' Path [3]. The metric,  $\rho$ , that we use is described below.

It is known that there are six different kinds of paths between two configurations  $\mathbf{P}_i$  and  $\mathbf{P}_f$  as calculated by the Dubins' method. Among those, four represent an approximation for the length of the PH curve, while the other two paths generate configurations that can compromise the convergence of the PH curve. Therefore, the following condition will be used:

$$\rho = \begin{cases} \min(L_i) & i = 1\dots4, \text{ if } d > d_{min}, \\ \infty & \text{elsewhere,} \end{cases} \quad (15)$$

where  $d = D/\rho_{min}$  represents the Euclidean distance between the points, and

$$d_{min} = \sqrt{4 - (|\cos(\alpha)| + 1)^2 + |\sin(\alpha)|}, \quad (16)$$

is a parameter used as a minimum distance between the two states. In this specific case, it is considered that the angle of arrival at the final state is always zero, and

$$\alpha = \psi_i - \tan^{-1} \left( \frac{y_f - y_i}{x_f - x_i} \right). \quad (17)$$

Each of the four possible distances can be calculated as follows [24]:

$$\begin{aligned} L_1 &= \sqrt{d^2 + 2 - 2\cos(\alpha) + 2d\sin(\alpha)} - \alpha, \\ L_2 &= \sqrt{d^2 + 2 - 2\cos(\alpha) - 2d\sin(\alpha)} + \alpha, \\ L_3 &= \sqrt{d^2 - 2 + 2\cos(\alpha) + 2d\sin(\alpha)} - \alpha - 2\mu - \gamma, \\ L_4 &= \sqrt{d^2 - 2 + 2\cos(\alpha) - 2d\sin(\alpha)} + \alpha + 2\nu - \gamma, \end{aligned} \quad (18)$$

where

$$\mu = \tan^{-1} \left( \frac{-2}{d^2 - 2 + 2\cos(\alpha) + 2d\sin(\alpha)} \right),$$

$$\nu = \tan^{-1} \left( \frac{2}{d^2 - 2 + 2\cos(\alpha) - 2d\sin(\alpha)} \right),$$

and

$$\gamma = \tan^{-1} \left( \frac{\cos(\alpha) + 1}{d - \sin(\alpha)} \right).$$

Minimizing the function  $\rho$  gives an approximation to the nearest node  $\mathbf{P}_{near}$  of the chosen random state. The last step is the creation of new state  $\mathbf{P}_{new}$ , which will be used to expand the tree. This will use the principles of the PH curve, as follows:

$$\mathbf{P}_{new} = \text{PH\_CURVE}(\mathbf{P}_{near}, \mathbf{P}_{rand}, \rho_{min}).$$

In that stage, the orientation  $\psi_{rand}$  of the random point is chosen such that it sets the direction of arrival in  $\mathbf{P}_{new}$  to zero,

$$\psi_{rand} = \tan^{-1} \left( \frac{y_{rand} - y_{near}}{x_{rand} - x_{near}} \right).$$

This is an arbitrary choice that aims at reducing the number of iterations of a regular RRT calculation (determination of the  $u_{new}$  entries), by the use of an analytic function. The problem is that the generated PH curve may lead to a critical point where obstacles may hinder the progress of the tree in the direction of the new point. In the event of a collision like this, a random value can be added to  $\psi_{rand}$  in order to avoid the obstacles.

The entry vector is given by the control points of the Bézier curves, as follows:

$$u_{new} = \mathbf{p}_k, \quad k = [0...5].$$

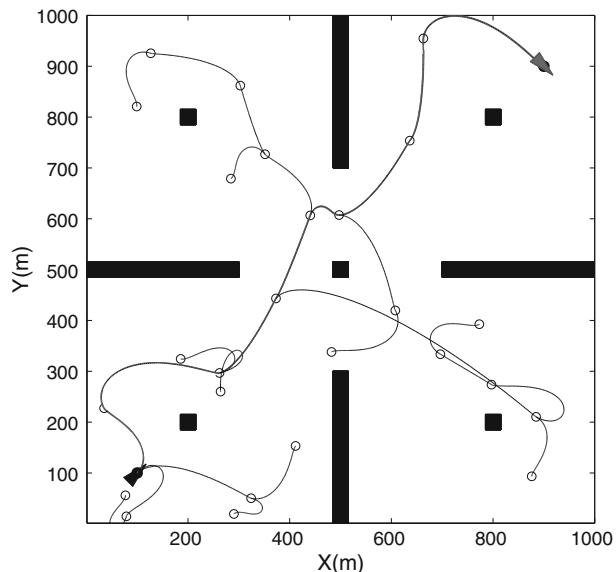
Finally, the trajectory curve can be defined by Eq. 8. Figure 3 presents the evolution of a RRT tree with the Pythagorean Hodograph interpolation between its vertices for an UAV with  $\rho_{min} = 10$  m.

We can see that the use of an analytical function as edges for the tree provides a fast convergence to the final result, since there is no need to limit the reach of  $\mathbf{P}_{new}$ . Compared with the traditional RRT algorithm presented in the Fig. 1, we find a much smaller number of vertices.

### 3.6 Trajectory Replanning

After the calculation of the first curve  $\vec{r}(t)$  for the case of a single UAV, is performed the replanning stage of this trajectory. The main objective is to minimize the total length of the path made by the vehicle, estimated by the random technique of RRT.

**Fig. 3** Example of a RRT with Pythagorean Hodograph interpolation. The *black lines* represent the RRT tree, with the *circles* representing its vertices. *Red lines* correspond to the final trajectory of the UAV



A simple way to achieve this replanning is the elimination of the vertexes of the main tree that culminate in path length prohibitively unnecessary. The Algorithm 2 presents the main steps used in the replanning task from that principle. The method receives as input a tree  $\mathcal{T}$  (with the trajectory description) created in the previous step.

---

**Algorithm 2** REPLANNING\_TRAJECTORY( $\mathcal{T}$ )

---

```

1:  $\mathcal{T}_R$ .init( $\mathcal{T}$ )
2:  $i \leftarrow 1$ 
3: while  $i < V$  do
4:   for  $j = V$  to  $i$  do
5:      $\vec{p}(\tau) \leftarrow \text{PH_CURVE}(\mathbf{P}_i, \mathbf{P}_j, \rho_{min})$ 
6:     if  $\neg \text{COLLISION_DETECTION}(\vec{p}(\tau))$  then
7:        $u_{new} \leftarrow \text{SELECT_INPUT}(\mathbf{P}_i, \mathbf{P}_j)$ 
8:        $\mathcal{T}_R.\text{add\_vertex}(\mathbf{P}_j)$ 
9:        $\mathcal{T}_R.\text{add\_edge}(\mathbf{P}_i, \mathbf{P}_j, u_{new})$ 
10:       $i \leftarrow j$ 
11:    end if
12:   end for
13: end while
14: return  $\mathcal{T}_R$ 

```

---

As stated before,  $V$  represents the total number of vertexes that compose the trajectory described in  $\mathcal{T}$ , while  $\mathcal{T}_R$  corresponds to the new redesigned trajectory.

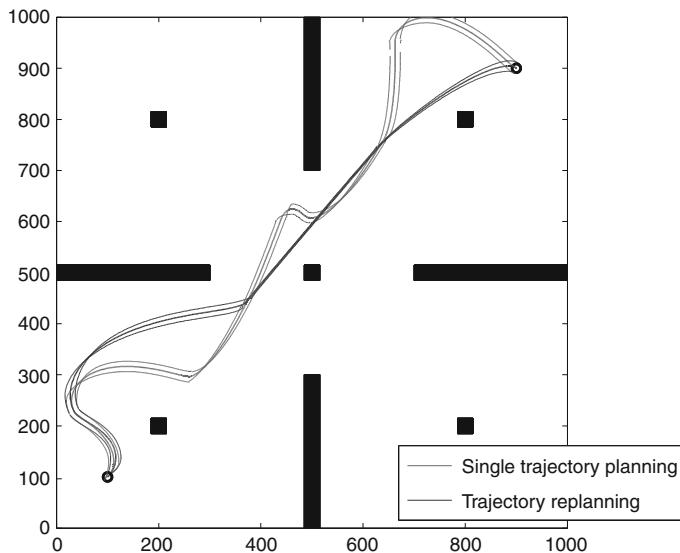
For each vertex  $\mathbf{P}_i$  that describes the trajectory (corresponding initially to  $\mathbf{P}_{init}$  of the tree), the algorithm tries to establish a direct connection with the vertex closest to the final goal ( $\mathbf{P}_{goal}$ ). This connection is again built upon the calculation of a Pythagorean Hodograph curve, still seeking respect the limits of the UAV navigation constraints. For this, there should be no collision of the curve with the obstacles of the environment, as found in the algorithm.

Figure 4 shows the result of trajectory replanning applied to the example of the trajectory on the previous section. As we can see, the new trajectory generated presents a total length less than the first result, showing that the replanning has a great advantage, especially because of its low cost. Another advantage is to reduce the number of vertexes of the tree, making it faster for the planning of multiple UAVs, where each robot is now regarded as a dynamic obstacle for the others.

### 3.7 Multiple Trajectory Planning

After the generation of a trajectory for a single vehicle, the expansion for multiple vehicles can be immediately obtained.

As discussed before, trajectories are parameterized by time, which assumes that after a trajectory is calculated it is possible to know in which position the vehicle will be at a specific instant in time. This means one can assume each plane as an obstacle that is at different positions at different instants in time. Then, the next step



**Fig. 4** Trajectory replanning algorithm result

is to parameterize  $\mathbf{P}_{free}$  by time, thus representing different free positions at different moments. The new configuration space is then given by the equation below:

$$\mathbf{P}_{free}^k(t) = \mathbf{P}_{free} - \sum_{j=1}^{k-1} \vec{r}_j(t). \quad (19)$$

For a certain robot  $k$ , during the step of generating each new state in the expansion of the tree, it will be calculated which are the free positions at an instant in time. In order to do that, the planner has to subtract from the free space, based only on the static obstacles ( $\mathbf{P}_{free}$ ), all the positions that will be occupied by other robots at that instant in time  $t$ , where  $\vec{r}_j(t)$  represents the position  $j$  of the robot in its trajectory at instant  $t$ .

Thus, the paths are computed sequentially, where the first UAV considers only the static obstacles already present in the environment and each UAV considers the obstacles and trajectories computed for the other UAVs.

#### 4 Experiments

Our technique was used to plan trajectories for a set of simulated small unmanned aerial vehicles. Those robots were modeled as fixed-wings aircrafts based on a UAV named AqVS (Fig. 5), developed at Universidade Federal de Minas Gerais/Brazil.

**Fig. 5** AqVS, an UAV from the Universidade Federal de Minas Gerais-Brazil [25]



This is a small hand launched hybrid electric motor sail plane, equipped with GPS receptor, barometric altimeter, infrared inclinometer, airspeed sensor and CCD camera, and controlled by a set of PID stabilizers running on a Palm® PDA for autonomous navigation [25]. The AqVS presents the following characteristics:

- $\rho_{min}$ : about 50 m,
- maximum cruising speed: approximately 50 km/h,
- uncertainty of localization: 12 m.

The above values were determined using data from actual flights, considering a speed of approximately 50 km/h. This vehicle has shown to be a good choice for testing our methodology because of its large uncertainty of localization.

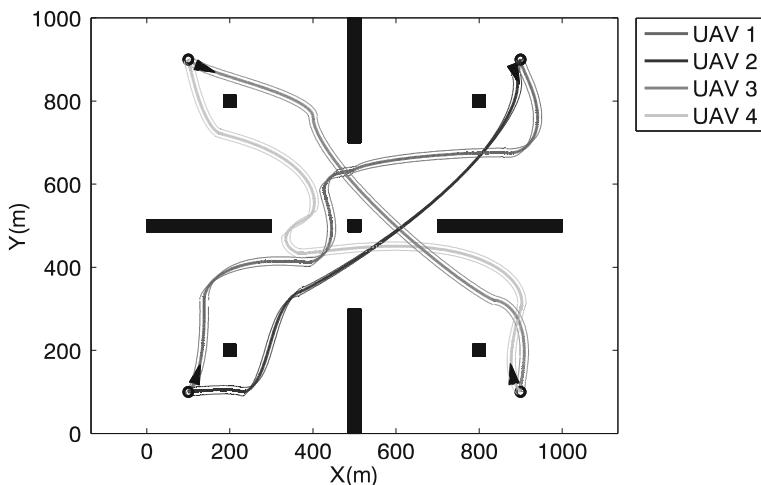
Three experiments were realized with using our approach. In the first experiment, a group of four AqVS/UAVs were used. The objective of each one of those robots was to exchange its position with another, flying through the environment without collision with obstacles or others UAVs. Its initial and goal configurations were chosen as:

- UAV<sub>1</sub>,  $\mathbf{P}_{init}(100, 100, \frac{\pi}{4})$  and  $\mathbf{P}_{goal}(900, 900, -\frac{3\pi}{4})$ ,
- UAV<sub>2</sub>,  $\mathbf{P}_{init}(900, 900, -\frac{3\pi}{4})$  and  $\mathbf{P}_{goal}(100, 100, \frac{\pi}{4})$ ,
- UAV<sub>3</sub>,  $\mathbf{P}_{init}(100, 900, -\frac{\pi}{4})$  and  $\mathbf{P}_{goal}(900, 100, \frac{3\pi}{4})$ ,
- UAV<sub>4</sub>,  $\mathbf{P}_{init}(900, 100, \frac{3\pi}{4})$  and  $\mathbf{P}_{goal}(100, 900, -\frac{\pi}{4})$ .

Figure 6 shows the result of the trajectory planning for all UAVs. In this experiment, we built a simple environment with few obstacles, just to assess the methodology. Although the paths self-intersect in space, the trajectories generated do not intersect for any given time.

One can still observe that the offset path to each PH curve that composes the path, which was calculated by means of Eq. 7. This represent a safety-flight area to the robot navigation because any uncertainty on the robot localization still is contained in  $\mathbf{P}_{free}(t)$ .

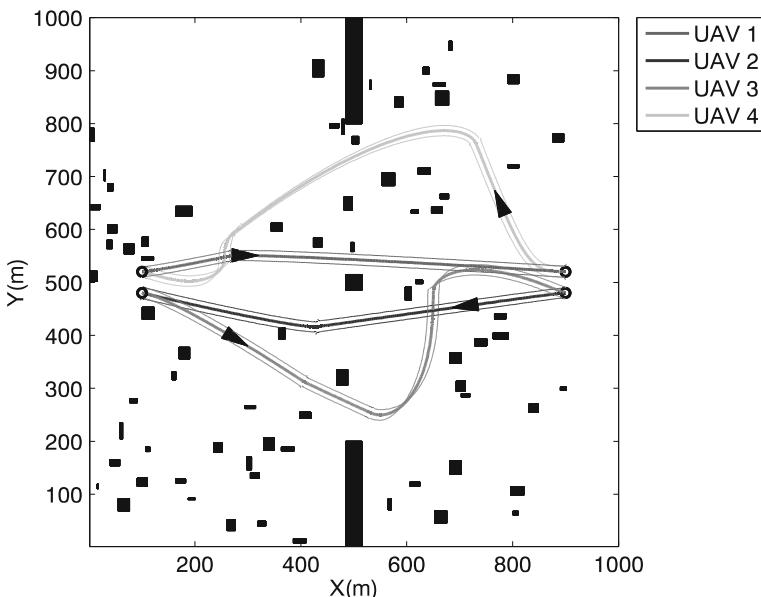
Another experiment was conducted, now considering a more complex environment, with about one hundred static obstacles and four UAVs. Most of the



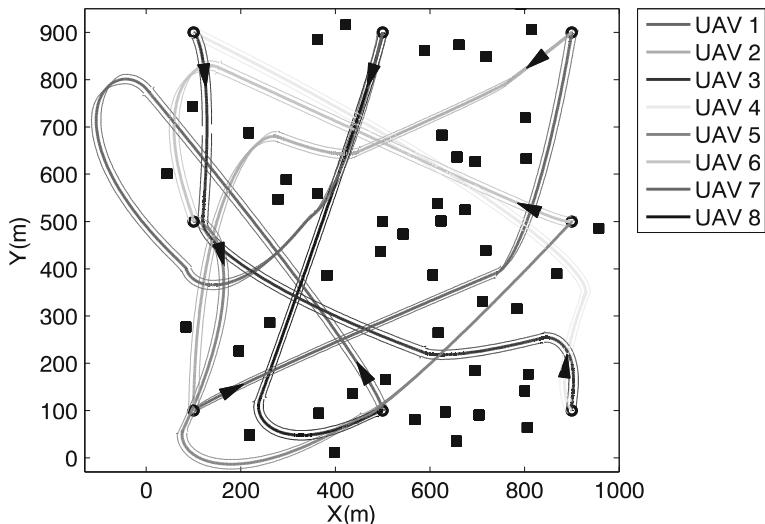
**Fig. 6** Trajectory planning for a set of four AqVS/UAVs in a simple environment

localization and geometry of those obstacles were chosen randomly, as shown in Fig. 7. In this case, the following UAV configurations were used:

- UAV<sub>1</sub>,  $\mathbf{P}_{init}(100, 520, 0)$  and  $\mathbf{P}_{goal}(900, 520, \pi)$ ,
- UAV<sub>2</sub>,  $\mathbf{P}_{init}(900, 480, \pi)$  and  $\mathbf{P}_{goal}(100, 480, 0)$ ,
- UAV<sub>3</sub>,  $\mathbf{P}_{init}(100, 480, 0)$  and  $\mathbf{P}_{goal}(900, 480, \pi)$ ,
- UAV<sub>4</sub>,  $\mathbf{P}_{init}(900, 520, \pi)$  and  $\mathbf{P}_{goal}(100, 520, 0)$ .



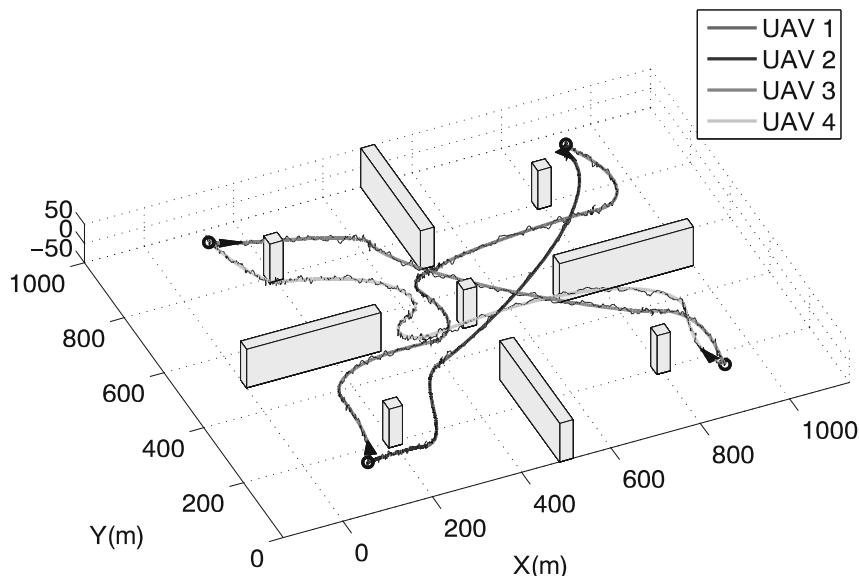
**Fig. 7** Trajectory planning for a set of four AqVS/UAVs in a more complex environment



**Fig. 8** Trajectory planning for a set of eight AqVS/UAVs in an environment with random obstacles

In the last experiment, we exercised our methodology with a set of eight UAVs and fifty obstacles disposed randomly in space. The configurations to the set of robots were chosen as:

- UAV<sub>1</sub>,  $\mathbf{P}_{init}$  (100, 100,  $\frac{\pi}{4}$ ) and  $\mathbf{P}_{goal}$  (900, 900,  $\frac{-3\pi}{4}$ ),
- UAV<sub>2</sub>,  $\mathbf{P}_{init}$  (900, 900,  $\frac{-3\pi}{4}$ ) and  $\mathbf{P}_{goal}$  (100, 100,  $\frac{\pi}{4}$ ),



**Fig. 9** AqVS/UAV trajectory following for the planning in the first test

- UAV<sub>3</sub>,  $\mathbf{P}_{init}$  (100, 900,  $\frac{-\pi}{4}$ ) and  $\mathbf{P}_{goal}$  (900, 100,  $\frac{3\pi}{4}$ ),
- UAV<sub>4</sub>,  $\mathbf{P}_{init}$  (900, 100,  $\frac{3\pi}{4}$ ) and  $\mathbf{P}_{goal}$  (100, 900,  $\frac{-\pi}{4}$ ),
- UAV<sub>5</sub>,  $\mathbf{P}_{init}$  (100, 500, 0) and  $\mathbf{P}_{goal}$  (900, 500,  $\pi$ ),
- UAV<sub>6</sub>,  $\mathbf{P}_{init}$  (900, 500,  $\pi$ ) and  $\mathbf{P}_{goal}$  (100, 500, 0),
- UAV<sub>7</sub>,  $\mathbf{P}_{init}$  (500, 100,  $\frac{\pi}{2}$ ) and  $\mathbf{P}_{goal}$  (500, 900,  $\frac{-\pi}{2}$ ),
- UAV<sub>8</sub>,  $\mathbf{P}_{init}$  (500, 900,  $\frac{-\pi}{2}$ ) and  $\mathbf{P}_{goal}$  (500, 100,  $\frac{\pi}{2}$ ).

Figure 8 shows the final result of the trajectory planning for all UAVs. Once more, there are no collisions.

The planned trajectories were tested with the dynamic model of the AqVS model implemented in Matlab. Figure 9 shows the final result for the first planning.

## 5 Conclusion and Future Works

We have described a novel technique that allows the planning of trajectories for multiple vehicles in a very complex, with a large number of obstacles. While the technique of RRT already allows the generation of smooth paths for nonholonomic vehicles, in some cases, the kinematic and dynamic models used in the integration step of the algorithm can become very complex. There is a great need for a reliable model of the vehicle, otherwise there is a chance that the generated path may not be achieved by a real robot. This is the case a real aircraft, where some of its aerodynamic coefficients are very difficult to model.

The use of analytical curves, such as Bézier curves, allows for greater flexibility of this model with a low computational cost. The PH curves, in particular allows for the calculation of offset curves and the parametric speed functions directly. The project of these curves takes into account very simple kinematics and dynamics constraints, which implies the simplification of the model of the vehicle around a few points of operation.

An important advantage of our method is the reduction in the computational time to convert the RRT algorithm, because the use of PH curves enable connections between vertices of the tree with unlimited range. In an environment with few obstacles, a very small quantity of vertices (sometimes only two) is sufficient to take the robot from  $\mathbf{P}_{init}$  to  $\mathbf{P}_{goal}$ .

As future work, we plan to expand the use of the technique for the three-dimensional space, considering other kinematic constraints, as the maximum torsion and maximum climb (dive) angle. It is possible to use an expansion of the PH curve to three dimensions, while maintaining the same benefits described in this paper.

Another possibility is to evaluate other metrics for  $\rho$  in the RRT algorithm, since the presented here cannot be applied in the three-dimensional case. Perhaps a metric based on the actual length of the PH curve might be a better solution, despite their higher computational cost.

**Acknowledgements** The authors gratefully thank prof. Paulo Iscold for the flight model of the AqVS/UAV. This work was developed with the support of CNPq, CAPES and FAPEMIG.

## References

1. Siegwart, R., Nourbakhsh, I.R.: Introduction to Autonomous Mobile Robots. MIT, Cambridge (2004)
2. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
3. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am. J. Math.* **79**, 497–516 (1957)
4. Kuwata, Y., Richards, A., Schouwenaars, T., How, J.P.: Robust constrained receding horizon control for trajectory planning. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2005)
5. Wzorek, M., Doherty, P.: Reconfigurable path planning for an autonomous unmanned aerial vehicle. Hybrid Information Technology, 2006. ICHIT '06. International Conference on vol. 2, pp. 242–249 (2006)
6. Bortoffl, S.A.: Path planning for UAVs. In: Proceedings of the American Control Conference (2000)
7. Dogan, A.: Probabilistic path planning for UAVs. In: Proceedings of 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations (2003)
8. Cheng, P., Shen, Z., LaValle, S.: RRT-based trajectory design for autonomous automobiles and spacecraft. *Arch. Control Sci.* **11**(3–4), 167–194 (2001)
9. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Maximizing miniature aerial vehicles. *Robot. Autom. Mag. IEEE* **13**(3), 34–43 (2006). doi:10.1109/MRA.2006.1678137
10. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Advances in unmanned aerial vehicles: state of the art and road to autonomy, chap. Obstacle and Terrain Avoidance for Miniature Aerial Vehicles, pp. 213–244. Springer, Tampa (2007)
11. Schouwenaars, T., How, J., Feron, E.: Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2004)
12. Shanmugavel, M., Tsourdos, A., Zbikowski, R., White, B.A., Rabbath, C.A., Léchevin, N.: A solution to simultaneous arrival of multiple UAVs using Pythagorean Hodograph curves. In: Proceedings of the IEEE American Control Conference (ACC), pp. 2813–2818. Minneapolis (2006)
13. Li, T.Y., Chou, H.C.: Motion planning for a crowd of robots. *Proc. IEEE Int. Conf. Robot. Autom.* **3**, 4215–4221 (2003)
14. Belta, C., Kumar, V.: Abstraction and control for groups of robots. *IEEE Trans. Robot.* **20**(5), 865–875 (2004)
15. Warren, C.: Multiple robot path coordination using artificial potential fields. *Robotics and Automation, 1990. Proceedings, 1990 IEEE International Conference on* vol. 1, pp. 500–505 (1990)
16. Marcolino, L.S., Chaimowicz, L.: No robot left behind: coordination to overcome local minima in swarm navigation. In: Proceedings of the 2008 IEEE International Conference on Robotics and Automation (2008)
17. Gayle, R., Sud, A., Lin, M., Manocha, D.: Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 3777–3783 (2007)
18. Leroy, S., Laumond, J.P.: Multiple path coordination for mobile robots: a geometric algorithm. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1118–1123 (1999)
19. Kreyszig, E.: Differential Geometry, vol. 1. Dover, New York (1991)
20. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Tech. Rep., Computer Science Dept., Iowa State University (1998)
21. Farouki, R.T., Sakkalis, T.: Pythagorean Hodographs. *IBM J. Res. Develop.* **34**(5), 736–752 (1990)
22. Farouki, R.T., Neff, C.A.: Hermite interpolation by Pythagorean Hodograph quintics. *Math. Comput.* **64**, 1589–1609 (1995)
23. Farouki, R.T.: The elastic bending energy of Pythagorean Hodograph curves. *Comput. Aided Geom. Des.* **13**, 227–241 (1996)
24. Shkel, A.M., Lumelsky, V.: Classification of the Dubins set. In: *Robotics and Autonomous Systems*, vol. 34, pp. 179–202 (2001)
25. Iscold, P.: Development of a small unmanned aerial vehicle for aerial reconnaissance. In: International Congress of Mobility Engineering. São Paulo, Brazil (2007)

# Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs

Emre Koyuncu · N. Kemal Ure · Gokhan Inalhan

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 23 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** In this work, we consider the problem of generating agile maneuver profiles for Unmanned Combat Aerial Vehicles in 3D Complex environments. This problem is complicated by the fact that, generation of the dynamically and geometrically feasible flight trajectories for agile maneuver profiles requires search of nonlinear state space of the aircraft dynamics. This work suggests a two layer feasible trajectory/maneuver generation system. Integrated Path planning (considers geometrical, velocity and acceleration constraints) and maneuver generation (considers saturation envelope and attitude continuity constraints) system enables each layer to solve its own reduced order dimensional feasibility problem, thus simplifies the problem and improves the real time implement ability. In Trajectory Planning layer, to solve the time depended path planning problem of an unmanned combat aerial vehicles, we suggest a two step planner. In the first step, the planner explores the environment through a randomized reachability tree search using an approximate line segment model. The resulting connecting path is converted into flight way points through a line-of-sight segmentation. In the second step, every consecutive way points are connected with B-Spline curves and these curves are repaired probabilistically to obtain a geometrically and dynamically feasible path. This generated feasible path is turned in to time depended trajectory with using time

---

E. Koyuncu (✉) · N. K. Ure  
Controls and Avionics Laboratory, Istanbul Technical University, Istanbul, Turkey  
e-mail: emre.koyuncu@itu.edu.tr

N. K. Ure  
e-mail: ure@itu.edu.tr

G. Inalhan  
Faculty of Aeronautics and Astronautics, Istanbul Technical University, Istanbul, Turkey  
e-mail: inalhan@itu.edu.tr

scale factor considering the velocity and acceleration limits of the aircraft. Maneuver planning layer is constructed upon multi modal control framework, where the flight trajectories are decomposed to sequences of maneuver modes and associated parameters. Maneuver generation algorithm, makes use of mode transition rules and agility metric graphs to derive feasible maneuver parameters for each mode and overall sequence. Resulting integrated system; tested on simulations for 3D complex environments, gives satisfactory results and promises successful real time implementation.

**Keywords** Path planning · Maneuver planning · Dynamic feasibility · Agile maneuvering · Unmanned combat aerial vehicles

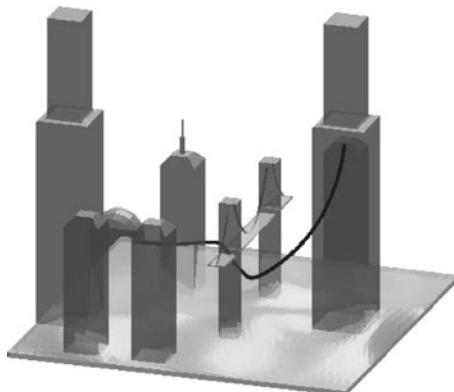
## 1 Introduction

Practical usage of Unmanned Air Vehicles has underlined two distinct concepts at which these vehicles are instrumental. First are the routine operations such as border or pipeline monitoring for which manned systems are expensive and inefficient. Second are scenarios such as an armed conflict reconnaissance or nuclear spill monitoring, in which there is a high risk for human life loss as the proximity to the scenario increases. In this work, we consider a specific case of the second type of scenarios which involves flying through a complex and dense city-like environment rather this be for reconnaissance or monitoring.

Although many kinodynamic motion planning methods that declares generating *dynamically feasible path* have been developed, they rarely can be used in practice especially for the aerial vehicles because of computational complexities. General kinodynamic motion planners require at least exponential time in that dimension of the state space of dynamical systems which is usually at least twice the dimension of the underlying configuration space [11]. In practice, kinodynamic planners are implementable only for systems that have small state-space dimensions. For example, the work presented in [11] suggests a path-planning relaxation which defines a class of maneuvers from a finite state machine, and uses a trajectory based controller to regulate the unmanned vehicle dynamics into these feasible trajectories. However, the trajectories to be controlled are limited to the trajectories generated by the finite state machine and the computational challenges of generating real-time implementable flight trajectories in 3D complex environments still remains as a challenge. Demonstration of path planner solution for flight in the 3D crowded *MelCity* model and landing to base is seen in Fig. 1.

The most important advantage of combination of path and maneuver planning that proposed in this work is related with generation of feasible agile flight trajectories. When the path planning problem is complicated with dynamic constraints on vehicle along with geometrical constraints, problem becomes challenging due to complexity of dimension. Therefore one cannot simply generate a trajectory that is dynamically feasible (feasible in the sense that it would be trackable by a control system in the flight envelope and actuator limits), relying on the path planning. This is where the maneuver planning comes in; by taking the advantage of working with a pre-defined dynamically feasible flight trajectory given by trajectory planner, it can identify and extract appropriate angular velocity and angular attack information

**Fig. 1** UCAV flight demonstration in the 3D complex city-like environment and landing to its base



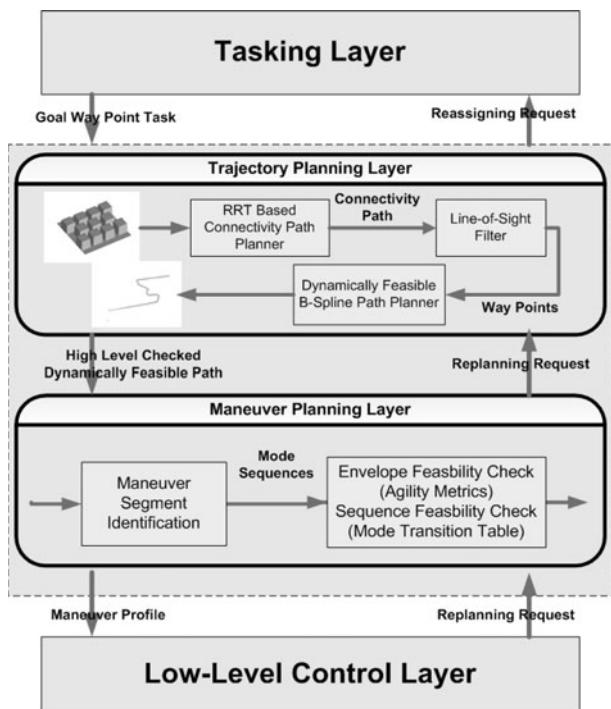
needed to create a maneuver profile without concerning with the trajectory generation and obstacle avoidance. However if the flight trajectory is generated without checking the velocity and acceleration bounds, maneuver planning layer will have hard time searching for angular velocities and angle of attack history that are in the feasible set. Therefore by sharing the dynamic feasibility checks between path planner and maneuver planner, these two layers covers their disadvantages and provides both dynamically and geometrically feasible flight trajectories and maneuver profiles for complex environments. This two step feasibility check is the main contribution of this paper to the field.

In the *Trajectory Planer* layer, we suggest a real-time implementable two-step path planner strategy. As the first step, 3D environment and the passages are rapidly explored using an RRT based planner. From this geometrically feasible but not dynamically feasible path, line-of-sight critical milestones are extracted. Although these milestones allow point-to-point flyable flight path segmentation, it does not necessarily correspond to a fast agile and continuous motion plan. To address this, as a second step, B-Spline method is used for generating  $C^2$  continuous flight path that pass through these milestones. In face of geometrically and dynamically unfeasibility, generated path is probabilistically reshaped to eliminate the collisions and dynamically unfeasibility thanks to local support property of the B-Spline curves and at the end the *time scale* is adjusted to allow dynamic achievability.

Main contribution of the *Maneuver Planner* layer is a new perspective on maneuver/motion planning algorithms, which does not require any pre-build maneuver libraries and relies on the parameterized modal decomposition of arbitrary flight maneuvers. Integration with a path planning algorithm results in capability of generating feasible flight trajectory and maneuver profile which can be tracked by a switched control system where every maneuver mode is locally controllable. Integrated architecture of the Path/Maneuver Planning is demonstrated in the Fig. 2.

Rest of paper is organized as follows. In Sections 2 and 3, we gave detailed descriptions of the Path Planner and Maneuver Planner respectively. Literature surveys and related framework has been given at the beginning of these sections. In Section 4, example solution of the integrated Path/Maneuver Planning is demonstrated for the selected environment and complete computational time-table for the other example environments also has been given. The conclusions are discussed in Section 5.

**Fig. 2** Integrated planning architecture of UCAVs



## 2 Dynamically Feasible Path Planning Algorithm

For developing a real-time implementable planner, motion planning researches have been focused on sampling based approaches that rapidly search either the configuration or the state space of the vehicle. In the last few decades, sampling-based motion planning algorithms have shown success in solving challenging motion planning problems in complex geometries while using a much simpler underlying dynamic model in comparison to an air vehicle. Roadmap-based planners, like well-known Probabilistic Road Mapping (PRM) method as mentioned in [18], are typically used as multi-query planners (i.e. simultaneous search of the environment from different points) that connect these multiple queries using a local planning algorithm. PRM planners converge quickly toward a solution path, if one exists, as the number of milestones increases. This convergence is much slower when the paths must go through narrow passages. For complex environments, some extended algorithms are suggested for PRM like planners in [3] and [14]. Tree-based planners build a new roadmap for each query and the newly produced samples are connected to the samples that are already exists in the tree as in [13, 17], and [23]. Rapidly-Exploring Random Tree (RRT) is the most popular representative of tree-based planners that is an exploration algorithm for quickly searching high-dimensional spaces that have both global and differential constraints. Sampling-based planners, especially tree-based planners (RRT and single-query PRM variants), have been adapted to solve dynamically feasible paths that accommodate kinodynamic constraints.

Kinodynamic planning refers to problems in which the motion must satisfy non-holonomic and/or dynamic constraints. The main philosophy behind kinodynamic planning is searching a higher dimensional state space that captures the dynamics of the system [16, 23].

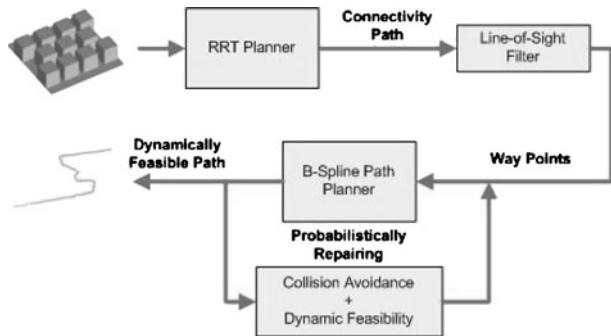
*Gradual motion planning* methods -our approach can be represented in this class- are recently proposed to solve complex path planning problem in cluttered environments. These methods first solve a relaxed form of the problem and then the approximate solution is refined to solve the original problem with a repairing method. In [15], a roadmap is initially generated by allowing some penetration into the collision workspace. Later, milestones are carried to collision-free space. In Iterative Relaxation of Constraints (IRC) method [1], first a relaxed version of problem is solved and then this coarse solution is used as a guide to solve original problem iteratively. The strategy of using an approximate solution to obtain a collision-free path is also used in Lazy PRM [2] and C-PRM [30]. In our earlier work [22], using a similar strategy, a Mode Based PRM method is refined with modal flight-path segments to obtain flyable trajectories.

From a deterministic perspective, B-Spline curves have been used in many dynamic path planning and control problem implementations. In [20], dynamic trajectory is generated with the minimum travel time for two-wheeled-driven type mobile robot. In [25] visibility-based path is modified to continuous feasible path via B-Spline curves. Using the well known local support property of B-Spline curves, real-time path modification methods are proposed for multiple mobile robots in [28] and robot manipulators in [7]. Constant acceleration time-scalable path generation method for the unmanned helicopters flying in the urban environments is used in our earlier work in [21] that we will use similar method but this time for the unmanned combat aerial vehicles.

In comparison, our method utilizes both the probabilistic and the deterministic aspects to obtain a real-time implementable planner strategy. In the first step, the algorithm rapidly explores the complex environment and the passages using an RRT planner because of its well quick spreading ability. In this part, our strategy focuses only finding an obstacle-free path that can be tracked from the initial point to the goal point with line segments in the configuration space. Vehicle's dynamic constraints are completely disregarded to decrease the computational time. This coarse obstacle-free path will be called as *connectivity path*. After finding the connectivity path, this path is filtered with the line-of-sight implementation to eliminate the points that cause long detours. Remained points that we call as *way points* naturally appear in entering and exiting regions of the narrow passages that are formed between the obstacles. An advantage of this refinement is that we can use these way points as guide-milestones that point hard regions and directions of the next coming hard regions in the environment.

In the second step of our strategy that dynamically feasible path is searched, every way-point connected with forth-order B-Spline curve and collision and dynamic feasibility cases are checked on curve. These forth-order B-spline curve presents  $C^2$  continuous flight path. If the generated curve is not feasible, probabilistic repairing is achieved by randomized waypoint expansion on the connecting line path and the unit flight time is expanded to limit the accelerations within controllable regime. Since B-Spline curves have local support property, these repairing processes can be made on local path segments of interest. All path planning process is illustrated in Fig. 3.

**Fig. 3** Dynamically feasible path planning process



## 2.1 First Step: Connectivity Path

In real-time applications, planners should be able give a reliable answer in minimal permitted time slot. In motion planning problem, especially in complex environments, it is hard to say when planners should stop searching or change the searching strategy (i.e. switching to a more complex planner etc.). Moreover, finding an obstacle free geometrical path does not necessarily mean that a dynamically feasible path can be implemented by the vehicle exists. Although geometrical paths can be implemented via point to point navigation by the helicopter like vehicles with a inefficient manner but this flight strategy is not applicable for agile combat vehicle operations in under-threat environments. For the vehicles that have complex dynamics like combat aerial vehicles, directly searching in high dimensional state-space—as kinodynamic planners do—consumes long computational time to find a feasible path. Specifically, in our earlier work [22], we observed that before the major feasible path planning phase, defining the geometrical obstacle free path and trackable way points significantly accelerates the searching ability and decreases the total computational time of planner.

For finding connectivity path, RRT algorithm is used because of its rapid spreading ability. RRT is considered as being an efficient algorithm to search even high dimensional spaces. However, one of the important drawbacks of using RRT as a stand-alone planner is biasing of the distribution of milestones towards the obstacle regions if the configuration space has large obstacles. Bi-directional RRT method shows performance more than single tree approach but it has also discontinuity problem on the connection points of the paths. Therefore, we choose to use single Goal-Biased RRT [23] approach that converges to goal configuration rapidly. We tested performance of the algorithm in different complex environments to conserve both rapid converging to solution and spreading abilities of the RRT, we chose the 50% percent goal biasing value. In this phase, we are only motivated by RRT's good property to obtain connectivity path. Our strategy does not focus on dynamically feasibility in this part of the path planner. Therefore, RRT algorithm is only used for searching configuration-space of the vehicle with primitive maneuvers that includes level and climbing flight and changing instantaneous heading direction. Construction of connectivity path algorithm is given as Goal Biased RRT Algorithm.

In Algorithm 1, to find the connectivity path, Goal Biased RRT method is used that one single tree is extended from the initial point. Each loop attempts to extend the  $\tau$  tree first toward the random selected point  $m_{rand}$ , and second toward the goal

**Algorithm 1:** Goal Biased RRT Algorithm

---

```

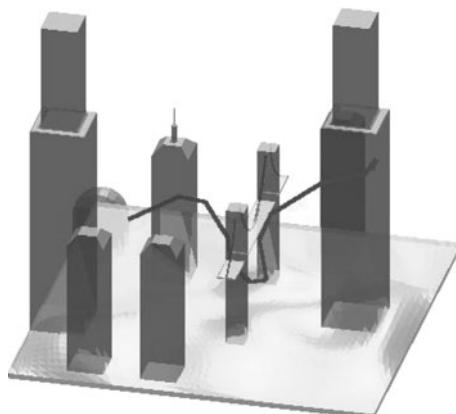
input : initial configuration  $q_{init}$  and goal configuration  $q_{goal}$ 
output: connectivity path
1  $\tau \leftarrow q_{init}$  and  $i \leftarrow 1$ 
2 repeat
3   Select random point  $m_{rand}$  in  $C$  and its neighbor point  $m_{near}$  in  $\tau$ 
4   Generate  $m_{new}$  is gone with trajectory  $e_{new}$  from  $m_{near}$  toward  $m_{rand}$ 
5   if  $e_{new}$  is in  $C_{free}$  then
6      $\tau \leftarrow m_{new}$  and  $i + 1$ 
7     if  $m_{new}$  is in end region then
8       break with success
9   Select neighbor point  $m_{near}$  of  $q_{goal}$  in  $\tau$ 
10  Generate  $m_{new}$  is gone with trajectory  $e_{new}$  from  $m_{near}$  toward  $q_{goal}$ 
11  if  $e_{new}$  is in  $C_{free}$  then
12     $\tau \leftarrow m_{new}$  and  $i + 1$ 
13    if  $m_{new}$  is in end region then
14      break with success
15  if  $i = N$  max iteration number then
16    break with fail
17 until end region is reached with success
18 Select connectivity path can be gone back from end region to initial point in  $\tau$ 

```

---

point by adding new points. To expand the tree, nearest point already within the  $\tau$  tree to the sampled random point (in Line 3) and the nearest point to the goal point is selected (in Line 9) respectively in every one loop. *Generate* function generates new points  $m_{new}$  on the direction of the selected nearest points  $m_{near}$  at random selected distances as shown in Line 4 and 10. If direction angles exceed predefined limits, max direction angles are selected. These boundaries should be chosen according to vehicle's kinematic boundaries. If new generated point and trajectory is within obstacle-free configuration (checked in Line 5 and 11) then  $m_{new}$  is added  $\tau$  tree as shown in Line 6 and 12. If  $\tau$  tree reaches *end region* anytime, algorithm returns *connectivity path*. *End region* can be obtained within a tolerable capture region as explained in [19]. A solution of the algorithm in a complex city-like environment is illustrated in Fig. 4.

**Fig. 4** Demonstration of the RRT based finding connectivity path algorithm



**Algorithm 2:** Line-of-Sight Filtering

---

```

input : connectivity path
output: way point set  $WP$ 
1  $m_{visib} \leftarrow m_1$  and  $m_i \leftarrow m_2$ 
2 repeat
3   | Generate line  $\ell_{visib}$  from  $m_{visib}$  to  $m_i$ 
4   | if  $\ell_{visib}$  is collide with  $C_{obst}$  then
5   |   |  $WP \leftarrow m_{i-1}$ 
6   | else
7   |   |  $i + 1$ 
8 until last point of connectivity path is reached

```

---

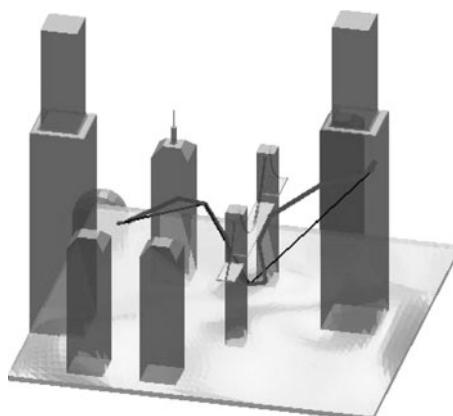
Because of the RRT's extending strategy and our simplifications, undesirable detours are frequently seen in obtained *connectivity path*. Since we only consider finding the obstacle-free region; we can simply remove the points that cause these detours. In this phase of our strategy, *connectivity path* is refined by Line-of-Sight Filter algorithm that erases points that result in useless fluctuations with using a line-of-sight arguments. As can be seen in Fig. 5, remaining points generally appear in nearby entering and exiting field of the narrow passages and inherently hard regions. Hence, these guard points also indicate where hard regions are beginning, what the direction of the next-coming hard region. These points also give a sense of agile maneuvering that are needed to fly over these points.

In this part of algorithm, a simple iteration checks if the selected point  $m_{visib}$  can connect with the previous points in *connectivity path* with a line segment without colliding with any obstacle. If the line segment collides with an obstacle, in other words, if the current point cannot be connected to the selected point, last connectible point is added to the *way point* sequence and the subsequent search continues from this point. This algorithm runs until the last point of *connectivity path* is reached with a line segment. A solution is illustrated in Fig. 5.

## 2.2 Second Step: Dynamically Feasible B-Spline Algorithm

In the first step, generated connectivity path with straight line segments result in a simple and implementable piecewise flight plan. However, this flight plan is not a fast

**Fig. 5** Demonstration of the refining connectivity path with the line-of-sight filter algorithm



agile and continuous motion plan - a desirable feature in many complex unmanned combat aerial vehicles applications. After obtaining the way points (we will call remaining points as *way point* set) on the environment, many deterministic and sampling based path planner methods can be used to find the dynamically feasible path between the way points. Moreover, generated path must be continuous on the way points dynamically. During the path generation phase, since feasibility is desired, trajectory generation method should allow reshaping to supply collision avoidance and dynamic feasibility. Therefore, local support is also a desirable property on the path generation method. Local support means that the paths only influence a region of the local interest. Thus, obstacle avoidance and dynamic-feasibility repairing can be achieved without changing the whole shape of the generated path. B-Spline approach can supply these main requirements. An overview of B-Spline can be found in [29].

Basically, output  $C(u)$  can be defined in terms an  $k$  order B-Spline curve;

$$C(u) = \sum_{i=0}^n P_i N_{i,k}(u) \quad 0 \leq u \leq u_{max} \quad (1)$$

The coefficients  $P_i$  in Eq. 1 are called control points that will represent way points and pseudo way points in our approach.

The B-Spline basis functions  $N_{i,k}$  are given by the Cox De Boor recursion;

$$N_{i,0}(u) = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \quad (3)$$

A B-Spline curve can be constructed from Bezier curves joined together with a prescribed level of continuity between them. A nondecreasing sequence of real numbers  $U = [u_0 \dots u_{max}]$  is called the knot vector. Frequently, the knot points are referred to as the break points on curve [26]. B-Spline basis function  $N_{i,k}$  is zero outside the interval  $[u_i, u_{i+k}]$  and non-negative for all values of  $k, i$  and  $u$ .

Derivatives of B-Spline curve exist on the knot vector span. Since, the  $k$ th-order B-spline is actually a degree  $(k-1)$  polynomial, produced curve can be differentiated  $k-1$  times.

$$C(u)^{(j)} = \sum_{i=0}^n P_i N_{i,p}^{(j)}(u) \quad 0 \leq u \leq u_{max} \quad (4)$$

A valuable characteristic of the B-Spline curves is that the curve is tangential to the control polygon (formed by the control points) at the starting and ending point if some modifications are supplied. This characteristic can be used in order to define the starting, ending and transition directions of the curve by inserting an extra *pseudo control points* in directions which are defined according to way points' orientations assigned in the first step as explained in [26].

In this strategy, we choose generate forth-order path B-Spline (cubic polynomials) to obtain continuous inertial velocity and acceleration.

For generating B-Spline trajectory pass through way points, two *pseudo control points* are inserted after and before the every waypoint(except initial and last way point) in direction of the their tangent vector that these tangent directions are assigned during the path planning step. Note that; for the first way point, only further pseudo way point and for the last way point, only back pseudo way point should be added to way point set. The distance value between the way-point and the added pseudo-way points will define the transition velocity and acceleration on the way points of the path. Hence,  $C^2$  continuity, in other words, continuous velocity and acceleration transition is naturally achieved on the way points.

For generate cubic B-Spline curves, we use specific nonuniform knot vector form  $\mathbf{U} = [0\ 0\ 0\ 0\ U_{mid}\ 1\ 1\ 1\ 1]$  to obtain the coincidence between the first and last control points and the first and the last ends of the generated B-Spline curve respectively. Detailed information about this effect can be found in [29] as *open uniform knot vector* effect.  $\mathbf{U}_{mid}$  is represents middle *knot vector* that is initially uniformly distributed in  $(0, 1)$  interval -number of points depends on the number of control points- and algorithm can add new knot points to the vector without preventing its uniform form. We choose using arbitrary  $[0, 1]$  interval for parameter  $u$  such that it represents unit-time scale [38]. This property is later used to allow dynamic feasibility via time scaling (i.e. expanding the time horizon of the maneuver). Overall B-Spline path planning algorithm can be demonstrated in Algorithm 3.

This algorithm tries to find dynamically feasible B-Spline curve passes through on the way points with their heading angles and runs until the last way point is connected with a feasible path. Initially,  $m$  number way points - generated in the first step- are added in *control point* set  $\mathbf{P}$  as seen in Line 2. Then, for every way point, except first and last way point, *back* pseudo way point  $gp_{i,b}$  and *further* pseudo way point  $gp_{i,f}$  is located on random selected distance  $d$  from way points on their heading tangent directions and these *pseudo way point* set  $\mathbf{gp}$  is also added in *control point* set  $\mathbf{P}$  that is demonstrated in Line 3 to 6. Different from other way points, for the first way point, only *further pseudo way point*  $gp_{1,f}$  is located and for the last way point, only *back pseudo way point*  $gp_{m,b}$  is located. Thus, algorithm initially begins with  $3m - 2$  control points where  $m$  indicates that number of way points but note that the algorithm can add new control points during to implementation to repair the B-Spline. As initial form, open uniform knot vector form that is chosen in unit interval  $[0,1]$  is used in our implementation as shown in Line 8. As depicted in Line 9, in a loop, B-Spline basis function is generated via  $u$  parameter and then collision and dynamic feasibility is checked on every discrete point of the curve as shown in Line 10. Since velocity and acceleration on the path is a function of time, for each point of the trajectory we have to check if the instantaneous velocity and acceleration is within the limits of the flight envelope. This Dynamic Feasibility check is done by checking the first and the second derivatives of the B-Spline curve which gives the velocities and accelerations of the aircraft respectively. If these velocity and acceleration values are within the limitations of the aircraft (*flight envelope*) using chosen *unit time scale*, generated path segment is accepted as dynamically feasible. One of the most critical step in the path planning layer is to determine the velocity and accelerations on the trajectory; if the aircrafts velocity constraints are not taken into account, maneuver planning layer would not be able to find feasible maneuver reference from this generated trajectory. This concept is used for giving a sense on *Dynamic Feasibility* on the side of the Path Planning layer as a first step

**Algorithm 3:** Dynamically Feasible Trajectory Generating with B-Spline

---

```

input : way point set  $\mathbf{g} = [g_1 \dots g_m]$ 
output: dynamically feasible path
1  $TimeScale \leftarrow$  unit-time scale
2  $\mathbf{P} \leftarrow \mathbf{g} = [g_1 \dots g_m]$  as control point set
3 foreach element  $g_i$  of the  $\mathbf{g}$  do
4   Insert back pseudo way point  $gp_{i,b}$  to the random selected distance  $d$  from  $g_i$  on its negative
      heading direction
5   Insert further pseudo way point  $gp_{i,f}$  to the random selected distance  $d$  from  $g_i$  on its positive
      heading direction
6  $\mathbf{P} \leftarrow \mathbf{gp} = [gp_{1,f} \ gp_{2,b} \ gp_{2,f} \dots gp_{m-1,b} \ gp_{m-1,f} \ gp_{m,b}]$  as control point set
7  $\mathbf{U} \leftarrow [0 \ 0 \ 0 \ 0 \ \mathbf{U}_{mid} \ 1 \ 1 \ 1]$ 
8 for  $u \leftarrow 0$  to 1 do
9   Evaluate B-Spline basis function
10  Check collision and dynamic feasibility
11  if collision occurs or point of the spline is not dynamically feasible then
12    Change locations of the pseudo way-points  $gp_{i,b}, gp_{i,f}$  of the interest curve segment
        according to  $u$ 
13    Set  $u$  value as indicates that local interest curve segment  $-k/2$  segment
14     $m_1++$ 
15    if  $m_1 > M_1$  then
16      Change locations of the way points  $g_i$  and its pseudo way-points  $gp_{i,b}, gp_{i,f}$  of the
        interest curve segment according to  $u$  in its small region
17      Set  $u$  value as indicates that local interest curve segment  $-k/2$  segment
18       $m_2++$  and  $m_1 = 0$ 
19      if  $m_2 > M_2$  then
20         $\mathbf{P} \leftarrow P_{new}$  as new control point around unfeasibility and update knot vector
21        Set  $u$  value as indicates that local interest curve segment  $-k/2$  segment
22         $m_3++$  and  $m_1, m_2 = 0$ 
23        if  $m_3 > M_3$  then
24          Change  $TimeScale$  to insert min/max values of velocity and acceleration in
            dynamically feasible region
25          Set  $u$  value as 0
26           $m_4++$  and  $m_1, m_2, m_3 = 0$ 
27          if  $m_4 > M_4$  then
28            break with fail
29 Set  $TimeScale$  as generated path can be implemented as possible as in optimal time
30 return B-SplineTrajectory

```

---

**Dynamic Feasibility check.** The more precise Dynamic Feasibility check will be done on Maneuver Planning layer as explained in following section. Maneuver planer layer may also request from Path Planner to re-plan of the trajectory if dynamic feasibility can not be repaired during Maneuver planning using the *Replanning request* connection seen in demonstration of the all interactions between the layers (Fig. 2).

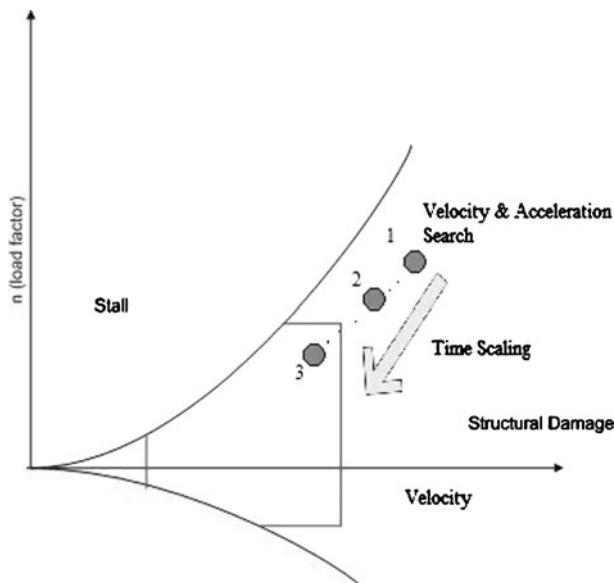
If feasibility cannot be obtained during the path planning, repairing methods are implemented hierarchically. Firstly, location pseudo control points are slid on the same tangent directions as shown in Line 12 and the algorithm decreases the  $u$  value from current interest curve segment  $-k/2$  curve segment where  $k$  is represents order that is four in this implementation. Note that, B-Splines' local support property

allows local control over the generated spline. Specifically, this control is over the curve segments with  $\pm k/2$  polygon spans around the displaced or newly added point. Therefore, when any changes are made on spline, instead of evaluating all spline over and over again,  $u$  value is decreased by value interval that spans local interest. Note that, all the repairing steps are tried with predefined threshold iteration times illustrated as  $M_{is}$  in the algorithm. After predefined number of trials, if the spline cannot be repaired, the way point and its pseudo way points within local interest are carried to a new collision-free locations and these locations are chosen as small random-selected distance away from the prior locations as seen in Line 16.

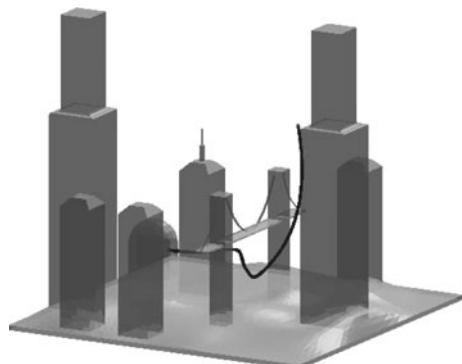
If the B-Spline still cannot be repaired, new control point  $P_{new}$  is added in control point vector  $\mathbf{P}$  around the region in which collision or infeasibility has occurred (Line 20). Since we know the infeasible knot value and its interval in the knot vector, new knot point is added to the midpoint of the infeasible knot interval. Hence, only a limited interval of the knot vector  $\mathbf{U}$  is updated. Reader should remember, only  $\pm k/2$  polygon spans around the displaced or newly added point will be effected with this change.

If all these processes can not repair the path, the *time scale* value is scaled in Line 24 to reallocate the min-max velocity and acceleration interval of the trajectory (time depended path) within the dynamically feasible interval that can be achieved by the aircraft (falls into limits of flight envelope). For example in Fig. 6, search begins in a point outside the flight envelope and in two steps it is moved into limits of flight envelope by time scaling. Finding the feasible velocity-acceleration by this method is similar to what authors had done for finding the feasible modal inputs in the [22]. Note that other than flight envelope check there isn't any dynamic model involved in path planning layer. All the other feasibility problems (actuator saturation, attitude discontinuity etc.) are left to maneuver planning layer.

**Fig. 6** Feasible velocity—acceleration search on flight envelope



**Fig. 7** Dynamically feasible path solution of the B-Spline planner algorithm in the complex 3D environment for UCAV



The end result is a time expanded or shortened flight path. Dynamic feasibility of the all generated spline should be checked from the beginning considering to newly changed time scale. After the generating B-Spline, *TimeScale* is also set again to fly over the all path in optimal time interval. Dynamically Feasible Path solution in the *MelCity* model for the UCAV is demonstrated in Fig. 7.

### 3 Maneuver Planning Algorithm

Multi Modal control framework basically consists of decomposition of the arbitrary maneuvers into set of maneuver modes and associated maneuver parameters. The main aim of the work was the help to reduce complexity of the both planning and control part. Complexity of maneuver planning part has been reduced by reducing the dimension of the problem (modal sequence has strictly lower dimension than state space description) and control part was relaxed by designing specific controllers for each mode and switch between them in order track maneuver mode sequence instead of designing a single controller for maneuver tracking over full flight envelope. In this paper we shall only focus on maneuver planning part, discussion of the switched control layer can be found on [34] and [35]. Motion planning problem for aerospace vehicles are complicated by the fact that, planners based on optimal performance begins to fail in means of computation, when one takes into account of constraints related with dynamical equations of aircraft. Due to fact that, aircrafts state space is at least 12 dimensional, input-state search becomes too complicated; therefore such planners are only successful for vehicles with small state space dimensions [11]. To reduce the complexity of this problem, motion description languages and quantized control concept have been adopted into motion planning [4]. Motion description languages, makes use of classified combination of simplified control laws to track generalized outputs. Most of these languages are strongly connected with the concept of hybrid systems, which in general, classifies the motion by using discrete states which switches in between according to input and state information and each discrete state having its own continuous dynamics. A subclass of such languages which is based on classification of behavior (or reaction) of the dynamic systems, has been successfully adapted for non-holonomic robotic

systems [36]. More recently, closed loop hybrid control systems were developed based on linear temporal logic for the same purpose by [10]. For aerospace vehicles, a hybrid model for aircraft traffic management was developed in [5]. Study showed that, hybrid system representation gives opportunity to calculate reachable sets of the system and design hybrid control laws to drive the system to safe states [5]. Frazzoli [11] suggested a maneuver automaton, which uses a number of feasible system trajectories to represent the building blocks of the motion plan of the aircraft, and a trajectory based (based on maneuver regulation principle) control system which asymptotically regulates the actual trajectory to the trajectory generated by maneuver automaton. However, motion plans and controllable trajectories are restricted to the library of the maneuver automaton. Such libraries can be built by using interpolation between feasible trajectories [6]. Feron [33] extended this system for online planning of feasible trajectories in partially unknown environments by using receding horizon iterations. Description of aircraft dynamics from hybrid system point of view has been studied previously in [11, 12, 27]. These works have been successful in using the advantages of hybrid system methodology in control of both single and multiple aircrafts. However, these approaches did not include the full flight envelope dynamics of the aircraft. Specifically, both mode selection and controller design is strictly based on selected maneuvers; therefore controllability is limited [11, 12, 27] to these predefined trajectories. In our work, we make use of parameterized sub maneuvers which builds up complex maneuver sequences. We show that it is possible to cover almost any arbitrary maneuver and the entire flight envelope by this approach.

In this section we shall detail the maneuver planning layer. As it is indicated in Fig. 2 this layer is below the path planning layer, in the sense that it receives path-flight trajectory information from PP layer and shapes the trajectory by adding aircrafts attitude angular rates time history while checking the dynamic feasibility of these maneuvers. Maneuver planning takes advantage of multimodal control framework described in [22] and [35].

Multi Modal control framework basically consists of decomposition of the arbitrary maneuvers into set of maneuver modes and associated maneuver parameters. The main aim of the work was the help to reduce complexity of the both planning and control part. Complexity of maneuver planning part has been reduced by reducing the dimension of the problem (modal sequence has strictly lower dimension than state space description) and control part was relaxed by designing specific controllers for each mode and switch between them in order track maneuver mode sequence instead of designing a single controller for maneuver tracking over full flight envelope. In this paper we shall only focus on maneuver planning part, discussion of the switched control layer can be found on [34] and [35].

In this section, we will first briefly review the multi modal control framework. Then, step by step maneuver generation algorithm is explained.

### 3.1 Brief Review of Multi Modal Control Framework

Basically, main idea is to divide an arbitrary flight maneuver into smaller maneuver segments (called maneuver modes) and associated maneuver parameters (called modal inputs). If the maneuver modes are found properly, one can describe any

maneuver by giving the maneuver mode sequence. This idea makes use of the fact that, 12 states of the conventional aircraft are not independent during all maneuvers and one does not need to give all the state trajectory of the aircraft to define a maneuver.

Complete list of modes and their modal inputs along with state constraints on each mode was given in [35]. We review this table (Table 1) below since the same modes will be used during design of the algorithms. Note that for 6 DOF flight state space variables are chosen as:

$$X = [V_T \alpha \beta \varphi \theta \psi \ P \ Q \ R \ n_p \ e_p \ h]^T \quad (5)$$

Where,  $V_T$  is the total speed,  $\alpha$  and  $\beta$  are aerodynamic angles, angle of attack and sideslip angle respectively.  $\Phi = [\varphi \theta \psi]^T$  is 3-2-1 Euler angle set(which are replaced with Quaternions during simulations).  $\omega = [P \ Q \ R]^T$  is the angular velocity vector in the body axes, and  $\rho = [n_p \ e_p \ h]^T$  is the set of Cartesian coordinates. Flight path Euler angles (or wind axis angles) are denoted with  $\Phi_w$ . Note that, safety mode on Table 1 is an artificial mode which serves as an emergency break for the control framework, in the case that aircraft goes out of domains of a particular mode or becomes unstable, it recovers the aircraft by setting it back to level flight.

So via Table 1 state trajectory  $X(t)$  is replaced by the triplet  $(q_i, \sigma_i, \tau_i)$ ,  $i = 1, 2, \dots, N$ , where  $q_i$  is the  $i^{th}$  maneuver mode,  $\sigma_i$  is the set of modal input values associated with  $i^{th}$  mode, and  $\tau_i$  is the time duration of the  $i^{th}$  mode.  $N$  is the number of maneuver modes in total. This triplet is abbreviated as simply “modal sequence”. Since this approach is of lower dimension than state space description, it reduces the complexity of motion planning problems.

Another advantage of maneuver decomposition methodology, other than reduction of the order of the problem, is; it gives opportunity to design specific controllers for each mode of the system. This task is very natural to the system, because each set of modal inputs also serve as a reference output profile for a tracking controller. It is also obvious that if a successful (and possibly nonlinear controller due to coupled nonlinear dynamics of agile maneuvers) each mode is designed, one can gain control over full flight maneuver sequence by switching the controllers. For the assignment of such a switched controller family see [35], and for design of an actual system based on Higher Order Sliding Modes see [34].

**Table 1** Flight modes and modal inputs

	Mode	State constraints	Modal inputs
$q_0$	Level flight	$\dot{h} = 0, (\dot{\varphi}, \dot{\theta}, \dot{\psi}) = 0$	$V_T, \alpha$
$q_1$	Climb/descent	$(\dot{\varphi}, \dot{\theta}, \dot{\psi}) = 0$	$V_T, (\dot{h}, \theta_w)$
$q_2$	Roll	$(\dot{\theta}, \dot{\psi}) = 0$	$V_T, \int P_w dt$
$q_3$	Longitudinal placeLoop	$(\dot{\varphi}, \dot{\psi}) = 0$	$(V_T, r_{loop}), \dot{\theta}$
$q_4$	Lateral placeLoop	$\dot{h} = 0, (\dot{\varphi}, \dot{\theta}) = 0$	$(V_T, r_{loop}), \dot{\psi}$
$q_5$	3D Mode	{}	$V_T, P, Q, R / V_T, \varphi_w, \theta_w, \psi_w$
$q_6$	Safety	{}	{0, 1}

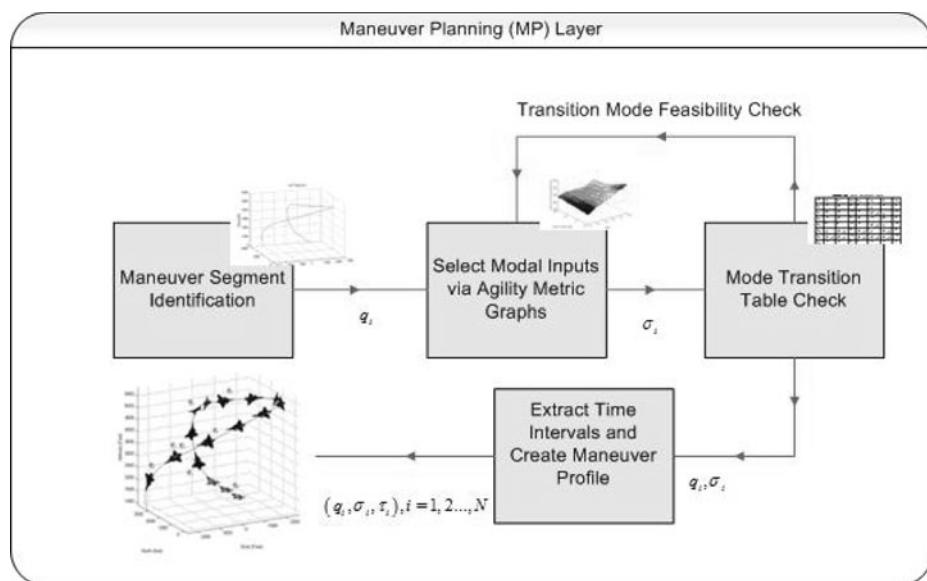
However to ensure that controllers are capable of tracking the maneuver, one must guarantee that maneuvers are feasible in the sense that they are executable by a piloted system, thus satisfying the saturation envelopes. An additional criterion for switching stability is the smooth connection of each mode to another in terms of kinematic parameters, if they are not; discontinuous jumps in output profiles while switching the control system can result in degradation of tracking performance or even instability.

In the next subsection, structure of the maneuver generation algorithm based on the multi modal control framework will be given.

### 3.2 Maneuver Generation Algorithm

Main aim of this section is to develop a maneuver planning algorithm, which extracts the mode sequence from given flight trajectory and derive modal inputs (maneuver parameters) for each mode based on the feasibility constraints. General structure of the algorithm is shown on Fig. 8.

In the Fig. 8, each block represents a part of the algorithm directed for a specific task. After receiving flight path with velocity history from path planning layer, segment identification part decomposes the flight path into a sequence of maneuver modes. Next feasible modal input for each mode is determined by the help of agility metric graphs. Mode Transition table checks if every two adjacent mode is compatible with each other, if not, transition modes are placed between each mode for sequential feasibility, then modal inputs for these transition modes are found similar to previous step. Finally, a time interval for each mode is determined and maneuver profile is generated.



**Fig. 8** Overview of the maneuver planning algorithm

### 3.2.1 Maneuver Segment Identification

Initially, path planning algorithm provides the flight trajectory history,  $n_p(t)$ ,  $e_p(t)$ ,  $h(t)$ . From this data it is easy to recover by the velocity variables in wind axes via Eq. 6 [31]

$$\begin{aligned} \psi_w(t) &= \tan^{-1}\left(\frac{\dot{e}_p}{\dot{n}_p}\right) \\ \begin{bmatrix} \dot{n}_p \\ \dot{e}_p \\ h \end{bmatrix} &= V_T \begin{bmatrix} \cos \theta_w \cos \psi_w \\ \cos \theta_w \sin \psi_w \\ \sin \theta_w \end{bmatrix} \Rightarrow \theta_w(t) = \tan^{-1}\left(\frac{\dot{h}}{\dot{e}_p} \sin \psi_w\right) \\ V_T(t) &= \sqrt{\dot{n}_p^2 + \dot{e}_p^2 + \dot{h}^2} \end{aligned} \quad (6)$$

Next flight trajectory is divided into waypoints, and by comparing the velocity variables between each waypoint, maneuver modes listed on Table 1 can be determined via Table 2.

On the Table 2. “C” means constant and “T” means time varying. Table is self-explanatory; for example in level flight, flight path angle (or wind axis pitch angle) must be zero so that aircrafts altitude doesn’t change, while heading can take any value as long as it doesn’t vary with time (i.e. zero derivative). This straightforward logic is applicable to every mode on the table.

Note that, at this point it is not possible to recover roll mode from given trajectory, since B-Splines (or any curve in space) do not carry this information. In the methodology, roll mode is counted as a transition mode between maneuver segments and inserted by maneuver planning layer. How the roll mode is inserted into maneuver profile is explained in the subsection: ‘Satisfying the Sequential Constraints’.

Since mode labeling action only depends on wind Axes Euler Angles, it is possible to recover the mode sequence and velocity on each mode from given flight trajectory. However complete modal sequence is incomplete because modal input sequence cannot be recovered without attitude history. In the next step each mode will be analyzed via agility metric graphs, and modal inputs will be recovered from flight equations or agility metric graphs.

### 3.2.2 Selection of Modal Inputs

Since velocity is already given by path planner, only remaining modal inputs to be obtained are angular velocities which rely heavily on information of angle of attack. It is obvious that angle of attack cannot be determined alone from given data, so we have to obtain it from some other methods. Selection of angle of attack is a very critical part of creating agile and feasible path, because larger values of angle of

**Table 2** Wind axes euler angles identification table

	$\theta_w$	$\dot{\theta}_w$	$\psi_w$	$\dot{\psi}_w$
Level flight	0	0	C	0
Climb / descent	C	0	C	0
Roll	C	0	C	0
Lon. loop	T	T	C	0
Lat. loop	C	0	T	T
3D Mode	T	T	T	T

attack may cause stall whereas smaller values can result in aerodynamic inefficiency or event saturation of true inputs of the aircraft (control surfaces and throttle). Therefore we have to consult the flight and saturation envelope of the aircraft for a healthy selection of agility metrics.

One way to combine flight envelope, actuator saturation envelope and aggressiveness properties is to use agility metrics. Agility metrics were initially developed for comparing agility characteristics of fighter aircrafts, because classical metrics (such as thrust to weight ratio) were incapable of showing the true agility potential of these aircrafts. Agility metrics are usually given in terms of aircraft states or a time for a specific task (such as time to go through 90 degrees of roll angle), then these metrics are plotted against velocity or angle of attack for various aircrafts.

The strategy for this section consists of specifying an agility metric for each mode (such that associated metric is closely related to the dominant states of each mode) then evaluating the metric from nonlinear flight model simulations for various angles of attack and Mach number (thus building a library of agility metrics for feasible velocity-angle of attack intervals). Then for each mode due to fact that velocity (thus the Mach number) is specified from path planning, is it possible to select an angle of attack from feasible interval (to speed up the process it is selected randomly, it may also be possible to optimize it for some cases, but optimization is not primary objective, because it will decelerate the process and complicate its real time implementability).

In this study a 6 DOF high fidelity nonlinear F-16 model is used for simulations [9]. Selected agility metrics were take from various NASA reports [8, 24, 37]. Selected agility metric for multi modal control frameworks are;

*Level and Climbing Flight* For level and climbing/diving flight total speed and acceleration capability is the most important parameter. Maximum and minimum achievable speed depends heavily on available power and thrust. Selected agility metric is power onset/loss parameter which can be written as:

$$\dot{P}_s = \frac{d}{dt} \left( \frac{V_T (T - D)}{W} \right) \quad (7)$$

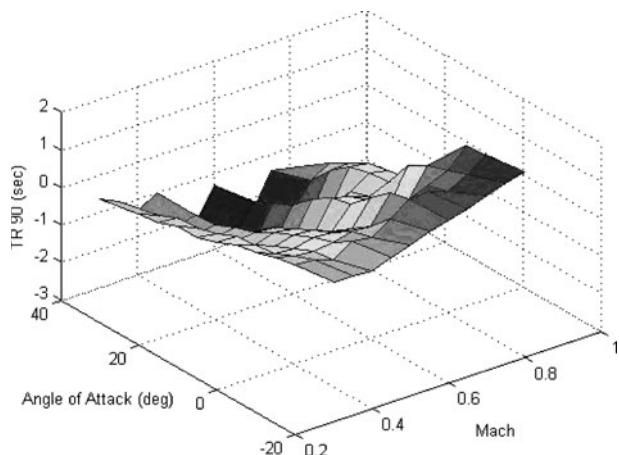
Where T is thrust, D is drag and W is weight. This agility metric quantifies maximum thrust and drag of the aircraft, which determines the total speed and acceleration capability.

*Roll Mode* For rolling motion either average roll rate or time to go through certain roll displacement can be used. Second one is more convenient as it gives transient performance more clearly. For specific angle 90 degrees can be used, because most of the rolling maneuvers consists of rolling aircraft to side (knife edge), or inverting it (180 degrees). Therefore selected agility metric is

$$TR_{90} \quad (8)$$

which means “Time To Go Through 90 Degrees Roll Angle”. 3D plot of this metric is shown in Fig. 9.

**Fig. 9** Time to capture 90 deg roll angle metric for sea level



**Longitudinal Loop** For pitch up/down motion, very simple and convenient metric is average pitch rate which can be written as

$$Q_{avg} = \frac{\int_{t_1}^{t_2} Q dt}{t_2 - t_1} \quad (9)$$

**Lateral Loop** For turning performance load factor and turning radius is chosen as a predominant factor, by using the simple kinematic equation [31]:

$$r_{loop} = \frac{V_T^2}{g(n^2 - 1)} \quad (10)$$

**3D Mode** In 3D mode both rolling and pitching motion becomes dominant; therefore we seek a metric which can combine these two properties. An appropriate metric is loaded roll which is given by the formula

$$PN = p_w N_{z,w} \quad (11)$$

This is simply the product of roll rate in wind axes and normal acceleration in wind axes. This metric belongs to torsional agility and combines the rolling motion of the aircraft with bending of the flight path.

Once the angle of attack history for each mode is obtained (note that final and initial angle of attack for each mode is selected appropriately for continuity), it is possible to recover all of the modal inputs for each mode shown in Table 1. From the rotation matrices and assumption of  $\beta = 0$ ,  $\psi_w = \psi$  it is possible to recover:

$$R(-\beta, \alpha, 0) \cdot R(\psi_w, \theta_w, \varphi_w) = R(\psi, \theta, \varphi) \quad (12)$$

By setting sideslip angle zero, and comparing two sides of these equations which do not contain the variable, we write the equations in closed form;

$$\begin{aligned} \theta &= f_a(\psi_w, \theta_w, \psi, \alpha) \\ \varphi &= f_b(\psi_w, \theta_w, \psi, \alpha) \end{aligned} \quad (13)$$

After obtaining the body axes Euler angles almost all of the modal inputs for each mode can be obtained. Total velocity is a modal input for every mode is available

from path planner. For level flight and Climb/Descent mode, everything needed is already available on Table 2 (Climbing/Diving rate). In roll mode, only desired roll angle displacement (value of the integral associated with roll mode on Table 1) is needed, which can be obtained from Euler roll angle time history. Loop modes require the body Euler angle rates as modal input, which can be obtained from  $\theta(t)$  and  $\psi(t)$  for Longitudinal Loop and Lateral Loop respectively. Things are a bit more complicated in 3D mode, because it is required to extract the angular body rates from a given 3D trajectory and attitude data. Since all the body Euler angles are available, kinematical equation for Euler angles can be solved inversely to acquire the angular rates, but this is not convenient since equations have singular points and requires manipulating trigonometric equations. More elegant approach would be converting the Euler angles to Quaternions (shown by  $b_i, i = 0, 1, 2, 3$ ), and solve the algebraic, singularity free Quaternion kinematical equation to obtain body angular rates. Equation 14 gives the well known formula for converting Euler angles to Quaternions and Eq. 15 shows the kinematical Quaternion equation which has to be solved inversely in order to obtain the angular rates.

$$B = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \pm \left( C\frac{\phi}{2}C\frac{\theta}{2}C\frac{\psi}{2} + S\frac{\phi}{2}S\frac{\theta}{2}S\frac{\psi}{2} \right) \\ \pm \left( S\frac{\phi}{2}C\frac{\theta}{2}C\frac{\psi}{2} - C\frac{\phi}{2}S\frac{\theta}{2}S\frac{\psi}{2} \right) \\ \pm \left( C\frac{\phi}{2}S\frac{\theta}{2}C\frac{\psi}{2} + S\frac{\phi}{2}C\frac{\theta}{2}S\frac{\psi}{2} \right) \\ \pm \left( C\frac{\phi}{2}C\frac{\theta}{2}S\frac{\psi}{2} - S\frac{\phi}{2}S\frac{\theta}{2}C\frac{\psi}{2} \right) \end{bmatrix} \quad S : \text{Sin}, C : \text{Cos} \quad (14)$$

$$\dot{B} = \begin{bmatrix} \dot{b}_0 \\ \dot{b}_1 \\ \dot{b}_2 \\ \dot{b}_3 \end{bmatrix} = \begin{bmatrix} 0 & -P & -Q & R \\ P & 0 & R & -Q \\ Q & -R & 0 & P \\ R & Q & -P & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (15)$$

### 3.2.3 Satisfying the Sequential Constraints

Multi modal control system is based on the successful execution of one mode after another, but this doesn't mean that mode sequence is arbitrary. We can simply cast the mode compatibility condition as; final states of the first maneuver mode must intersect with the second modes initial conditions [32]. This is valid for kinematical variables, or to be more specific the attitude angles (since geometric path is given continuous from path planning layer). The only problem is associated with pitch angle and roll angle, (for example level flight ends with zero roll angle where as the coordinated turn starts with non-zero roll angle). If we neglect this issue, reference maneuver profiles will have discontinuous attitude histories which could degrade the performance of low level controller and may even cause instability.

Therefore a transition mode (roll mode to change and longitudinal loop mode to change) must be inserted between incompatible modes. To speed up the process a mode transition table (Table 3) was prepared; showing which mode is compatible with each other, and which needs a transition mode.

**Table 3** Mode transition table

$\delta_{ij}$	$q_0$	$q_1$	$q_3$	$q_4$	$q_5$	$q_6$
$q_0$	1	$\theta^*$	1	$\varphi^*$	$\theta^*, \varphi^*$	1
$q_1$	$\theta^*$	$\theta^*$	$\theta^*$	$\theta^*, \varphi^*$	$\theta^*, \varphi^*$	1
$q_3$	1	$\theta^*$	1	$\theta^*, \varphi^*$	$\theta^*, \varphi^*$	1
$q_4$	$\varphi^*$	$\theta^*, \varphi^*$	$\theta^*, \varphi^*$	$\varphi^*$	$\theta^*, \varphi^*$	1
$q_5$	$\theta^*, \varphi^*$	$\theta^*, \varphi$	$\theta^*, \varphi^*$	$\theta^*, \varphi^*$	$\theta^*, \varphi^*$	1
$q_6$	1	0	0	0	0	0

in which 1 means that modes are always compatible and sequential feasibility check is not needed at all.

At this step, mode transitions which satisfy the table are neglected and the modes which require attitude tweaking (in either roll or pitch angle) is checked to be compatible. If they are not, additional translational modes (roll mode and longitudinal loop) are inserted between these modes to connect the attitudes of each mode. Since roll mode cannot be identified from trajectory data it only acts as a transition mode and it is not included in the table. Note that table is symmetrical (except for safety mode which is accessible from every mode but only allows transition to level flight).

This attitude changes are made quickly as possible to not to change shape of the flight trajectory (note that feasibility of modal inputs of transition modes are also checked via agility metric plots which corresponds to the loop in Fig. 8), if these transitions make dramatic changes on the flight trajectory, re-planning of the trajectory may be required to make sure that path avoids the obstacles, but this is a very rare case since most of the time environment is big enough to avoid obstacles during execution of transition modes.

### 3.2.4 Recovering the Feasible Modal Sequence

After checking the mode transition table, and finding the appropriate attitude changes for sequential feasibility, these transition modes are added to original mode sequence and final modal sequence is recovered. This modal sequence is feasible in the sense that it satisfies the envelope and sequential constraints.

---

**Algorithm 4:** Maneuver Generation
 

---

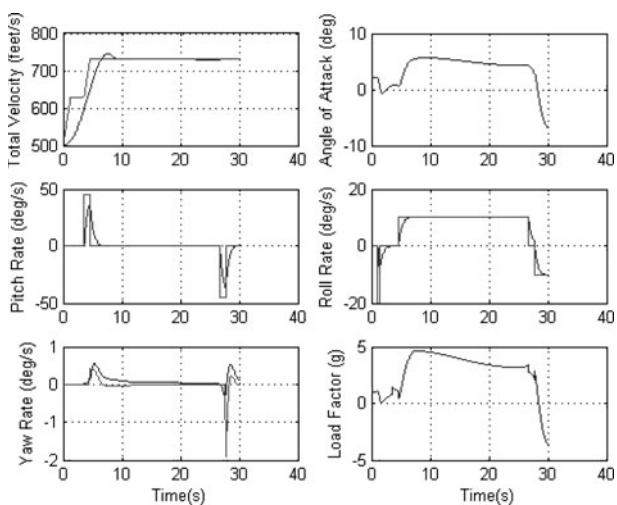
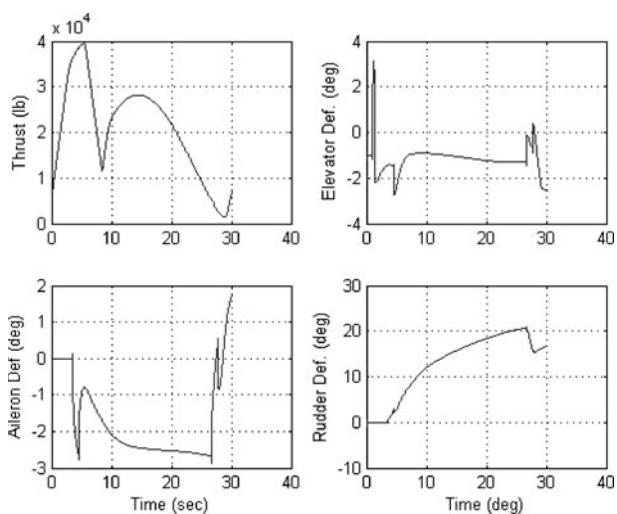
```

input : Flight Trajectory ( $n_p(t), e_p(t), h(t)$ )
output: Feasible Modal Sequence ( $q_i, \sigma_i, \tau_i, i = 1, 2, \dots, N$ )
1 Solve the velocity variables from equation 6 Discretize the flight trajectory, with constant time step
   $\Delta t$  to M waypoints repeat
  2   Label  $q_i$  according to  $(\theta_{w_{j+1}} - \theta_{w_j})$  from table 2.
  3   until  $j=M$ 
  4   repeat
  5     switch Mode Label do
  6       Check Agility Metric Graph, Recover  $\alpha$  Recover Modal inputs  $\sigma_i$  through table 2 and
         equations 14 and 15 Adjust  $\sigma_i$  such that it is compatible with  $\sigma_{i-1}$ 
  7   until  $i=L$ 
  8   repeat
  9     Check Mode Transition Between  $q_{i-1}$  and  $q_i$  via table 3 Insert Transition modes between
         modes if necessary Recover Modal inputs of transition modes via previous step
  10  until  $i=L$ 
  11 Gather Feasible modal sequence by combining identified modes with transition modes
  
```

---

**Table 4** Generated mode sequences

Mode	Label	Time intervals (sec)
$q_0$	Level flight	[0, 0.83]
$q_3$	Longitudinal loop (transition)	[0.83, 1]
$q_1$	Dive	[1, 3.57]
$q_2$	Roll mode (transition)	[3.57, 4.57]
$q_4$	Lateral loop	[4.57, 26.7]
$q_2$	Roll mode (transition)	[26.7, 27.7]
$q_0$	Level flight	[27.7, 30.3]

**Fig. 10** State history and tracking performance**Fig. 11** Time history of true inputs of the aircraft

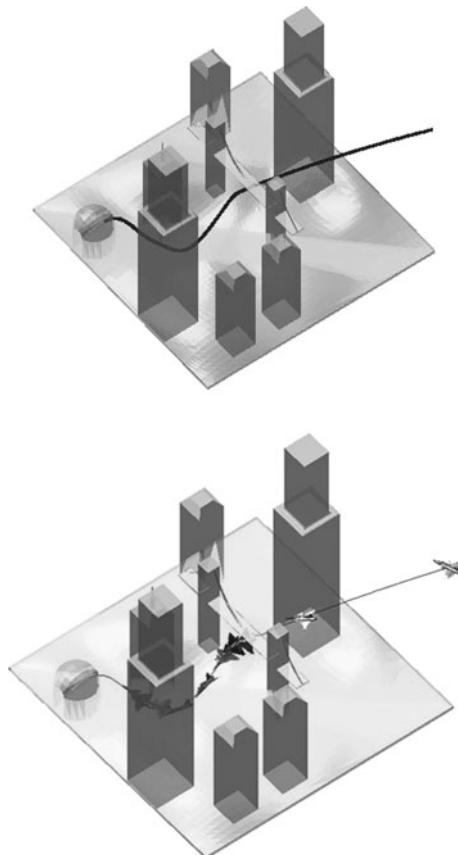
In the next section integration of path and maneuver planning algorithms for a complex environment is shown for an example mission.

#### 4 Simulation Results

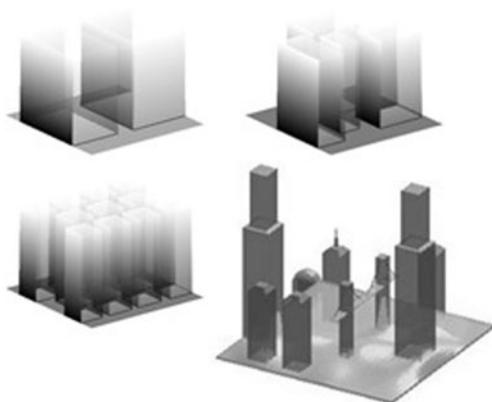
For simulation purposes we consider an example  $200 \times 200 \times 200$  unit cube complex city-like environment. In first part; path planning layer, constructs a 3D flight trajectory which avoids the obstacles while satisfying the velocity and acceleration constraints (by checking the flight envelope). This flight trajectory with velocity is given to maneuver planning layer. Example solution of the Path Planner in the 3D MelCity model environment is seen in Fig. 12.

Maneuver planning algorithm decomposes the path into flight modes in multi modal control framework and derives the feasible modal sequence based on the search of agility metric graphs and mode transition rules. Generated model sequences on the solution of the path planner is seen in Fig. 12 and timetable of the maneuver sequences is seen in Table 4. Three transition modes are placed between the other modes to obtain the attitude continuity.

**Fig. 12** Integration and solutions of the path planner and maneuver planner



**Fig. 13** 3D Test environments for the performance test

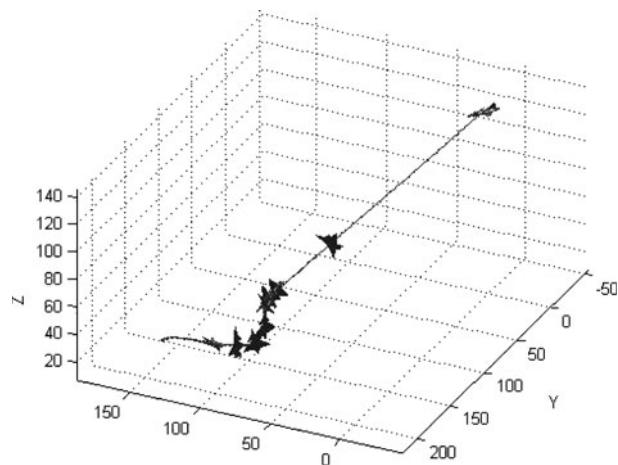


For low level control purposes the switched Higher Order Sliding Mode Control system were used, [34], time histories of important states (with reference tracking performance) and control inputs are shown on Figs. 10 and 11. Judging the magnitude of state and control inputs we conclude that overall architecture has been successful in finding a feasible maneuver sequence (Figs. 12 and 13).

To get a better understanding of what we have gained by this integrated architecture, two extra simulations were done. In the first simulation, path planning step has no flight envelope feasibility check, which results in breakdown in maneuver planning section, because the algorithm fails to compute feasible modal inputs from agility metric graphs and velocity given by path planner is out of range, so no results could be obtained from this simulation. This shows that flight envelope check is a critical part of the integrated architecture.

In second simulation, no maneuver planning is used; trajectory from path planner is directly given to a trajectory tracking control system. Results of this simulation is shown on Fig. 14. This simulation results show that, attitude is unstable on flight path and there are deviations from the trajectory, which has two reasons. First, nonlinear

**Fig. 14** Direct path planning trajectory tracking without maneuver planning



**Table 5** Dynamically feasible path and maneuver sequences construction times (seconds)

		Trajectory planner			Maneuver planner	
		Connectivity path planner	B-Spline path planner	Total	Solution time	Number of mode sequence
Single-passage	avr	0.440	0.206	0.646	0.233	6
Problem	std	0.287	0.011	0.293	0.001	
City-like	avr	0.977	0.326	1.303	0.354	9
Environment	std	0.935	0.254	0.930	0.005	
Mostly-blocked	avr	3.930	2.182	5.837	0.421	13
Environment	std	2.504	3.347	3.912	0.006	
MelCity model	avr	3.306	0.538	3.844	0.322	6
Volume; $2^3 \times$	std	1.528	0.650	1.212s	0.001	

controllers based on trajectory references are likely to become unstable on agile trajectories while multi modal control framework uses stable controllers for each mode [34]. Second reason is the absence of maneuver planning part; since there isn't any feasibility check on the agility metrics and attitude continuity, it is very reasonable to get unstable attitude and deviations from flight path due to saturated inputs, which have been supported by the simulation results.

Overall, these simulation results show that, when they are isolated path and maneuver planning layers have critical defects. They must be integrated together to get a feasible and controllable flight path.

We tested the performance of our method on some environments in varying ratio of obstacle-space. The computational times of the all phases of the algorithm are illustrated in Table 5 for 3D single-narrow-passage problem, city-like environment, mostly-blocked environment and MelCity model environment that has volume  $2^3$  times greater than the others as all seen in Fig. 13. All the experiments were conducted on a 3.00 GHz Intel Pentium(R) 4 processor with 2 Gb memory and the average results are obtained over 50 runs.

On side of the path planner, increasing complexity of the environment, as shown in Table 5, mainly increases computational time of the *connectivity path* that is implemented with a simplified version of RRT. Since repairing part of the algorithm is visited much more in planning complex environments, computational time of the B-Spline based planner phase is also rises. However, this rising rate does not grow exponentially and computational times mostly based on Finding Connectivity Path phase. On side of the maneuver planning, since length of the generated path by the path planner increases when the complexity of the environment is increased, computational time of the maneuver planner phase slightly rises according to environment complexity. The complete solution times suggest that our method will be applicable for real-time implementations as the solution time is favorably comparable to implementation times.

## 5 Conclusion

Trajectory design of an air vehicle in dense and complex environments, while pushing the limits of the vehicle to full performance is a challenging problem in two facets. The first facet is the control system design over the full flight envelope and the

second is the trajectory planning utilizing the full performance of the aircraft. In this work, we try to address the mostly second facet via the generating dynamically feasible trajectory planning and refining of the flight trajectory using the flight modes from which almost any aggressive maneuver can be decomposed. Hence, a real-time implementable two layer planner strategy is implemented for obtaining 3D flight-path generation for an Unmanned Combat Aerial Vehicles in 3D Complex environments. Integrated path planning and maneuver generation system enabled each layer to solve its own reduced order dimensional feasibility problem, thus simplified the problem and improved the real time implement ability.

In Trajectory Planning layer, to solve the motion planning problem of an unmanned combat aerial vehicles, we suggested a two step planner. Initially, simplified version of the RRT planner is used for rapidly exploring the environment with an approximate line segments. The resulting connecting path is converted into flight way points through a line-of-sight segmentation. In the second step, remaining way points are connected with cubic B-Spline curves and these curves are repaired probabilistically to obtain a geometrically (prevents collisions) and dynamically feasible (considers velocity and acceleration constraints) path. In the maneuver planning layer, the flight trajectory are decomposed to sequences of maneuver modes and associated parameters (considers saturation envelope and attitude continuity constraints). Maneuver generation algorithm derives feasible maneuver parameters for each mode and overall sequence by using of mode transition rules and agility metric graphs. Resulting integrated system is tested on simulations for 3D complex environments and it gave satisfactory results to used for real time implementation for UCAVs operating in challenging urban environments.

One of the venues considered for future work is maneuvering in extreme narrow passages in which the aircraft has to roll or tilt to pass through the very narrow passages. In the problems we have examined distance between obstacles are far wider compared to wing span of the aircraft, so we did not include this case. Another venue for future work includes expanding the maneuver modes by adding un-coordinated turns (non-zero sideslip angle). In a framework sense, these extensions will require tighter integration of the path and the maneuver planning layers in the case of uncertain environments. Moreover, extension of the algorithms presented to UAV fleets is another natural application of this work.

## References

1. Bayazit, O.B., Xie, D., Amato, N.M.: Iterative relaxation of constraints: a framework for improving automated motion planning. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 3433–3440 (2005)
2. Bohlin, R., Kavraki, L.E.: A randomized algorithm for robot path planning based on lazy evaluation. *Handbook on Randomized Computing*, pp. 221–249. Kluwer, Dordrecht (2001)
3. Boor, V., Overmars, M.H., van der Stappen, A.F.: The Gaussian sampling strategy for probabilistic roadmap planners. *IEEE Int. Conf. Robot. Autom.* **6** (1999)
4. Brockett, R.W.: Languages for motion description and map making. *Proc. Symp. Appl. Math.* **14**, 181–293 (1990)
5. Sastry, S., Tomlin, C., Pappas, G.J.: Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE Trans. Automat. Contr.* **43** (1998)
6. Dever, C., Mettlera, B., Feron, E., Popovic, J., McConley, M.: Nonlinear trajectory generation for autonomous vehicles via parameterized maneuver classes. *J. Guid. Control Dyn.* **29**, 289–302 (2006)

7. Dyllong, E., Visioli, A.: Planning and real-time modifications of a trajectory using spline techniques. *Robotica*, **21**(5), 475–482 (2003)
8. Murphy, P.C., et al.: Fighter agility metrics. Candidate Control Design Metrics for an Agile Fighter (1991)
9. Nguyen, L.T., et al.: Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. NASA Technical Paper 1538 (1979)
10. Fainekos, G., Gazit, H.K., Pappas, G.J.: Hybrid controllers for path planning: a temporal logic approach. In: IEEE Conference on Decision and Control (2005)
11. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. *AIAA J. Guid. Control* **25**(1), 116–129 (2002)
12. Ghosh, R., Tomlin, C.: Nonlinear inverse dynamic control for mode-based flight. In: Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit (2000)
13. Hsu, D.: Randomized single-query motion planning in expansive spaces, p. 134. PhD Thesis (2000)
14. Hsu, D., Jiang, T., Reif, J., Sun, Z.: The bridge test for sampling narrow passages with probabilistic roadmap planners. In: IEEE International Conference on Robotics & Automation (2003)
15. Hsu, D., Kavraki, L.E., Latombe, J.-C., Motwani, R., Sorkin, S.: On finding narrow passages with probabilistic roadmap planners. In: International Workshop on Algorithmic Foundations of Robotics, pp. 141–153 (1998)
16. Hsu, D., Kindel, R., Latombe, J.-C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. *Int. J. Rob. Res.* **21**(2), 233–255 (2002)
17. Hsu, D., Latombe, J.-C., Motwani, R.: Path planning in expansive configuration spaces. *Int. J. Comput. Geom. Appl.* **4**, 495–512 (1999)
18. Kavraki, L., Svestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **12**(4), 566–580 (1996)
19. Kindel, R., Hsu, D., Robert, J.C., Latombe, S.: Randomized kinodynamic motion planning with moving obstacles. *Int. J. Rob. Res.* **21**(3), 233–255 (2000)
20. Komoriya, K., Tanie, K.: Trajectory design and control of a wheel-type mobile robot using b-spline curve. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems '89. The Autonomous Mobile Robots and its Applications. IROS '89. Proceedings, pp. 398–405 (1989)
21. Koyuncu, E., Inalhan, G.: A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems 2008. IROS 2008, pp. 815–821 (2008)
22. Koyuncu, E., Ure, N.K., Inalhan, G.: A probabilistic algorithm for mode based motion planning of agile unmanned air vehicles in complex environments. Int. Federation of Automatic Control (IFAC'08) World Congress (2008)
23. LaValle, S., Kuffner, J.: Randomized kinodynamic planning. In: 1999 IEEE International Conference on Robotics and Automation. Proceedings, vol. 1, pp. 473–479 (1999)
24. Liefer, R.K.: Fighter agility metrics. NASA Technical Paper Report No: AD-A22447 (1990)
25. Munoz, V., Ollero, A., Prado, M., Simon, A.: Mobile robot trajectory planning with dynamic and kinematic constraints. In: 1994 IEEE International Conference on Robotics and Automation. Proceedings, vol. 4, pp. 2802–2807 (1994)
26. Nikolos, I.K., Valavanis, K.P., Tsourveloudis, N.C., Kostaras, A.N.: Evolutionary algorithm based offline/online path planner for uav navigation. *IEEE Trans. Syst. Man Cybern., Part B* **33**(6), 898–912 (2003)
27. Oishi, M., Tomlin, C.: Nonlinear control of a vstol aircraft. In: The Proceedings of the 38th IEEE Conference on Decision and Control (1999)
28. Paulos, E.: On-line collision avoidance for multiple robots using b-splines. University of California Berkeley Computer Science Division (EECS) Technical Report, (Report No. UCB//CSD-98-977) (1998)
29. Piegl, L.A., Tiller, W.: The NURBS Book. Springer, New York (1997)
30. Song, G., Amato, N.: Randomized motion planning for car-like robots with c-prm. In: 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Proceedings, vol. 1, pp. 37–42 (2001)
31. Stevens, B.L., Lewis, F.L.: Aircraft Simulation and Control. Wiley, New York (2002)
32. Pappas, G.J., Koo, T.J., Sastry, S.: Modal control of systems with constraints. In: Proceedings of the 40th IEEE Conference Decision and Control, pp. 2075–2080 (2001)
33. Feron, E., Schouwenaars, T., How, J.: Receding horizon path planning with implicit safety guarantees. In: American Control Conference (2004)

34. Ure, N.K., Inalhan, G.: Design of higher order sliding mode control laws for multi modal agile maneuvering ucavs. In: 2nd Int. Symposium on Systems and Controls in Aerospace (2008)
35. Ure, N.K., Inalhan, G.: Design of a multi modal control framework for agile maneuvering ucavs. In: IEEE Aerospace Conference (2009)
36. Krishnaprasad, P.S., Manikonda, V., Hendler, J.: Languages, behaviors, hybrid architectures and motion control. Mathematical Control Theory (1998)
37. Valasek, J., Downing, D.R.: An investigation of fighter aircraft agility. NASA Technical Paper 588 (1993)
38. Vazquez, G.B., Sossa, A.H., Diaz de Leon, S.J.L.: Auto guided vehicle control using expanded time b-splines. In: IEEE International Conference on Systems, Man, and Cybernetics, Humans, Information and Technology, vol. 3, pp. 2786–2791 (1994)

# A Cost Effective Tracking System for Small Unmanned Aerial Systems

Michail Kontitsis · Kimon Valavanis

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 14 November 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** In this work we address the problem of object tracking in a largely unknown dynamic environment under the additional constraint of real-time operation and limited computational power. The main design directives remain that of real time execution and low price, high availability components. It is in a sense an investigation for the minimum required hardware and algorithmic complexity to accomplish the desired tasks. We present a system that is based on simple techniques such as template matching adapted for use in a dynamically changing environment. After development, the system was evaluated as to its suitability in a traffic monitoring application where it demonstrated adequate performance.

**Keywords** Unmanned systems · Vision · Tracking · Uncalibrated camera · VTOL

## 1 Introduction

In recent years unmanned aircraft systems (UAS) have been successfully used in a wide variety of applications. Their value as surveillance platforms has been proven repeatedly in both military and civilian domains. As substitutes to human inhabited aircraft, they fulfill missions that are dull, dirty and dangerous [1]. Representative examples of successful use of UAS are in areas including battlefield assessment, reconnaissance, port security, wildlife protection, wildfire detection, search and rescue

---

M. Kontitsis (✉) · K. Valavanis (✉)  
Department of Electrical and Computer Engineering,  
School of Engineering and Computer Science,  
University of Denver, Denver, CO, USA

M. Kontitsis  
e-mail: mkontits@mail.usf.edu

K. Valavanis  
e-mail: Kimon.Valavanis@du.edu

missions, border security and patrol, resource exploration and oil spill detection, to name just a few. The main common component among all those diverse applications is that they are practically variations of remote sensing and surveillance missions.

The reliance of almost every Unmanned Aerial Vehicle (UAV) application on the ability to sense, detect and track objects from a distance has motivated this paper, attempting to further investigate this issue. In particular, among the various types of UAS, small scale unmanned rotorcraft or Vertically Take-off and Landing (VTOL) vehicles have been chosen to serve as the sensor carrier platforms because of their operational flexibility. Having the advantage of being able to operate from almost anywhere, since they require little to none preexisting infrastructure, outweighs their deficit of speed and endurance when compared to their fixed wing counterparts. Their ability to hover and fly in relatively confined spaces makes them almost ideal for deployment at low altitude and in urban settings in which the majority of fixed wing platforms would be challenged to operate. Therefore, and although reported research findings are general enough, the focus of the paper is on designing and implementing an object tracking system for a small unmanned custom made VTOL vehicle.

To accomplish the aforementioned tasks autonomously, any UAS must be equipped with the appropriate sensors to collect data and have enough on-board processing power for data interpretation and decision making. It must also employ a collection of algorithms capable of dealing with a variety of tasks.

Cameras have been used as part of the UAS's sensor suite primarily as data collection equipment rather than navigational aids. Their function usually is to passively sense the environment for the presence of a specifically defined object of interest, record and transmit visual data back to a ground station for evaluation. As previously stated, an essential ability of an autonomous aerial vehicle is that of recognizing and tracking objects of interest, thus, keeping them within the field of view of the camera while recording their trajectory. This enhances the utility of the unmanned vehicle and facilitates the work of the ground control personnel. It allows the UAV to be used as a surveillance tool that expands the covered area without requiring constant attention. However, tracking arbitrary objects from an overflying moving platform in an uncontrolled environment can be extremely challenging given the variability of the parameters that influence the process. In an outdoors environment varying lighting conditions, unstructured clutter, motion, random occlusions and visual obstructions must be dealt with by the detection and tracking algorithms. A very important design directive for an autonomous UAV is the requirement of real-time operation. All the tasks must be completed as fast as possible. In the worst case, the computation time of the decision making components must not exceed the 33 ms barrier that is considered to denote real-time performance. An additional constraint is imposed on the algorithm by the carrying platform itself. Small aerial vehicles set a bound on the electrical power that can be carried along, which in turn limits the available processing power. With limited processing power at hand, the complexity of the algorithms becomes an important factor. Between algorithms that accomplish the same task, the one with low complexity is always desirable. In this case, it is crucial that the selected algorithm be able to run in real-time on less than state of the art equipment.

Another line of distinction exists between systems that are designed so that the processing takes place on-the-ground station and the ones that use an on-board

computer. Obviously the former are not affected by any payload limitations therefore allowing for more powerful computers to be used.

In this paper we address the problem of object tracking in a largely unknown dynamic environment under the additional constraint of real-time operation and limited computational power.

## 2 Method of Solution

To address the tracking problem a simple template matching algorithm based on a similarity measure such as the sum of absolute differences, was implemented. The template is being continuously updated to maintain its relevancy throughout the time period that the object it describes is being tracked. The updated template at any time  $k$  is a linear combination of the best matching image patch and the template at time  $k - 1$ . Finally the tracking system has been designed so that it can concurrently accept input from both a human operator and an automated source such as another program.

## 3 Contributions

The paper's contribution to the area of vision systems for unmanned aerial systems is the design and implementation of a cost effective system capable of performing object tracking in real-time that:

- Requires minimal information about the dynamic environment in which it operates;
- Uses a single uncalibrated, not stabilized camera;
- Tracks multiple objects without requiring a-priori knowledge of or using any assumptions about their trajectories;
- Does not require an IMU.

The result is a system that can be assembled by commercially available hardware and can be configured to perform surveillance without calibration of the camera or detailed knowledge of the operating environment. It becomes evident that the use of an uncalibrated, not stabilized camera makes the problem very challenging and to some extend limits the accuracy of obtained results. However, this is one major point addressed in this work: even with an uncalibrated, unstabilized camera, results are sufficient to complete assigned missions.

## 4 Paper Outline

This paper consists of five sections. The first section introduces the work and briefly describes the problem, the method of solution and the contributions. The second section provides a review of related work and some remarks on them. The third section is devoted to the detailed description of the proposed solution and the implemented system. The performance evaluation is presented in the fourth section along with a description of the actual scenarios where the system was deployed and

the specific tasks that it carried out. Concluding remarks follow in the fifth section along with future research topics that can enhance the current implementation.

## 5 Related Work

Vision systems, techniques and algorithms suitable for UAVs range in complexity from simple color segmentation to statistical pattern recognition. This literature review considers a publication as being related to this work if the implemented system is specifically designed for use by UAVs. Furthermore, a work is considered directly comparable if the resulting vision system is physically placed on an unmanned VTOL/UAV and has been shown to function under real operating conditions in an outdoors environment.

Published related work and proposed machine vision architectures indicate the use of both “on-board” [2–9] and “on-the-ground” processing setups [10–19]. For on-board vision systems, due to the limited processing power provided by the on-board computer, derived algorithms have the additional constraint to run in real-time, requiring reduction of the computational load sometimes accomplished by processing selected image windows instead of the entire image. Table 1 summarizes

**Table 1** Summary of system characteristics and functionality

Institution	Berkeley University	Georgia Tech	Univ. of South California	COMETS <sup>a</sup> [31]	WITAS <sup>b</sup> [8]	CNRS <sup>c</sup> [32]
Experimental setup	Dynamic observer	X	X	X	X	X
	Dynamic environment				X	X
	Static/man-made environment	X		X		X
	Known landmarks	X	X	X		
	Natural landmarks				X	
	Calibrated cameras				X	
Capabilities	Depth mapping		X	X		
	Object identification	X	X	X	X	
	Object tracking		X	X	X	
Methods used	Optic flow			X	X	X
	Motion estimation	X			X	X
	IMU data					X
	Template matching	X		X	X	

<sup>a</sup> COMETS is a multi-national effort supported by the European Commission

<sup>b</sup> Wallenberg laboratory for research on Information Technology and Autonomous Systems (WITAS)

<sup>c</sup> Centre National de la Recherche Scientifique (CNRS) in France

functionality and capabilities of existing fully operational vision systems, including techniques employed by each one of them.

The problem of object tracking has been studied extensively in computer vision. Although several methods exist that can track objects in a controlled setting, special reference will be made to those of them that have been adapted for use in unmanned aerial systems since it is believed that they relate more closely to the problem at hand. One such example is illustrated in the work of Ludington et al. [20] that presents a method for target tracking using a technique based on particle filters. Each target is described as a four dimensional particle containing the image coordinates and the dimensions of the rectangle that surrounds it. The assumption for the system to operate is that the target moves smoothly between frames and that the frame rate remains sufficiently high.

The motion is modeled as Gaussian random walk and the measurement model relies on color and motion cues. Finally, a neural network is responsible for constructing a performance estimate according to which adaptations are made to the particle filter. Another example is the system described in [21, 22] where features such as rectangles are first extracted using the Douglas-Peucker algorithm [23] and then tracked using a Kalman filter.

Although not explicitly designed for an unmanned vehicle the system presented in [24] is addressing the problem of tracking moving objects in aerial video. It employs two different trackers; one for short and another for long term tracking. The short term tracking is accomplished by first registering successive frames using an affine transformation to correct for background motion and then extracting and matching Kanade–Lucas–Tomasi [25] features between successive image pairs.

The long term tracker relies on model matching to follow a specific pattern through the sequence. The Lucas–Kanade tracker is also utilized by Kanade et al. [26] in conjunction with a motion prediction scheme that relies on Kalman filtering, an image pyramid and two dimensional affine motion models to deal with large motion in image sequences taken from a micro-unmanned aerial vehicle.

Motion tracking using low cost off the shelf components is also investigated in [27] where a fixed wing UAV is relaying data back to a ground station where the processing takes place. The authors use a proprietary vision system which according to their accounts “lost target track on a regular basis”. Tracking salient points is demonstrated in [28] using SIFT features and the RANSAC algorithm where the authors are reporting correct projection in 55.4% to 82.5% of the frames while spending 0.63 to 1.93 s per frame.

Another area where tracking a ground target is important is that of autonomous landing. As shown in [29] the helipad is usually marked by an “H” which is extracted from the images using fixed threshold segmentation and tracked during the landing maneuver by means of second and third order moments of inertia.

## 6 Challenges of Implementing Vision for a VTOL

Helicopters are attractive as unmanned vehicles due to their ability to take off from almost anywhere without the requirement of a runway. Furthermore, their ability to hover makes them ideal for surveillance since they can keep the onboard sensors pointed towards the same area without having to execute elaborate maneuvers. The

price to pay for that flexibility is their lower speed, limited endurance and inherent instability when compared to fixed wing aircraft. Small unmanned helicopters are even more unstable and susceptible to even modest changes in environmental conditions. This instability affects the images that any onboard camera would acquire and requires either the use of stabilization hardware which adds weight or stabilization software, which adds in turn complexity and demands more processing power.

The unmanned aerial vehicles that were used for the purposes of this work are designed to operate outdoors. Such an environment is notoriously difficult for computer vision primarily due to variations in lighting. Furthermore, there is usually limited availability of a-priori knowledge of the environment and certainly no three dimensional map which leaves little room for helpful assumptions to be made.

The low cost requirement and the low payload allow only for a single camera to be carried on-board. A second camera could have been utilized to allow for a stereoscopic system and provide additional data for verification purposes leading to a more robust detection. Its absence can be viewed as an additional design constraint for the vision algorithms.

## 7 Tracking System

The system is required to be able to track multiple objects through time. It is designed in a way that allows the user to designate the objects to be tracked during runtime. The main challenges include trajectory prediction as well as occlusion handling and motion segmentation. The demand for low computational cost, real-time execution, and a minimal object description model still applies.

Although the helicopter is perceived to be stationary when hovering and attempting to track ground bound objects, this is rarely the case. Given the relatively small size of both the Raptor 90 and the Maxxi Joker, even slight variations in the wind's direction or speed can result in an unstable flight profoundly influencing the quality of the acquired images. This translates to relatively high disparities between corresponding objects in subsequent image frames. Furthermore, it makes tracking objects close to the boundaries of the image almost impossible because they may appear and disappear erratically due to the relative motion of the camera with respect to them.

Occlusions present a significant challenge when attempting to track a specific object in a dynamically changing environment. Objects are expected to move almost randomly and therefore occlude each other. The background environment although static can contribute to this problem whenever it includes obstacles comparable in size with the objects of interest. Tree lines for example are such typical obstructions. Also, since the camera is mounted on a moving platform it is possible for any object to become occluded even by remaining stationary. As one might expect, occlusions are greatly reduced in frequency when the optical axis of the camera is perpendicular to the terrain. However hovering directly above the area of interest may not always be feasible or desirable due to safety concerns.

Motion or background segmentation is another challenge due to the nature of the environment that the unmanned vehicle operates in. Typical background extraction techniques such as frame differencing or Kalman filtering do not cope well with rapidly varying scenes. In particular, frame differencing degenerates to a crude edge

detector when applied to a sequence of images acquired by a moving camera. On the other hand, motion estimation algorithms like the ones used in the MPEG standard that were also considered, found to be highly demanding in terms of processing power. However with dedicated hardware that accelerates MPEG encoding this could be a viable choice for motion estimation.

### 7.1 Tracking System Module Overview

Having made a review of the typical problems related to object tracking within the context of small unmanned VTOLs, we now describe the operation of our system along with the modules that constitute it. At first the object to be tracked has to be specified. This information can either come from an object detection system that is automatically searching for a pre-specified object or from a user who activates the tracking system by manually selecting an object in the image. The target selection module then creates a description of the selected object and forwards it to the matching module. The latter will attempt to find a corresponding template in the subsequent image. Once such a match is found the original template that describes the object is updated with information from the most currently acquired frame. Finally the Pan-Tilt controller signals the Pan-Tilt mechanism to move accordingly so that the tracked object is at the center of the image. Briefly stated the tracking system is comprised of:

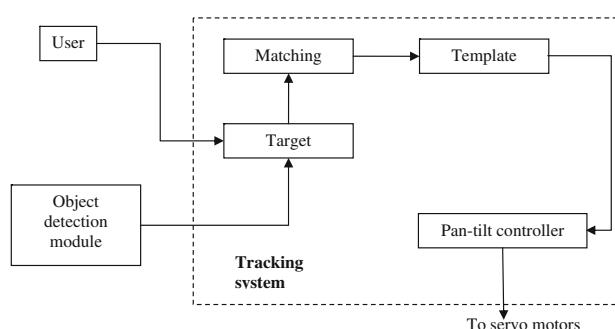
- The target selection module.
- The matching module.
- Template update module.
- Pan-Tilt controller.

A block diagram showing the interconnections between the aforementioned modules can be seen in Fig. 1. The operation of each of the modules at any point in time is described in the following paragraphs and shown in Fig. 2.

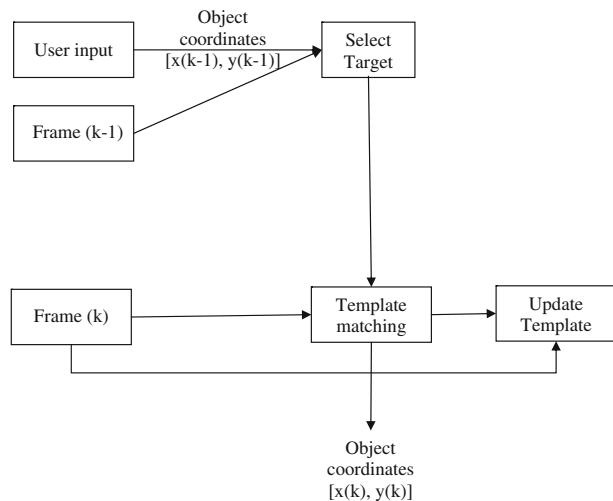
### 7.2 Target Selection Module

This component is responsible for receiving the user's input and creating a description for the object to be tracked. In both cases the input to this module is a series of

**Fig. 1** A block diagram of the tracking system



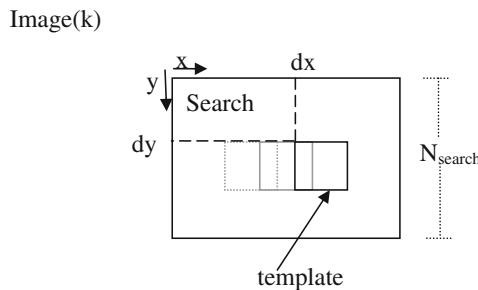
**Fig. 2** Operation of the tracking system at a given time  $k$



coordinates ( $x, y$ ) representing the object's position in the image's column and row coordinate system as seen in Fig. 3. The design choice has been made to employ a  $N_t \times N_t$  area taken around the image point ( $x, y$ ) as a template for matching into subsequent image frames.

### 7.3 Matching

This module attempts to find a match for the template within the current frame. To reduce the search space for this match the search is limited within a  $(N_{\text{search}} \times N_{\text{search}})$  area around the latest previously known position. Assuming continuity of motion for the object under tracking, it is reasonable to expect that it will appear in the next frame relatively close to its current position. The similarity measure for the match



**Fig. 3** The template matching process

is the sum of absolute differences between the template and all the  $N_t \times N_t$  square sub-images within the search space:

$$SAD(d_x, d_y) = \sum |template(x, y) - image_k(x + d_x, y + d_y)| \quad (1)$$

The row and column position that minimize the similarity measure become the output of this module.

$$\arg \min_{d_x, d_y} (SAD(d_x, d_y)) \quad (2)$$

Obviously a trade off exists when selecting the size of the search space. A large value for  $N_{search}$  will allow for a larger disparity between the positions of the object in subsequent frames. The penalty for a larger search space is obviously the extra computational cost which increases with the square of  $N_{search}$ . On the other hand, decreasing the search space may save some computing time but it entails the possibility of not finding a proper match just because the object moved more than  $N_{search}/2$  pixels in any direction. A good compromise was achieved by making a selection based on the maximum expected apparent motion *Max\_disparity*.

For our applications we selected:  $N_{search} = 2 \times Max\_disparity = W/10$ , where  $W$  is the width of the captured image. Our implied assumption is that the apparent motion of the observed object will not be exhibiting inter-frame displacements of more than  $W/20$  pixels. Figure 3 shows the relation between the sliding template, the image and the search space.

#### 7.4 Template Update Module

The typical weakness of a tracking system based on template matching is the fact that with time the template may become irrelevant. Objects are moving and their pose with respect to the camera is almost constantly changing. As a result, the projected two dimensional image of any given object differs significantly within the time span of a few seconds making any attempt for a match with the original template almost impossible.

To mitigate this effect the template is updated at every cycle of the algorithm's execution. Every new captured image, within which a match was found, contributes to the template by introducing information to it thereby forming a new one. The new template is a linear combination of the existing one and the neighborhood of the best match. In most cases the linear combination of the current template and the best matching patch is sufficient to maintain the relevancy of the template without incurring a significant processing power penalty.

If  $Template(k)$  is the template at time  $k$  and  $Match(k)$  the  $N_t \times N_t$  neighborhood around the coordinates of the best match then:

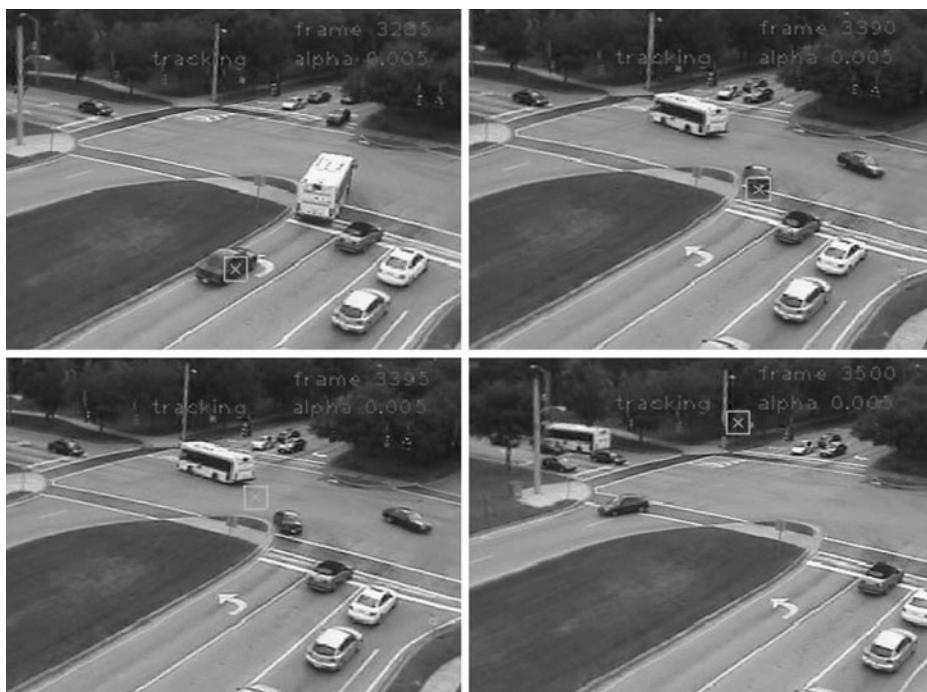
$$Template(k+1) = a Match(k) + (1-a) Template(k), \text{ where } a \in [0, 1].$$

The design decision to be made when calculating this new template is about the amount of new information that will be incorporated versus the amount that will be retained from the “old” template. Apparently there is a trade-off. If one chooses

to retain more of the older template the result will be a slower changing template unable to accommodate pose variations that happen within the time span of a few frames. This, however, will make the template impervious to short duration random illumination variations as well as to the occasional miss-match. On the other hand if the choice is made to aggressively update the template with new information then it has a better chance of remaining relevant and being able to cope even with objects whose pose and appearance vary rapidly. The caveat in this case is that the template becomes susceptible to noise and to the fact that even a single mis-match by the matching module can easily throw-off the tracking system by forcing it to follow the new miss matched object. The balance between the new and old information is controlled by the constant  $a$ . Lower values of  $a$  place more weight on the old template rather than the newly acquired image and higher values of course have the opposite effect. After some experimentation the value  $a=0.025$  was chosen as the one that yielded the best compromise between robustness and adaptation.

#### 7.4.1 Evaluating the Template Update Module

To further investigate the way the values of  $a$  influence the output of the tracking system, a series of experiments were conducted. Each time, the system was instructed to track the same object through the sequence while the value of  $a$  remained constant. This was repeated for values of  $a$  ranging from 0.005 to 0.995 with a step of 0.01. The sequence selected for that purpose is one that contains a car executing a u-turn



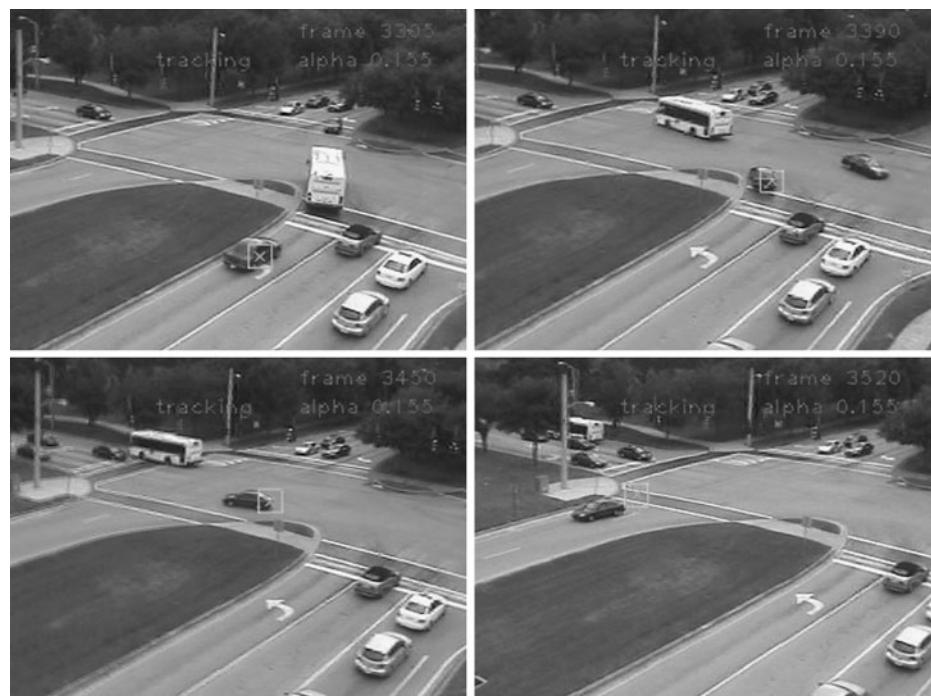
**Fig. 4** Tracking results for  $a = 0.005$

maneuver. The choice was made because that particular video excerpt contains an object that changes its pose relatively fast allowing the opportunity to validate the effectiveness of the way the template is updated to accommodate the changing appearance of the target. However, the template must also retain some of the past information to ensure the identity of the target. During the experiment, it was verified that for low values such as 0.005 the template does not adapt fast enough to maintain the track. Figure 4 shows exactly that. Conversely, values above 0.155 forced out of the template enough past information so that the tracking system abandoned the initial target as shown in Fig. 5.

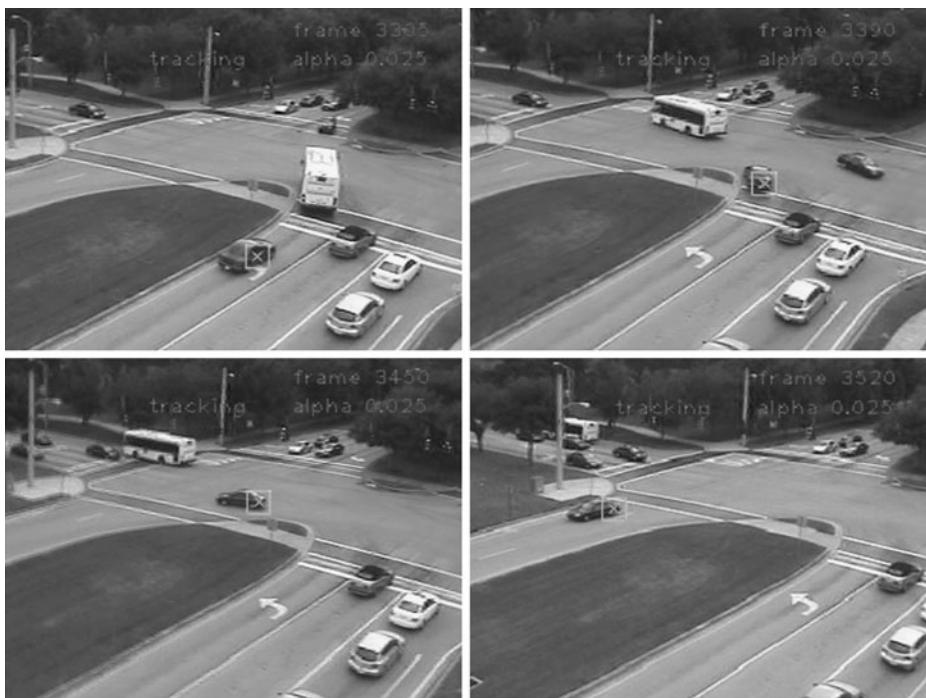
Finally, the value of  $a$  for which the system exhibited the desired behavior was 0.025. As shown in Fig. 6 the vehicle is consistently tracked through the sequence. Having established this trajectory as the ground truth, the root mean square (RMS) error was calculated for the ones produced by each of the 100 different values of  $a$ . Isolating some characteristic values of  $a$  and plotting the error for them yields Fig. 7. One can notice the sharp increases that correspond to the time that the tracking failed.

## 7.5 Pan–Tilt Controller

It is usually desirable, if not required, that the tracked object remains in the camera's field of view (FOV). This task is accomplished by the Pan–Tilt controller, which as the name implies, sends control signals to the servos that adjust the pan and tilt angles

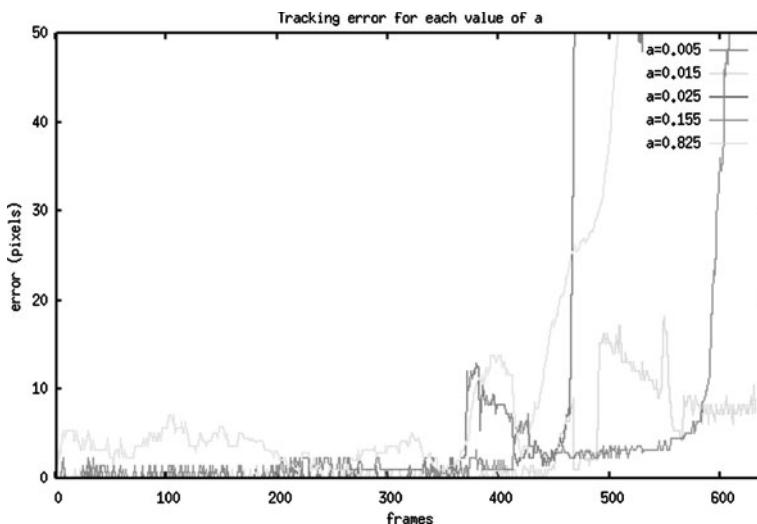


**Fig. 5** Tracking results for  $\alpha = 0.155$



**Fig. 6** Tracking results for  $a = 0.025$

of the camera. The goal is to keep the tracked object approximately at the center of the captured images. For that reason an error vector is calculated between the center of the image and the image point where the object is located. The vertical and



**Fig. 7** Plot showing the RMS error for some characteristic values of  $a$

horizontal components of this error vector are then used to adjust the pan and tilt so that the error is minimized. The control rules for the Pan–Tilt are:

- If  $\text{error}_x < -N$  then pan left by  $5^\circ$
- If  $\text{error}_x > N$  then pan right by  $5^\circ$
- If  $\text{error}_y < -N$  then tilt up by  $5^\circ$
- If  $\text{error}_y > N$  then tilt down by  $5^\circ$

To avoid oscillation and constant corrections the object is kept within a  $N \times N$  window centered around the center of the image.

## 8 Traffic Monitoring

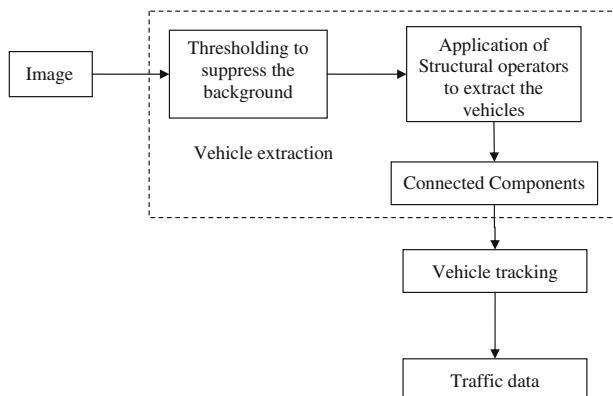
A very interesting application for an unmanned VTOL is that of traffic monitoring. The VTOL is a surprisingly suitable platform since it can hover over a particular traffic node for varying periods of time depending on configuration. A medium sized autonomous helicopter such as the Bergen Observer can hover for a period of 45 min and up to more than an hour and a half if equipped with extra fuel tanks. However the electric powered Maxxi Joker 2 can only provide a 12 to 15 min hover, although this is expected to increase as new, higher density battery technology becomes available. Furthermore, a traffic monitoring system based on autonomous or semi-autonomous VTOLs can be deployed very rapidly in areas that provide little or no infrastructure. Once deployed, it can provide real-time data to operators on the ground or to traffic simulation models which can then provide more accurate predictions for traffic parameters based on more current observations. The operator of the system can also direct the system to track a certain vehicle by manually selecting it with a pointing device (e.g. a mouse). Additionally, such a system can serve as an ad-hoc replacement for already existing traffic monitoring infrastructure in case the latter has failed or has been accidentally destroyed. In emergencies such as a hurricane evacuation or a large scale automotive accident an autonomous VTOL deployed from a first responder's vehicle can provide helpful information.

Briefly stated, the autonomous VTOL provides the capability of a traffic monitoring system without the requirement of extensive infrastructure. It has two modes of operation, one automatically extracting traffic data and another tracking manually selected vehicles.

## 9 Description of the System

As stated above one of the objectives of the system is to extract meaningful real-time data from the video stream captured by the onboard camera on the VTOL. Such data include the total number of vehicles that pass through a given part of the road network, the number of vehicles that follow a certain path and the overall traffic flow. This task can be separated into three distinct steps. Initially the image areas that correspond to vehicles have to be differentiated from the environment. Secondly, the extracted vehicles must be consistently tracked as they traverse the portion of the road network that is being examined. Lastly, the result of the tracking procedure is converted to meaningful traffic measures. A block diagram showing the succession

**Fig. 8** A block diagram representation of the data extraction mode of the traffic monitoring system

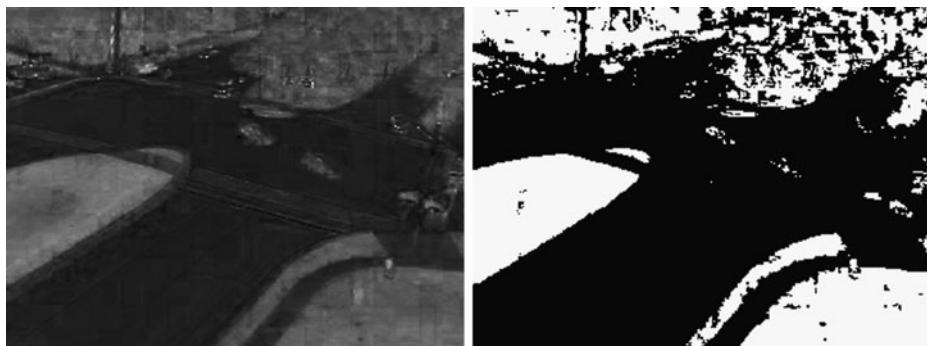


of these steps is shown in Fig. 8. The following paragraphs provide a more detailed description of the whole process.

The first step towards extracting the desired traffic data involves identifying the vehicles. In keeping with the goals of this work, the vehicle extraction process must be simple, computationally cheap and requiring minimal a priori knowledge. The selected method takes advantage of the fact that the areas of the image corresponding to a paved road usually have extremely low saturation values. A simple sufficiently low threshold is then applied to suppress the part of the background that is the road. Following that, a pair of erode/dilate morphological operators are used to further distinguish the blobs that correspond to vehicles. A size filter is employed to avoid some residual areas of the background being categorized as vehicles. In particular, for any formation of pixels to be accepted as a vehicle it has to fit within a bounding rectangle no smaller than  $W/10 \times H/10$  and no larger than  $W/3 \times H/3$ , where  $W$  and  $H$  are the width and height of the image respectively. Its effectiveness is based on the assumption supported by the observation that areas smaller than  $W/10 \times H/10$  usually correspond to noise whereas areas with dimensions larger than  $W/3 \times H/3$  are usually representative of various background formations. The application of such a filter enforces a measure of scale on the problem albeit a reasonable one. Figures 9,



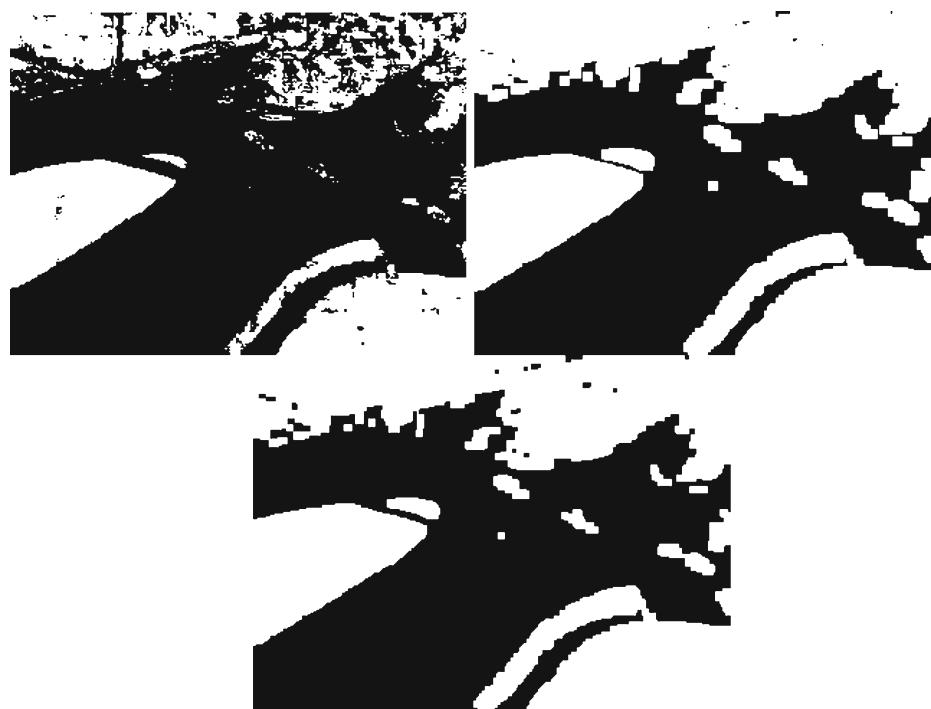
**Fig. 9** The RGB input image in the *left* is converted to HSI. The saturation component is shown in the *right*



**Fig. 10** The application of a threshold on the saturation component (*left*) eliminates most of the pixels belonging to the road (*right*)

10, 11 and 12 depict the image processing steps leading from the input image to the extracted vehicles.

In succession, a set  $track_{k-1}$  is constructed containing the center of gravity of all the regions that pass the size filter at a given time  $k - 1$ . This set containing pairs of  $(i, j)$  image coordinates is presented as input to the tracking module described



**Fig. 11** Applying morphological operators to the binary image (*upper left*). Dilation (*upper right*) then erosion (*lower center*)

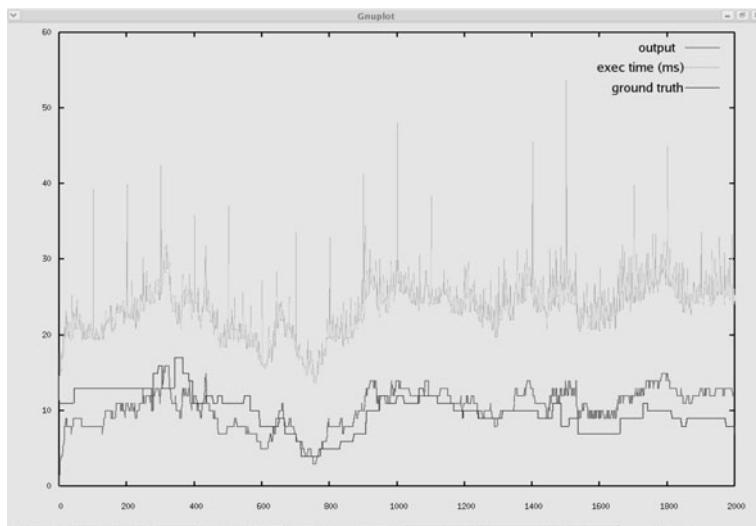


**Fig. 12** Extraction of regions using connected components and a size filter. Notice the rejection of very large and very small blobs

in Section 3. The output of the tracking module forms a set  $track_k$  that holds the respective matching coordinates at the current frame  $k$ . It is worth mentioning that time in this case has been discretized to coincide with the acquisition rate of the camera. So frame  $k$  and time  $k$  are interchangeable.

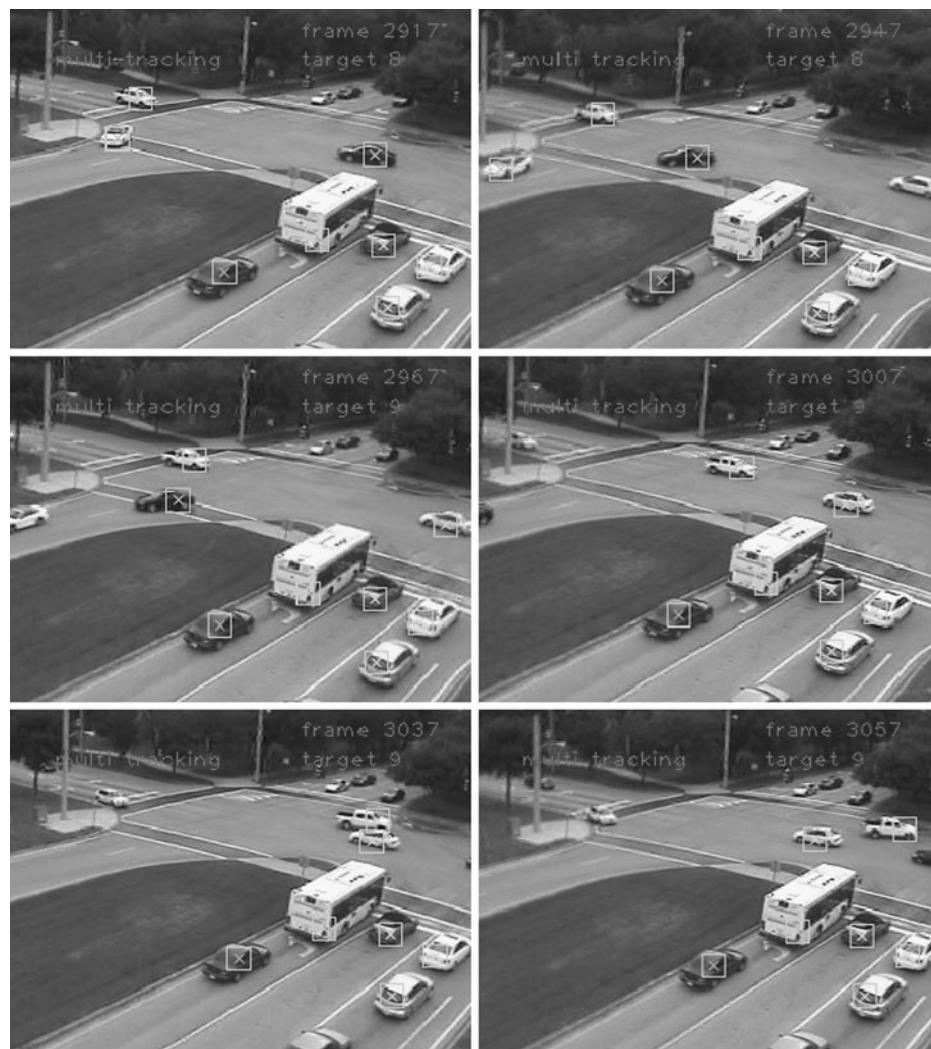
Having completed the vehicle extraction and tracking the remaining task is to calculate the traffic parameters of choice. The number of vehicles currently present on the road is simply equal to number of objects represent in the tracking set. In other words the cardinality of  $track_k$  gives the number of current vehicles. The amount of vehicles over any given period of time can be found by integrating the function of vehicles over time.

Plotting the function of vehicles over time, results in the graph shown in Fig. 13. The red line represents the number of vehicles that the system estimates are currently



**Fig. 13** The output of the traffic load estimator compared to the ground truth

in view, the blue line is the actual number of vehicles present and the green line shows the execution time in milliseconds. On average the system estimated the current traffic with an accuracy of 81%. For comparison, traffic monitoring systems that are utilizing fixed cameras [30] have reported higher rates of 95% to 97%. They benefit from the more consistent background and motion extraction since the sensors remain static and all apparent motion in the image can only be attributed to vehicles. Note that the execution time remains well below 33 ms which signifies real-time operation since it coincides with the capture period of the camera. The occasional spikes correspond to write cycles during which the operating system logged the output of the vision program to the disk. They are not due to executing the core vision code



**Fig. 14** Tracking is maintained despite the unpredictable motion of the VTOL and the parallax inducing motion of the vehicles



**Fig. 15** Tracking of a vehicle making a u-turn

and are not present during normal operation. The computer used in this case was a Pentium 4 at 3 GHz.

The system can also function in another mode of operation that relies on user input for target designation. It allows a certain number of manually selected vehicles to be tracked. The operator has only to point and click on the vehicles to be tracked. The image coordinates are gathered and presented as input to the tracking system described in Section 3. Targets are abandoned by the system as they exit the field of view in order to make the resources of memory and processing power available for reallocation to tracking newly acquired targets. Figure 14 shows that the tracking can be maintained despite the unpredictable motion of the VTOL and the parallax inducing motion of the vehicles. Further resilience to parallax is demonstrated in Fig. 15 where vehicles are shown to be tracked while executing turns and u-turns that change their pose with respect to the camera.

## 10 Conclusion

This paper presented a system for object tracking to be used by unmanned aerial systems. It is based on template matching using the sum of absolute differences as a similarity measure. A template update occurs in every loop of the algorithm by incorporating new information so that the former remains relevant allowing the track

to continue despite changes in the object's appearance. Furthermore it allows for manual entry of targets by the human operator as well as receiving input from other image processing modules. The following paragraphs further discuss the results and contributions of this paper as well as provide some final remarks.

## 11 Discussion

The tracking system developed in this work was able to correctly track on average 81% of the visible vehicles when assigned to the task of traffic monitoring. It was shown capable of coping with significant disturbance, resulting from the nature of the carrying platform and the unstabilized camera, as well as with occasional brief occlusion and change of pose of the tracked object. However, there are specific situations that are beyond the ability of the system to compensate for. A severe disturbance that results in an apparent motion of more than  $W/20$  between successive frames, where  $W$  is the width of the frame, will also result in a lost track. For an image acquisition rate of 30 fps, this apparent motion amounts to 1.5  $W$  pixels/s which is arguably a high enough limit. Reversely, assuming that the camera is perfectly stable, only objects able of covering the largest visible dimension in two thirds of a second will avoid being tracked. Another case where the track can be lost is when an occlusion occurs that covers more than 50% of the template corresponding to the object being tracked. In such a circumstance, within a few repetitions of the template update process the template will lose enough information to make finding the proper match problematic.

This work contributes to the area of vision systems for unmanned aerial systems by proposing a monocular, uncalibrated, not gyro-stabilized design capable of identifying and tracking multiple objects in real-time, without strict assumptions about their trajectories, by relying on minimal information about the environment while forgoing the need for motion compensation usually requiring an inertial measurement unit (IMU). By selecting a monocular, non stabilized system the design avoids the extra cost and weight of a binocular configuration along with that of a gyro-stabilized turret. The fact that the camera does not require calibration for the system to operate, further simplifies the assembly and setup process.

The ability to track multiple objects at a low computational cost allows the design to run in real-time on inexpensive, less powerful, power efficient computing platforms. This is especially critical for relatively small unmanned with limited payload capabilities for which carrying a large energy reserve to support a powerful computer is impossible. Although the prevailing trends in computer design and battery chemistry promise to alleviate this problem by providing more efficient, powerful hardware and higher energy density batteries respectively, the desire for the minimalist's approach in the design of a vision system will most likely continue to be relevant. This is especially true for the class of unmanned aerial systems that are even smaller than the Maxxi Joker 2 platform that was used in this work.

By keeping the requirement of knowledge about the environment to a minimum, the design is not explicitly required to perform elaborate background extraction thus further reducing the computational burden. No a priori known landmarks are used to facilitate detection and tracking and minimal assumptions, such as the one used in the traffic monitoring scenario about the tarmac being gray, are made. This allows for

a fairly versatile system capable of performing in different environments with only a few, if any, modifications.

Future improvements should include an adaptive search space for the tracking system based on probabilistic metrics. Also, the availability of relatively inexpensive, lightweight cameras with electronically controlled optics offers the opportunity to further explore the problem from the perspective of active vision. Furthermore, varying optics can be employed by an attention focusing mechanism to extract more information in cases where disambiguation between objects is required. This zooming capability can also be of significant help to the human operator who can employ it to further investigate an object of interest without interfering with the flight of the VTOL.

In summary, this work produced a design suitable for tracking multiple objects in real-time. It demonstrated that it is possible to endow relatively small, inexpensive VTOLs with capabilities usually reserved for more expensive, higher end unmanned systems.

**Acknowledgement** This work has been partially supported by NSF Grant, IIP-0856311 (DU grant number 36563).

## References

1. Unmanned systems roadmap: 2007–2032. Office of the Secretary of Defense (2007)
2. Vision hardware. University of Southern California. [http://www robotics.usc.edu/~avatar/vision\\_hw.htm](http://www robotics.usc.edu/~avatar/vision_hw.htm) (2004). Accessed 27 Jan 2005
3. Mejias, L., Saripalli, S., Sukhatme, G., Cervera, P.: Detection and tracking of external features in an urban environment using an autonomous helicopter. In: IEEE International Conference on Robotics and Automation (2005)
4. Hrabar, S., Sukhatme, G.S.: A comparison of two camera configurations for optic-flow based navigation of a UAV through urban canyons. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2673–2680 (2004)
5. Sattigeri, R., Calise, A.J.: An adaptive approach to vision based information control. In: Guidance, Navigation and Control Conference, no. AIAA-2003-5727, Austin, TX (2003)
6. Enhanced Vision System (EVS) overview, CMC electronics. [www.cmcelectronics.ca/En\\_ProdServ/Commav/commav\\_evs\\_overview\\_en.html](http://www.cmcelectronics.ca/En_ProdServ/Commav/commav_evs_overview_en.html) (2001). Accessed 22 Jan 2005
7. Groven, J., Holk, E., Humbert, C., Krall, J., Schue, D.: Rose-Hulman institute of technology autonomous helicopter for the 2004 international aerial robotics competition (2004)
8. Nordberg, K., Doherty, P., Farneback, G., Forssen, P.-E., Granlund, G., Moe, A., Wiklund, J.: Vision for a UAV helicopter. In: 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems—IROS 2002. Proc. Workshop WS6 Aerial Robotics, pp. 29–34, Lausanne (2002)
9. Anderson, J.D., Lee, D.-J., Edwards, B., Archibald, J., Greco, C.R.: Real-time feature tracking on an embedded vision sensor for small vision-guided unmanned vehicles, pp. 55–60. <http://ieeexplore.ieee.org/iel5/4269829/4269830/04269882.pdf> (2007). Accessed 20 July 2007
10. Sastry, V.: Vision based detection of autonomous vehicles for pursuit evasion games. In: 15th Triennial World Congress. Barcelona, Spain (2002)
11. Hrabar, S., Sukhatme, G.S.: Omnidirectional vision for an autonomous helicopter. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3602–3609 (2003)
12. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2799–2804 (2002)
13. Woodley, B.R., Jones, H.L., LeMaster, E.A., Frew, E.W., Rock, S.M.: Carrier phase GPS and computer vision for control of an autonomous helicopter. In: ION GPS-96, Kansas City, Missouri (1996)
14. Debitetto, J.: Modeling and simulation for small autonomous helicopter development. In: AIAA-1997-3511 Modeling and Simulation Technologies Conference, New Orleans, LA (1997)

15. Woodley, B., Jones, H., Frew, E., LeMaster, E., Rock, S.: A contestant in the 1997 international aerial robotics competition. <http://sun-valley.stanford.edu/papers/WoodleyJFLR:97.pdf> (1997)
16. Holifield, N., Lallinger, J., Underwood, G.: International aerial robotics competition 2004. University of Texas at Austin, IEEE Robot Team Aerial Robotics Project 2004 [http://iarc1.ece.utexas.edu/~lynca/final\\_documentation/utiarc 2004.pdf](http://iarc1.ece.utexas.edu/~lynca/final_documentation/utiarc 2004.pdf) (2004)
17. Burleson, W., Salhany, W., Hudak, J.: Southern Polytechnic State University autonomous remote reconnaissance system. [http://a-robotics.spsu.edu/SPSU\\_paper2005.pdf](http://a-robotics.spsu.edu/SPSU_paper2005.pdf) (2005)
18. Musial, M., Brandenburg, U.W., Hommel, G.: MARVIN—technische universität Berlin's flying robot for the IARC millennial event. In: Proc. Symposium of the Association for Unmanned Vehicle Systems 2000, Orlando, Florida, USA (2000)
19. RMAX Type I/ Type II G, Yamaha. <http://www.yamahamotor.co.jp/global/business/sky/lineup/rmax/index.html> (2004). Accessed 27 Jan 2005
20. Ludington, B., Reinmann, J., Vachtsevanos, G.: Target tracking and adversarial reasoning for unmanned aerial vehicles. In: Aerospace Conference, 2007 IEEE volume, Issue, pp. 1–17, 3–10 March 2007
21. Mejias, L., Saripalli, S., Campoy, P., Sukhatme, G.: Visual servoing of an autonomous helicopter in urban areas using feature tracking. *J. Field Robot.* **23**(3/4), 185–199, Wiley Periodicals (2006)
22. Mejias, L., Correa, J.F., Mondragon, I., Campoy, P.: COLIBRI: a vision-guided UAV for surveillance and visual inspection. In: Proceedings IEEE International Conference on Robotics and Automation, pp. 2760–2761, Rome, Italy (2007)
23. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of number of points required to represent a digitized line or its caricature. *Can. Cartogr.* **10**(2), 112–122 (1973)
24. Bell, W., Felzenszwalb, P., Huttenlocher, D.: Detection and long term tracking of moving objects in aerial video (1999)
25. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600 (1994)
26. Kanade, T., Amidi, O., Ke, Q.: Real-time and 3D vision for autonomous small and micro air vehicles. In: 43rd IEEE Conference on Decision and Control (CDC 2004) (2004)
27. Dobrokhotov, V.N., Kaminer, I.I., Jones, K.D., Ghabcheloo, R.: Vision-based tracking and motion estimation for moving targets using small UAVs. In: Proceedings of the 2006 American Control Conference, pp. 1428–1433, Minneapolis, Minnesota, USA, 14–16 June 2006
28. Mondragon, I.F., Campoy, P., Correa, J.F., Mejias, L.: Visual model feature tracking for UAV control. In: IEEE International Symposium on Intelligent Signal Processing, pp. 1–6 (2007)
29. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Visually guided landing of an unmanned aerial vehicle. *IEEE Trans. Robot. Autom.* **19**(3), 371–380 (2003)
30. Wan, C.L., Dickinson, K.W.: Computer vision and neural networks for traffic monitoring. In: Road Traffic Monitoring 1992 (IEE Conf. Pub. 355), p. 143 (1992)
31. Ollero, A., Ferruz, J., Caballero, F., Hurtado, S., Merino, L.: Motion compensation and object detection for autonomous helicopter visual navigation in the COMETS system. In: IEEE International Conference on Robotics and Automation, New Orleans, LA (USA) (2004)
32. Ruffier, F., Franceschini, N.: Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction. In: Proceedings. ICRA '04. 2004 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2339–2346, 26 Apr–1 May 2004

## Vision-Based Road-Following Using Proportional Navigation

Ryan S. Holt · Randal W. Beard

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 25 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper describes a new approach for autonomous road following for an unmanned air vehicle (UAV) using a visual sensor. A road is defined as any continuous, extended, curvilinear feature, which can include city streets, highways, and dirt roads, as well as forest-fire perimeters, shorelines, and fenced borders. To achieve autonomous road-following, this paper utilizes Proportional Navigation as the basis for the guidance law, where visual information is directly fed back into the controller. The tracking target for the Proportional Navigation algorithm is chosen as the position on the edge of the camera frame at which the road flows into the image. Therefore, each frame in the video stream only needs to be searched on the edge of the frame, thereby significantly reducing the computational requirements of the computer vision algorithms. The tracking error defined in the camera reference frame shows that the Proportional Navigation guidance law results in a steady-state error caused by bends and turns in the road, which are perceived as road motion. The guidance algorithm is therefore adjusted using Augmented Proportional Navigation Guidance to account for the perceived road accelerations and to force the

---

This research was supported by NASA under STTR contract No. NNA04AA19C to Scientific Systems Company, Inc. and Brigham Young University and by NSF award no. CCF-0428004. Portions of this work were performed while the second author was a National Research Council Fellow at the Air Force Research Laboratory, Munitions Directorate, Eglin, AFB.

R. S. Holt (✉)  
Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02420, USA  
e-mail: rholt@ll.mit.edu

R. W. Beard  
Electrical and Computer Engineering Department, Brigham Young University,  
450 Clyde Bldg., Provo, UT 84602, USA  
e-mail: beard@byu.edu

steady-state error to zero. The effectiveness of the solution is demonstrated through high-fidelity simulations, and with flight tests using a small autonomous UAV.

**Keywords** Unmanned air vehicles · Vision based guidance · Proportional navigation · Road following

## 1 Introduction

The objective of this paper is to develop a guidance law for small UAVs using on-board computer vision to follow roads and other visually distinct contours on the ground. There are a variety of applications that could potentially use this technology including border patrol, pipeline surveillance, powerline maintenance, and military convoy support. We are particularly interested in the application to forest fire monitoring [1], where the boundary of the fire is unknown *a priori*, and will be changing in time. Fire borders also have the property that they may not be contiguous. However, they will be visually distinct when using a thermal camera. Due to the difficulty associated with flight testing our approach on actual fires, we focus on vision based guidance algorithms for following rural roads.

Vision based road following has been addressed in other publications [2, 3], where it has been noted that road following consists of two challenges: (1) detecting and estimating the position of the road in the image, and (2) devising a guidance strategy to follow the road. The vision system described in [2] uses color segmentation and a Bayesian pixel classifier to distinguish road from non-road pixels. The position and direction of the road is then determined using connected component analysis, Hough transforms, and a robust line fitting scheme. The vision system described in [3] uses intensity profiles, Hough transforms, and vanishing point analysis to determine potential positions and orientations of the road. A RANSAC algorithm [4] is used to robustly identify the road.

Our approach to road detection is similar to [2] in that we use color segmentation and a connected component analysis to distinguish between road and non-road pixels. However, rather than searching the entire image and identifying the road in the image, we simply identify the position on the boundary of the camera plane where the road flows into the image. The associated image processing does not require image rectification or camera calibration and should represent significantly reduced computational requirements relative to the approaches described in [2] and [3].

In [2], three guidance strategies are proposed for following a road. The first strategy is to wrap a PID loop around the lateral cross track error, which, according to [2], did not work well enough in simulation to warrant flight testing. In the other two approaches reported in [2], the MAV is commanded to track a position on the road a specified distance in front of the projected position of the UAV. The algorithms were demonstrated in simulation and limited flight tests. The guidance strategy proposed in [3] constructs a connecting curve between the current position of the UAV and a position on the desired road a specified distance in front of the

projected position of the UAV. This method was also successfully demonstrated with flight results. However, as acknowledged in [3], a weakness of this method is that it will be susceptible to wind.

In this paper we present a guidance strategy that is significantly different than those proposed in [2, 3]. Our approach is based on the Proportional Navigation, and Augmented Proportional Navigation schemes that are well known in the missile guidance literature [5–7]. Proportional Navigation is most commonly used to guide missiles to evading targets, both on the ground and in the air. The basic idea is that the missile will be on a collision course with the target if the line of sight vector between the pursuer and the evader is not rotating in the inertial frame. The proportional navigation algorithm is readily implemented by commanding body frame accelerations that are proportional to the line-of-sight rates multiplied by a closing velocity. Typically, LOS rates are available from sensors on a gimbaled seeker and the closing velocity is taken from radar measurements. Though noisy, these inputs to the guidance law are directly measurable. A Kalman filter can be used to mitigate the sensor noise, and the output of this filter is the input to the guidance law.

Proportional navigation has been extensively analyzed in the literature. It has been shown to be optimal under certain conditions [8] and to produce zero miss distances for a constant target acceleration [9]. If rich information regarding the time-to-go is available, augmented proportional navigation [6] improves the performance by adding terms that account for target and pursuer accelerations. A three-dimensional expression of proportional navigation can be found in [10], where the authors compare proportional navigation to a minimum-energy optimal guidance law.

Proportional navigation has many advantages which account for its use on most of the missile systems that have been deployed in the past forty years. The advantages include robustness to ambient winds, and robustness to sensor calibration and alignment errors [5–7]. If the target that is being tracked is accelerating, then the standard proportional navigation algorithm can be augmented to account for that motion.

The application of proportional navigation to road following is not immediately obvious. In contrast to missile guidance problems, our objective is not to collide with the road, but rather to track the road at a fixed altitude. The primary contribution of this paper is to show how the proportional navigation algorithm can be adapted to vision based road following. The basic idea is that we treat the position that the road flows into the image as the target, and pursue this target as if it were at the same altitude as the UAV. A key insight is that when the road bends and curves, the apparent motion in the image plane is identical to that of an accelerating target. Therefore augmented proportional navigation can be used to track roads with significant curvature. In other words, our method is not restricted to locally linear structures like the methods proposed in [2, 3]. Our method also inherits the well-known strengths of proportional navigation including robustness to wind, and robustness to camera calibration and alignment errors.

One of the significant differences between missiles and small UAVs is that most missiles are skid-to-turn platforms, whereas most small UAVs are bank-to-turn platforms. For missiles that are bank-to-turn platforms, the rolling dynamics are relatively fast and can therefore be approximated as skid-to-turn platforms, whereas

that may not be the case for small UAVs. The proportional navigation algorithms are most naturally derived for skid-to-turn models. In this paper we will derive a road following algorithm using a skid-to-turn kinematic model and then show how to map the result to a bank-to-turn vehicle with a single axis gimballed camera.

In Section 2, we define both bank-to-turn and skid-to-turn kinematics models, as well as the relevant reference frames. Section 3 contains the main result of the paper, where we derive the vision based proportional navigation road following algorithm for the skid-to-turn model and then map it to a bank-to-turn platform with a two axis gimbal. In Section 4, we introduce the image steady-state error and use augmented proportional navigation to force the error to zero. In Section 5, the experimental simulation and hardware testbeds are described, and simulation and flight results are presented in Section 6. Conclusions are contained in Section 7.

## 2 Navigation and Camera Models

To reduce cost and complexity, many small UAV platforms are not fully actuated. In particular, many small UAVs do not have rudder control and must therefore bank to turn. However, because it is much simpler to design a guidance law for skid-to-turn systems, we will begin the derivation of the proportional navigation algorithm with a fully actuated (skid-to-turn) model.

For a fully actuated skid-to-turn vehicle, we assume that the actuators consist of ailerons maintaining zero roll angle while a rudder steers the heading of the vehicle. The kinematics can be modeled as

$$\dot{p}_n = V \cos \chi \cos \gamma, \quad (1)$$

$$\dot{p}_e = V \sin \chi \cos \gamma, \quad (2)$$

$$\dot{h} = V \sin \gamma, \quad (3)$$

$$\dot{\chi} = u_1, \quad (4)$$

$$\dot{\gamma} = u_2, \quad (5)$$

where  $p_n$  and  $p_e$  are the North and East positions of the UAV,  $h$  is the altitude,  $V$  is the airspeed (assumed to be constant),  $\chi$  is the course angle,  $\gamma$  is the flight path angle, and  $u_1$  and  $u_2$  are control inputs. We will also assume that the skid-to-turn platform is equipped with a strap-down camera, where the camera is attached to the belly of the vehicle and directed downward. The kinematic model (1)–(5) assumes that the windspeed is negligible.

For an under-actuated bank-to-turn vehicle, we assume that the actuators are elevators and ailerons and that winglets are used to maintain the small UAV in a coordinated turn. The kinematics for this type of platform can be modeled as:

$$\dot{p}_n = V \cos \chi \cos \gamma, \quad (6)$$

$$\dot{p}_e = V \sin \chi \cos \gamma, \quad (7)$$

$$\dot{h} = V \sin \gamma, \quad (8)$$

$$\dot{\chi} = \frac{g}{V} \tan \phi, \quad (9)$$

$$\dot{\phi} = u_3, \quad (10)$$

$$\dot{\gamma} = u_4, \quad (11)$$

where  $g$  is the gravitational constant and  $u_3$  and  $u_4$  are control inputs. The bank-to-turn platform is assumed to have a gimballed camera where we define  $\alpha_{az}$  to be the gimbal azimuth angle and  $\alpha_{el}$  to be the gimbal elevation angle. These angles are assumed to evolve according to the first order model

$$\dot{\alpha}_{az} = u_5, \quad (12)$$

$$\dot{\alpha}_{el} = u_6, \quad (13)$$

where  $u_5$  and  $u_6$  are gimbal controls.

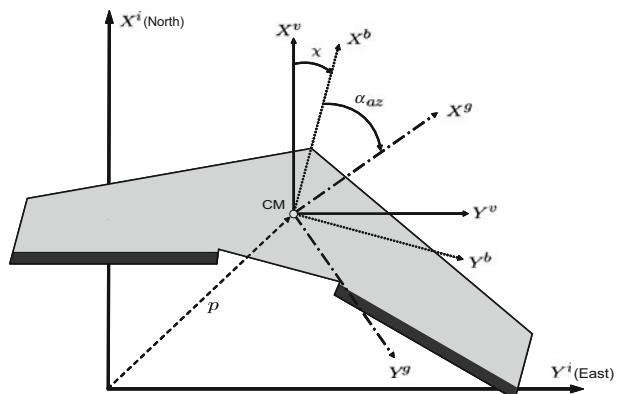
We will briefly define the coordinate frames of interest, specifically, the inertial, vehicle, body, gimbal, and camera coordinate frames, and how they relate to one another. Table 1 presents the notation for the respective axes in each coordinate frame. The superscript on each axis is used to denote the reference frame with respect to which the vector is defined.

Figures 1 and 2 illustrate each of the coordinate frames and their corresponding relationships. For the inertial frame, we use the NED coordinates  $(p_n, p_e, -h)$ . The relationship between the inertial frame and the vehicle frame is a simple translation

**Table 1** Notation of coordinate frame axes

Coordinate frame	Axes
Inertial	$(X^i, Y^i, Z^i)$
Vehicle	$(X^v, Y^v, Z^v)$
Body	$(X^b, Y^b, Z^b)$
Gimbal	$(X^g, Y^g, Z^g)$
Camera	$(X^c, Y^c, Z^c)$

**Fig. 1** A lateral perspective of the coordinate frames. The inertial and vehicle frames are aligned with the world, the body frame is aligned along the airframe, and the gimbal and camera frames are aligned along the camera

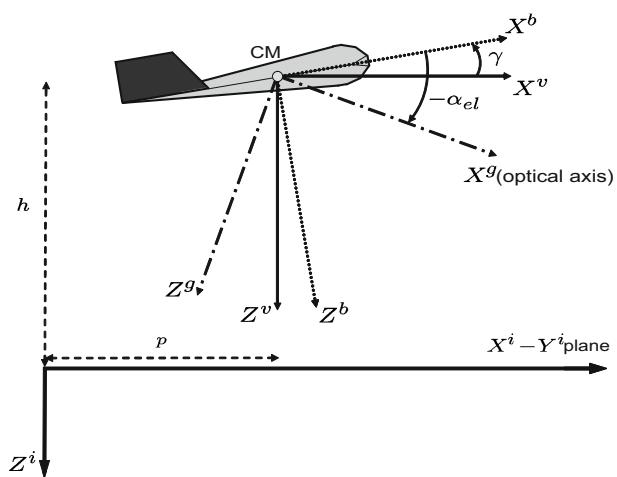


from the origin of the inertial frame to the center of mass (CM) of the UAV. The body frame requires a rotation from the vehicle frame so that  $X^b$  is directed out the nose,  $Y^b$  is directed out the right wing, and  $Z^b$  is directed out the belly of the vehicle.

In general, the rotation of a vector from coordinate system  $j$  to coordinate system  $k$  can be expressed as a rotation matrix  $R_j^k$ . Because every rotation matrix is unitary, its inverse is equal to its transpose:  $R_k^j = (R_j^k)^{-1} \triangleq (R_j^k)^T$ . For the vehicle-to-body rotation, the rotation matrix is

$$R_v^b = \begin{pmatrix} c_\gamma c_\chi & c_\gamma s_\chi & -s_\gamma \\ s_\phi s_\gamma c_\chi - c_\phi s_\chi & s_\phi s_\gamma s_\chi + c_\phi c_\chi & s_\phi c_\gamma \\ c_\phi s_\gamma c_\chi + s_\phi s_\chi & c_\phi s_\gamma s_\chi - s_\phi c_\chi & c_\phi c_\gamma \end{pmatrix}, \quad (14)$$

**Fig. 2** A longitudinal perspective of the coordinate frames



where  $c_* \triangleq \cos(*)$  and  $s_* \triangleq \sin(*)$ . Assuming that the camera is located at the CM of the UAV, the rotation from the body coordinate system to the gimbal coordinate system is defined as

$$R_b^g = \begin{pmatrix} c_{\alpha_{el}}c_{\alpha_{az}} & c_{\alpha_{el}}s_{\alpha_{az}} & -s_{\alpha_{el}} \\ -s_{\alpha_{az}} & c_{\alpha_{az}} & 0 \\ s_{\alpha_{el}}c_{\alpha_{az}} & s_{\alpha_{el}}s_{\alpha_{az}} & c_{\alpha_{el}} \end{pmatrix}. \quad (15)$$

In the camera frame,  $X^c$  is directed to the right of the image,  $Y^c$  is directed toward the bottom of the image, and  $Z^c$  is directed along the optical axis of the camera. The origin for the camera frame is located in the center of the image. The rotation from the gimbal coordinate system to the camera coordinate system is

$$R_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \quad (16)$$

### 3 Vision-Based Proportional Navigation

Our objective is to design a vision-based guidance law that commands the UAV to track a road, where image processing is used to extract the location of the road in the image. While this paper will not focus on the image-processing algorithm, we will briefly describe the necessary information that is to be extracted from the image. Once this information is defined, we will derive the proportional-navigation equations for the skid-to-turn model equipped with a strap-down camera. We will then extend those equations to the bank-to-turn model using a gimballed camera.

#### 3.1 Image-Processing Algorithm

In the missile literature, the proportional navigation (PN) guidance law is based upon the relative position and velocity between the interceptor and the target [5]. Therefore, in order to use PN for road-following, we must define both the position of the road as a single point and its associated velocity in the camera frame. Our approach is unique in that most road-following techniques involve fitting a line to the road rather than a single point [2, 11–13]. As will be shown, the amount of computations for representing the road by a point is significantly less than fitting a line to the road. We begin by defining the position and velocity of the road in the camera frame.

In road-following, the desired behavior is for the UAV to travel directly above the road in the inertial frame. This allows the UAV more time to react to changes in the direction of the road. We define the configuration where the UAV is traveling directly above the road, to be the desired, steady-state behavior that the UAV should converge to. Once steady state is reached, any deviation of the road or change in the position of the UAV is viewed as a disturbance to the system. The UAV should adjust to disturbances and return to steady state.

In steady state, the road will be centered vertically in the image, extending from the top to the bottom. We will refer to the road as *entering* from the top and *exiting*

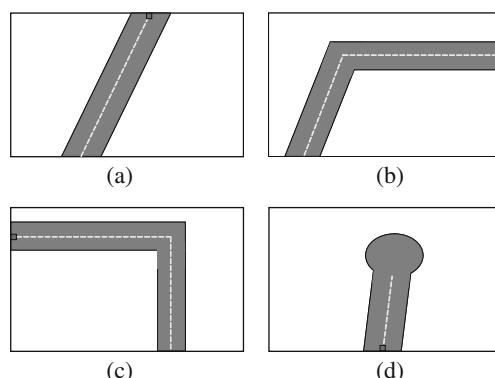
the bottom of the image. Note that once the road is in the image, there will always be an entrance as well as an exit location. For example, in the case where there is a 90 degree right-hand turn in the image, the entrance location is the right side of the image and the exit location is the bottom of the image. In the case where the road is a dead-end, the entrance location is the same as the exit location at the bottom of the image.

For a downward-facing camera, the center of the image corresponds to the North/East position of the UAV. To have sufficient time to react to any deviation in the road direction, the UAV must look beyond its current position. Therefore, it must use the information pertaining to the road in the upper half of the image. In order to maximize reaction time to road deviations while in steady-state, we define the position of the road to be the pixel on edge of the image corresponding to the center of the road precisely where the road *enters* the image. Figure 3 illustrates which pixel location is identified as the road by the image-processing algorithm for four different scenarios. The velocity of the road is defined as the rate of change of this pixel in the video sequence.

For this paper, a simple thresholding in the HSV (hue-saturation-value) color space was used to separate road pixels from non-road pixels [14]. This was followed by a connected-components analysis that groups road pixels together into components. The largest component was then selected as the road and the top center pixel was returned to the UAV controller. While this approach was simple and effective, there are a variety of other algorithms that could be used for image processing as long as the delivery time of the needed information meets the demands of the guidance law to ensure stable motion.

When fitting a line to the road, the entire image needs to be searched. However, by following the approach described above, we can significantly limit the number of image pixels that are processed. Most of the time the road will enter from the top of the image, which implies that only the top section needs to be searched. If the road is not detected in the top portion of the image, then the sides can be processed. If that fails, then the bottom of the image can be searched. If this fails as well, then the road is not in the image. Therefore, the center portion of the image does not need to be processed, thereby significantly speeding up the algorithm.

**Fig. 3** Images from the perspective of the UAV in four different road scenarios, where the pixel corresponding to the road position returned by the image-processing algorithm is highlighted in red. **a** Straight road. **b** Right turn. **c** Left turn. **d** Dead end



The size definition of the top, sides, and bottom sections is dependent upon the quality of the road classification. If the road in the image is hard to extract from its surrounding environment, then the sections should be made larger. To give an example of adequate sizes, the flight-tests described in this paper were performed using a top section of  $640 \times 140$  pixels, side sections of  $100 \times 480$  pixels, and a bottom sections of  $640 \times 100$  pixels, where the entire image was  $640 \times 480$ . In the best case, which occurs most of the time, only the top section is searched limiting the analyzed pixels to 29.17% of the total number of pixels. In the worst case, where all the sections are searched, 57.81% of the image is analyzed. This is a significant improvement in the number of pixel operations needed to extract the road information from the image.

### 3.2 Skid-to-Turn Model with Strap-Down Camera

This section derives the guidance law for vision-based road-following for a skid-to-turn vehicle model. We will follow the derivation of 3D proportional navigation presented in [15], adjusting the equations and assumptions for road-following.

We begin by defining the inertial position and velocity of the UAV as

$$\mathbf{r}_{UAV}^i = (p_n, p_e, -h)^T \quad (17)$$

and

$$\mathbf{v}_{UAV}^i = \begin{pmatrix} V \cos \chi \cos \gamma \\ V \sin \chi \cos \gamma \\ -V \sin \gamma \end{pmatrix}. \quad (18)$$

Assuming a small angle-of-attack, the velocity vector of the UAV can be approximated in body coordinates as

$$\mathbf{v}_{UAV}^b = (V, 0, 0)^T. \quad (19)$$

The inertial position and velocity of the road are denoted as  $\mathbf{r}_{road}^i$  and  $\mathbf{v}_{road}^i$ . Let  $\mathbf{R}^i \triangleq \mathbf{r}_{road}^i - \mathbf{r}_{UAV}^i$  be the relative position vector between the road and the UAV expressed in inertial coordinates.

The 3D equations for pure proportional navigation guidance (PPNG) are given in [10] as

$$\mathbf{a}_{UAV} = N\Omega_{\perp} \times \mathbf{v}_{UAV}, \quad (20)$$

where  $N$  is the navigation constant and

$$\Omega_{\perp} = \frac{\mathbf{R} \times (\mathbf{v}_{road}^i - \mathbf{v}_{UAV}^i)}{\|\mathbf{R}\|^2}, \quad (21)$$

is the component of the angular rate of the LOS that is perpendicular to the LOS. By noting that  $\dot{\mathbf{R}} = \dot{\mathbf{r}}_{road} - \dot{\mathbf{r}}_{UAV} = \mathbf{v}_{road} - \mathbf{v}_{UAV}$  and defining

$$\hat{\ell} \triangleq \frac{\mathbf{R}}{\|\mathbf{R}\|} \quad (22)$$

and

$$\hat{w} \triangleq \frac{\dot{R}}{\|R\|}, \quad (23)$$

we can express  $\Omega_{\perp}$  as

$$\begin{aligned} \Omega_{\perp} &= \frac{R}{\|R\|} \times \frac{\dot{R}}{\|R\|} \\ &= \hat{\ell} \times \hat{w}. \end{aligned} \quad (24)$$

The control inputs for the models defined in Section 2 are in body coordinates. Therefore, the acceleration command in Eq. 20 should also be expressed in body coordinates as

$$a_{UAV}^b = N\Omega_{\perp}^b \times v_{UAV}^b. \quad (25)$$

Using Eq. 19, Eq. 25 becomes

$$a_{UAV}^b = N \begin{pmatrix} \Omega_{\perp,x}^b \\ \Omega_{\perp,y}^b \\ \Omega_{\perp,z}^b \end{pmatrix} \times \begin{pmatrix} V \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ NV\Omega_{\perp,z}^b \\ -NV\Omega_{\perp,y}^b \end{pmatrix}. \quad (26)$$

To derive the commanded acceleration, we need to estimate  $\Omega_{\perp}^b$  from the visual information. Our approach will be to estimate  $\Omega_{\perp}$  in camera coordinates and then to transform it into body coordinates using

$$\Omega_{\perp}^b = R_g^b R_c^g \Omega_{\perp}^c. \quad (27)$$

Let  $\epsilon_x$  and  $\epsilon_y$  be the X and Y pixel location corresponding to the position of the road as described in Section 3.1 and let  $\dot{\epsilon}_x$  and  $\dot{\epsilon}_y$  be the X and Y time derivatives of the pixel location. The focal length  $f$  of the camera in units of pixels is defined as

$$f = \frac{M_W}{2 \tan(\frac{\eta}{2})}, \quad (28)$$

where  $\eta$  is the horizontal field-of-view and  $M_W$  is the width of the image in pixels.

The 3D geometry of the camera model is shown in Fig. 4. The displacement of the road from the origin of the camera in the image plane is the vector having a direction of  $(P\epsilon_x, P\epsilon_y, Pf)$  and a magnitude of  $P\bar{f}$ , where

$$\bar{f} = \sqrt{\epsilon_x^2 + \epsilon_y^2 + f^2}, \quad (29)$$

and where  $P$  is a conversion factor from pixels to meters. The displacement of the road from the origin of the camera in the world plane is the vector  $(R_x^c, R_y^c, R_z^c)$  having a magnitude of  $\|R\|$ .

By using similar triangles, we can form ratios among these displacements as

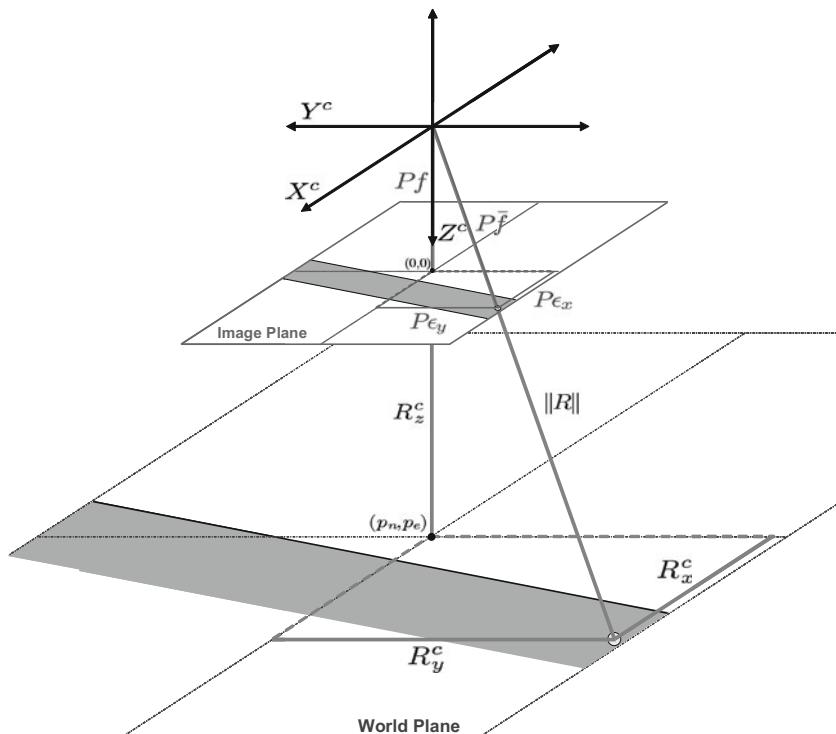
$$\hat{e}_x^c = \frac{R_x^c}{\|R\|} = \frac{P\epsilon_x}{P\bar{f}} = \frac{\epsilon_x}{\bar{f}}, \quad (30)$$

$$\hat{e}_y^c = \frac{R_y^c}{\|R\|} = \frac{P\epsilon_y}{P\bar{f}} = \frac{\epsilon_y}{\bar{f}}, \quad (31)$$

$$\hat{e}_z^c = \frac{R_z^c}{\|R\|} = \frac{Pf}{P\bar{f}} = \frac{f}{\bar{f}}. \quad (32)$$

Thus, the first term on the right side of Eq. 24 is expressed as

$$\hat{\ell}^c = \frac{1}{\bar{f}} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}. \quad (33)$$



**Fig. 4** The 3D camera model. The displacements of the road in the image plane are indicated in red while the displacements in the world plane are shown in blue

From the definition  $\hat{\ell} = R/\|R\|$  we have that

$$\begin{aligned}\frac{d\hat{\ell}}{dt} &= \frac{\|R\|\dot{R} - R\frac{d\|R\|}{dt}}{\|R\|^2} \\ &= \frac{\dot{R}}{\|R\|} - \frac{R}{\|R\|}\frac{d\|R\|}{\|R\|}.\end{aligned}$$

Therefore, since  $\hat{w} = \dot{R}/\|R\|$  we have

$$\hat{w} = \frac{d\hat{\ell}}{dt} + \hat{\ell}\frac{\frac{d\|R\|}{dt}}{\|R\|}. \quad (34)$$

From Eq. 33 we have

$$\begin{aligned}\frac{d\hat{\ell}^c}{dt} &= \frac{\bar{f}\begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \dot{\bar{f}}\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{\bar{f}^2} \\ &= \frac{1}{\bar{f}}\begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \frac{\dot{\bar{f}}}{\bar{f}}\hat{\ell}^c.\end{aligned} \quad (35)$$

From the relation  $\|R\| = R_z^c \frac{\bar{f}}{f}$  we have

$$\begin{aligned}\frac{d\|R\|}{dt} &= \dot{R}_z^c \frac{\bar{f}}{f} + R_z^c \left( \frac{f\dot{\bar{f}} - \bar{f}\dot{f}}{f^2} \right) \\ &= R_z^c \frac{\dot{\bar{f}}}{f} \\ &= \|R\| \frac{\dot{\bar{f}}}{\bar{f}}.\end{aligned}$$

Therefore

$$\frac{\frac{d\|R\|}{dt}}{\|R\|} = \frac{\dot{\bar{f}}}{\bar{f}}. \quad (36)$$

Substituting Eqs. 35 and 36 into Eq. 34 gives

$$\hat{w}^c = \frac{1}{\bar{f}}\begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix}. \quad (37)$$

We have shown that  $\hat{\ell}^c$  and  $\hat{w}^c$  can be estimated solely upon the visual information. Therefore,  $\Omega_{\perp}^c$  can be expressed as

$$\begin{aligned}\Omega_{\perp}^c &= \hat{\ell}^c \times \hat{w}^c \\ &= \frac{1}{\bar{f}^2} \begin{pmatrix} -\dot{\epsilon}_y f \\ \dot{\epsilon}_x f \\ \epsilon_x \dot{\epsilon}_y - \epsilon_y \dot{\epsilon}_x \end{pmatrix}. \end{aligned} \quad (38)$$

We now rotate  $\Omega_{\perp}^c$  into body coordinates using the expression

$$\begin{aligned}\Omega_{\perp}^b &= R_g^b R_c^g \Omega_{\perp}^c \\ &= \frac{1}{\bar{f}^2} \begin{pmatrix} -\dot{\epsilon}_x f \\ -\dot{\epsilon}_y f \\ \epsilon_x \dot{\epsilon}_y - \epsilon_y \dot{\epsilon}_x \end{pmatrix}. \end{aligned} \quad (39)$$

By using Eqs. 26 and 39, the PPNG acceleration in body coordinates is

$$\begin{aligned}a_{UAV}^b &= \begin{pmatrix} 0 \\ NV \Omega_{\perp,z}^b \\ -NV \Omega_{\perp,y}^b \end{pmatrix} \\ &= \frac{1}{\bar{f}^2} \begin{pmatrix} 0 \\ NV (\epsilon_x \dot{\epsilon}_y - \epsilon_y \dot{\epsilon}_x) \\ NV (\dot{\epsilon}_y f) \end{pmatrix}. \end{aligned} \quad (40)$$

In target-tracking,  $a_{UAV,y}^b$  will direct the heading angle of the UAV toward the inertial position of the target, while  $a_{UAV,z}^b$  will direct the pitch angle toward the altitude of the target. However, in road-following, the UAV is not trying to intercept the road but rather maintain constant altitude while tracking the road. Therefore,  $a_{UAV,z}^b$  is ignored and the desired altitude is commanded using the standard altitude hold mode in the autopilot. The angular acceleration  $\dot{\chi}$  is related to the linear acceleration  $a_{UAV,y}^b$  by the equation

$$\dot{\chi} = \frac{1}{V} a_{UAV,y}^b. \quad (41)$$

We summarize the derivation for the skid-to-turn model with a strap-down camera by defining the control inputs in Eqs. 4 and 5 as

$$u_1 = N \left( \frac{\epsilon_x \dot{\epsilon}_y - \epsilon_y \dot{\epsilon}_x}{\epsilon_x^2 + \epsilon_y^2 + \bar{f}^2} \right) \quad (42)$$

and

$$u_2 = k_h (h^d - h), \quad (43)$$

where  $k_h$  is a proportional gain, and  $h^d$  is the desired altitude.

### 3.3 Bank-to-Turn Model with Gimbaled Camera

In this section, we derive the proportional navigation equations for the bank-to-turn model equipped with a gimbaled camera. We begin the derivation by noting that if the camera is directed downward along the  $Z^v$ -axis, regardless of the pitch and roll angles of the UAV, the camera always has the same position and orientation as a strapped-down camera on a skid-to-turn UAV. Therefore, if the gimbal can be commanded to cancel out the pitch and roll effects of the UAV, the results derived in Section 3.2 are directly applicable to the bank-to-turn model.

In Section 3.2, we assumed that the UAV maintains constant altitude. If winglets maintain the vehicle in a coordinated turn, then the pitch angle is essentially zero. Therefore, the gimbal need only cancel out the effects caused by rolling. A roll is defined as a rotation about the  $X^b$ -axis. If the gimbal is oriented so that it also rotates about this axis, a rotation angle equal to the roll angle and opposite in direction will cancel out the effects of the roll on the image. To implement this idea, we position the gimbal so that its initial orientation is pointed downward along the  $Z^b$ -axis, while the UAV is in level flight. The gimbal is then only allowed to move in the  $Y^b$ - $Z^b$  plane. If the azimuth angle  $\alpha_{az} = \pi/2$ , then the negative elevation angle  $-\alpha_{el}$  is the displacement angle from the  $Y^b$ -axis in the  $Y^b$ - $Z^b$  plane. Defining  $\tilde{\alpha}_{el} = \alpha_{el} + \pi/2$ , then when  $\tilde{\alpha}_{el} > 0$ , the gimbal will rotate toward the right wing and when  $\tilde{\alpha}_{el} < 0$  the gimbal will rotate toward the left wing. Note that this rotation is defined opposite in direction to that of the UAV roll angle.

With this setup, the equations in Section 3.2 are used to derive the control inputs defined in Eqs. 10, 11, 12, and 13. In Eq. 4,  $u_1$  commands  $\dot{\chi}$ , which we can combine with Eq. 9 to derive a desired roll angle:

$$\begin{aligned}\phi^d &= \frac{V}{g} \arctan(\dot{\chi}) \\ &= \frac{V}{g} \arctan(u_1).\end{aligned}\quad (44)$$

Using the desired roll angle,  $\phi^d$ , we can define the control inputs introduced in Eqs. 10–13 for the bank-to-turn model equipped with a gimbaled camera as

$$\phi^d = \frac{V}{g} \arctan(u_1), \quad (45)$$

$$u_3 = k_\phi (\phi^d - \phi), \quad (46)$$

$$u_4 = k_h (h^d - h), \quad (47)$$

$$u_5 = 0, \quad (48)$$

$$u_6 = k_{el} \left( \phi - \frac{\pi}{2} + \alpha_{el} \right), \quad (49)$$

where  $k_\phi$ ,  $k_h$ , and  $k_{el}$  are positive control gains. Note that for the above assumptions to hold,  $\alpha_{el}$  needs to be updated frequently enough to track the roll angle. Also, the response of the gimbal needs to be fast enough to track the UAV roll dynamics.

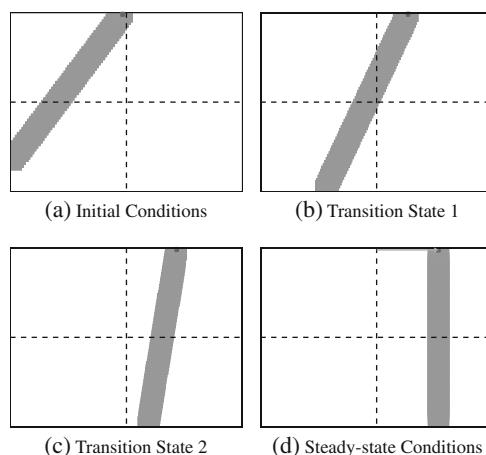
By using  $u_1$  to calculate  $u_3$ , the rotation matrix  $R_g^b$  is equivalent to that expressed in the skid-to-turn model derivation and is *not* recomputed with the new camera elevation angle. By assuming that the camera is directed downward along the  $Z^i$ -axis, the controller is unaware that the vehicle is rolling and that the gimbal is countering its motion.

#### 4 Image Steady-State Error

In this section, we introduce the idea of an image steady-state error and present an adjustment to the guidance laws developed in Sections 3.2 and 3.3 to force this error to zero. Consider the situation for the skid-to-turn model when the road is vertical in the image but not centered. Both  $\dot{\epsilon}_x$  and  $\dot{\epsilon}_y$  are zero, causing the control input  $u_1$  to be zero. Therefore the UAV no longer maneuvers to center the road in the image plane. Figure 5 demonstrates this situation by showing snapshots from a numerical simulation: (a) represents initial conditions, (b) and (c) take place during the tracking phase, and (d) occurs when the UAV has reached a steady state. While the UAV is still tracking the road in this setting, it is non-ideal. If a sharp turn in the road were to occur, the time to react would be severely limited and may cause the road to leave the image. Hence, it is important for the road to converge to the center of the image. We define the image steady-state error as the displacement of the road position from the top center pixel of the image.

The image steady-state error results from the manner in which the image processing was designed. Selecting the top pixel corresponding to the road for each video frame produces the effect that the road is traveling at the same velocity as the UAV. Proportional navigation is designed to command the UAV to intercept the target. However, if the UAV and target are traveling at the same velocity, any movement by the UAV that is not in the direction of the target's heading will cause the UAV to move farther away from the target. Therefore, the best maneuver for the UAV is to direct its heading to that of the target so that the two travel in parallel. We can classify the effective road velocity between video frames as a maneuver by the road,

**Fig. 5** Four screen shots of the road-following algorithm in a Matlab simulation, demonstrating the steady-state error where **a** is the initial state, **b** and **c** are different states that occurred during the transition to steady-state, and **d** is the steady-state condition



since it changes as the vehicle's velocity changes. Any curves or turns in the road can also be viewed as a maneuver.

Equation 20 has been shown to produce zero miss-distance for targets with zero acceleration, but for maneuverable targets the commanded acceleration needs to be adjusted [16]. This is referred to in the missile guidance literature as augmented proportional navigation guidance (APNG) where the adjustment to the control law is based upon the acceleration of the target.

To describe the acceleration of the road, first note that if there is no acceleration the road position should converge to the top center of the image. If the given pixel location is not at the desired location then the road is viewed as having performed a maneuver. Therefore, the image steady-state error can be used to augment the PPNG law.

The image steady-state error in the camera frame is defined as

$$e_{ss}^c = \begin{pmatrix} \frac{\epsilon_x - \epsilon_x^d}{M_W} + \text{sign}(\epsilon_x - \epsilon_x^d) \frac{|\epsilon_y - \epsilon_y^d|}{M_H} \\ \frac{\epsilon_y - \epsilon_y^d}{M_H} + \text{sign}(\epsilon_y - \epsilon_y^d) \frac{|\epsilon_x - \epsilon_x^d|}{M_W} \\ 0 \end{pmatrix}, \quad (50)$$

where  $M_W$  is the number of pixels across the width of the image,  $M_H$  is the number of pixels across the height, and the *sign* function is defined as

$$\text{sign}(x) \triangleq \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}. \quad (51)$$

The first term in each error component is the main error source in that dimension. The second term is used as an additional push in the correct direction when the road deviation is large. To augment the acceleration command in Eq. 40,  $e_{ss}^c$  must be rotated into the body frame as

$$e_{ss}^b = R_g^b R_c^b e_{ss}^c$$

$$= \begin{pmatrix} -\frac{\epsilon_y - \epsilon_y^d}{M_H} + \text{sign}(\epsilon_y - \epsilon_y^d) \frac{|\epsilon_x - \epsilon_x^d|}{M_W} \\ \frac{\epsilon_x - \epsilon_x^d}{M_W} + \text{sign}(\epsilon_x - \epsilon_x^d) \frac{|\epsilon_y - \epsilon_y^d|}{M_H} \\ 0 \end{pmatrix}. \quad (52)$$

The new command input is therefore

$$u_1 = N \left( \frac{\epsilon_x \dot{\epsilon}_y - \epsilon_y \dot{\epsilon}_x}{\bar{f}^2} \right) + L \left( \frac{\epsilon_x - \epsilon_x^d}{M_W} + \text{sign}(\epsilon_x - \epsilon_x^d) \frac{|\epsilon_y - \epsilon_y^d|}{M_H} \right), \quad (53)$$

where  $L$  is a tunable gain. The first term on the right side of Eq. 53 will be zero only when the road is vertical in the image. The second term will be zero only when the top of the road is centered in the image. The combination of these two terms guarantees a single, steady-state condition where the road is vertical and centered in the image. Note that the only adjustment to the guidance inputs defined in Sections 3.2 and 3.3

is to  $u_1$ , since  $u_2$ ,  $u_4$ ,  $u_5$ , and  $u_6$  are independent of the PPNG law and  $u_3$  will be adjusted by the augmentation to  $u_1$ .

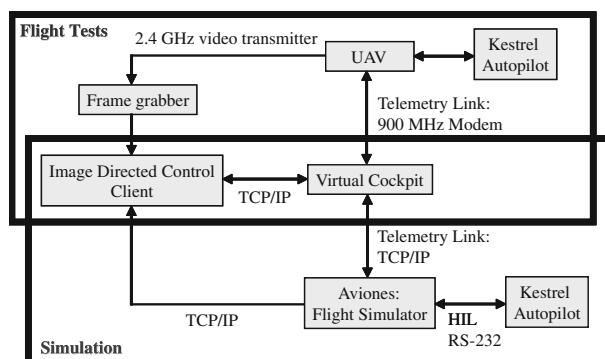
## 5 Experimental Testbed

This section describes the experimental platform used to test and validate the road following algorithm developed in the previous section. The testbed is comprised of both hardware and software. The hardware consists of a UAV equipped with an autopilot and various sensors. The software includes a simulation program that is used to test the autopilot code, and a program that is used for interfacing with both the simulator and the actual autopilot. Figure 6 is a flowchart describing the interaction of the hardware and software.

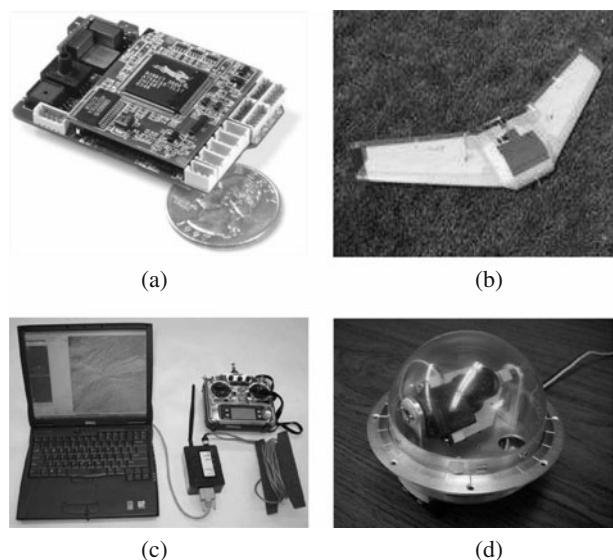
### 5.1 Hardware

Figure 7 displays the main hardware components of the experimental testbed. Figure 7a shows the Kestrel autopilot designed at Brigham Young University (BYU) and manufactured by Procerus Technologies. It is equipped with a Rabbit 3000 29-MHz microprocessor, rate gyroscopes, accelerometers, absolute and differential pressure sensors. The autopilot measures  $2.0 \times 1.37 \times 0.47$  in. and weighs 16.65 g, making it ideal for small aircraft. Figure 7b shows the airframe used in the flight tests. It is a custom designed flying wing constructed of expanded polypropylene (EPP) foam and coated with Kevlar, with a wingspan of 48 in. This airframe is hand-deployable and resilient to crashes, making it a good research and test vehicle. Embedded within the foam are the autopilot, batteries, a 1000-mW/900-MHz radio modem, a GPS receiver, a video transmitter, and a small analog camera mounted on a pan-tilt gimbal. The fully loaded weight is just under two pounds. With this setup the UAV can fly for approximately 40 min at a cruise speed of 13 m/s. The gimbal, which was designed at BYU is shown in Fig. 7d. The azimuth axis of the gimbal is actuated by a Hitec HS-77BB servo with 61.1 oz-in of torque. The elevation axis is actuated by a Hitec HS-50 servo with 8.22 oz-in of torque. The camera is a Panasonic KX-141 camera with 480 lines of resolution. Figure 7c shows the components that comprise the ground station. A laptop runs the Virtual Cockpit ground-control software, allowing the user to interface with the UAV via a communication box.

**Fig. 6** A flowchart depicting the layout of the basic hardware and software components used both in simulation and flight-tests



**Fig. 7** The hardware components used in flight-testing the algorithms presented in this paper. **a** Kestrel autopilot. **b** Airframes. **c** Ground station. **d** Gimbal



A remote control (RC) transmitter also is attached to the communication box, acting as a standby fail-safe mechanism to facilitate safe operation.

## 5.2 Software

There are two main software components that were used in simulating and testing the developed algorithms. The first, Aviones, is a flight simulator developed at BYU, which emulates both the physics of the airframe as well as the communication between the ground station and the UAV. The motion of each virtual UAV is



(a) Screen Capture of Aviones      (b) Screen Capture of Virtual Cockpit

**Fig. 8** **a** Screen capture of Aviones with a simulated aircraft in flight. **b** Screen capture of Virtual Cockpit with the navigation screen displayed on the right-hand side. The UAV tracks the path defined by the four blue stars. The artificial horizon along with the desired and actual attitude values are located on the left-hand side

calculated from full nonlinear, six-degree-of-freedom equations of motion [17]. Aviones is adaptable to many styles of aircraft and various terrain conditions. The most powerful aspect of Aviones is that the autopilot code tested in simulation is identical to the code on the actual autopilot, allowing for quick transfer from software development to flight-testing. Figure 8a shows a screen capture of the Aviones software during a simulated flight-test.

The second piece of software that is utilized in both simulation and flight-testing is Virtual Cockpit. Virtual Cockpit connects to Aviones through a TCP/IP connection and allows the user to interface with the simulated UAV in the same manner as during a real-world flight-test. In both simulation and testing, Virtual Cockpit allows the user to view the current attitude of the UAV, tune the control loops in real-time, adjust various algorithmic variables, and command a list of waypoints for the UAV to follow. Additionally, a frame processor may be connected to Virtual Cockpit through a TCP/IP connection, allowing for video images from the UAV to be converted into control commands and sent to the UAV. Figure 8b shows a screen capture of the Virtual Cockpit software.

## 6 Results

### 6.1 Simulation Results

To verify the feasibility of the developed road-following guidance law, medium fidelity simulations were conducted using the software described in Section 5.2. The UAV model used in the simulations was the bank-to-turn platform with a gimballed camera, where the following constraints were imposed upon the air vehicles:

$$V = 13 \text{ m/s}, \quad (54)$$

$$-45 \text{ deg} \leq \phi \leq 45 \text{ deg}, \quad (55)$$

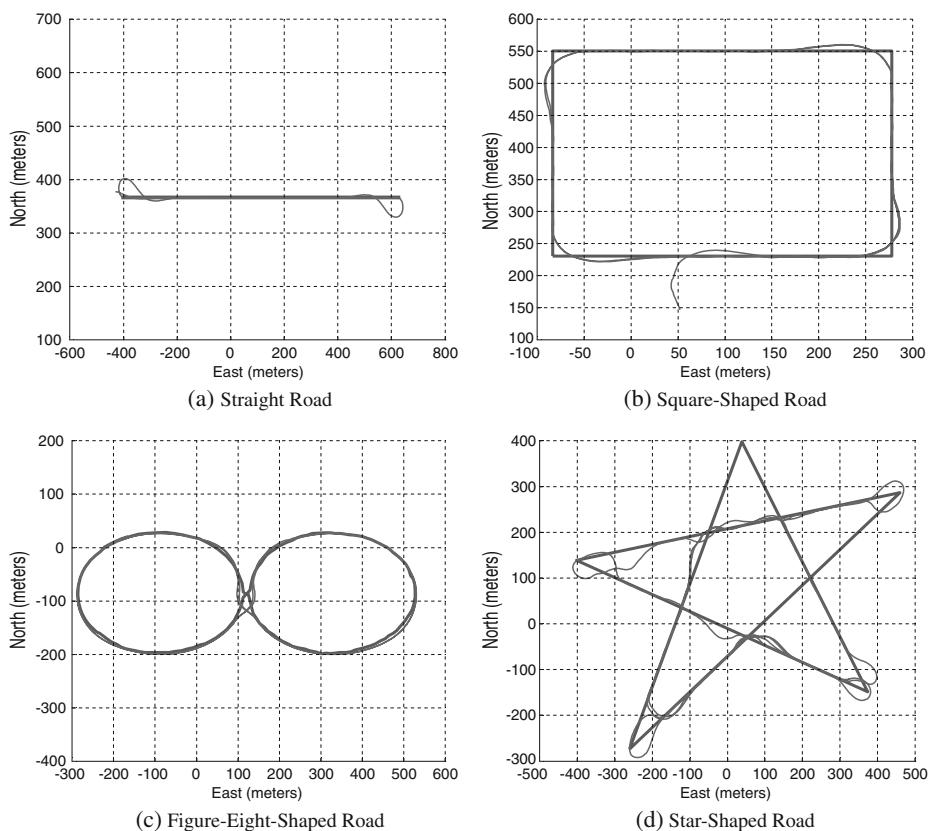
$$-60 \text{ deg} \leq \alpha_{el} \leq 60 \text{ deg}, \quad (56)$$

$$-40 \text{ deg/s} \leq u_1 \leq 40 \text{ deg/s}, \quad (57)$$

$$-60 \text{ deg/s} \leq u_2, u_4 \leq 60 \text{ deg/s}, \quad (58)$$

$$-100 \text{ deg/s} \leq u_3 \leq 100 \text{ deg/s}. \quad (59)$$

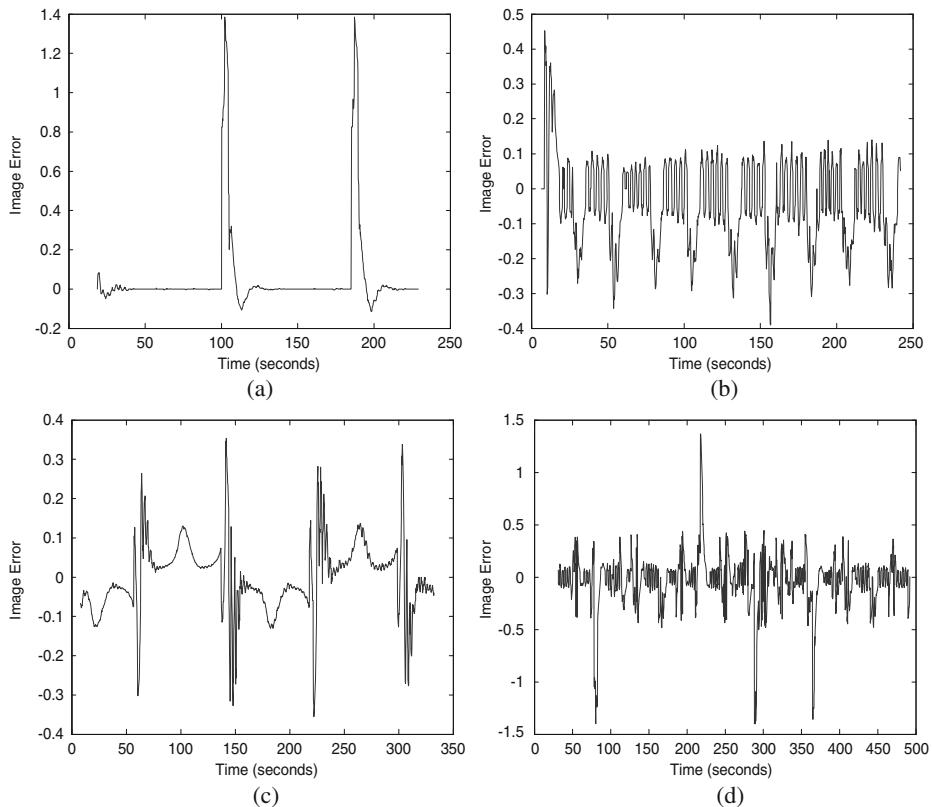
The simulations involved four different scenarios where the road configuration was a straight line, a square, a figure eight, and a star. Each scenario consisted of binary video footage, where road pixels were white and non-road pixels were black. Figure 9 shows the inertial position of the road (blue) for each scenario overlaid by the position of the UAV (red).



**Fig. 9** A plot of the road position used in simulation overlaid by the GPS position of the UAV for four different scenarios, where **a** is a straight road, **b** is a road in the shape of a box, **c** is a road in the shape of a figure eight, and **d** is a road in the shape of a star

In the first three scenarios, the guidance law performed as expected, with the position of the UAV converging over the center of the road. When a change in the road direction occurred, the UAV responded and tracked the change. The error plots are shown in Fig. 10. The large spikes in these plots correspond to sharp changes in the road position and are regulated to zero.

In the star-shaped scenario, the error also goes to zero after transitions in the road. However, the UAV did not track the entire shape like it did in the other scenarios. This is a result of how the image-processing algorithm functions when multiple roads are in the image at the same instance. Sometimes the algorithm switched between which road it selected, causing the UAV to change road segments. However, despite this issue, the guidance algorithm still maneuvered the UAV to whichever road was returned by the image-processing algorithm. In order to allow the desired road to be selected every time, even if the image contains multiple roads, decision-making capabilities must be added to the image-processing algorithm.



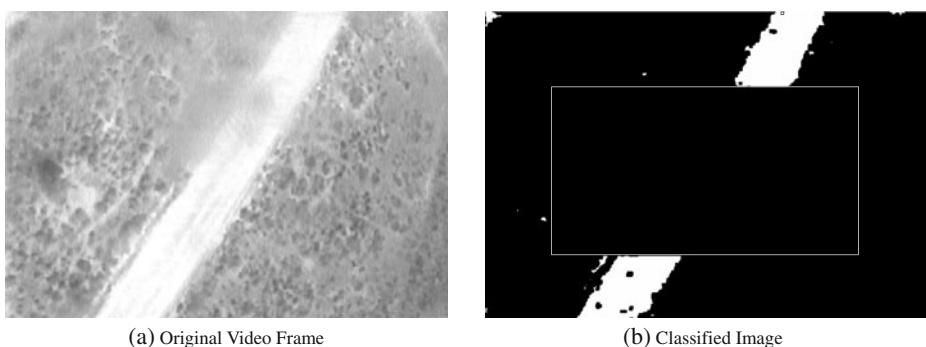
**Fig. 10** The image error for each road scenario. The large spikes correspond to sudden changes in the road, such as a turn or a dead-end. **a** Straight road. **b** Square-shaped road. **c** Figure-eight-shaped road. **d** Star-shaped road

## 6.2 Flight Test Results

To verify the practicality of using the road-following guidance law, flight-tests were performed. The UAV platform used was the fixed-wing design described in Section 5.1, equipped with a pan-and-tilt gimbaled camera oriented as described in Section 3.3. Figure 11 shows the results of the image-processing algorithm, where (a) is the original video frame and (b) is the classified image. Each edge section has been processed in order to illustrate the sections sizes. Note that the middle section has not been classified.

In the flight test, we initialized the algorithm when the UAV was not directly over the road in order to demonstrate how the guidance law will cause the UAV to change its heading. Figure 12 shows an aerial view of the road that was used for the experiment, where the actual GPS location of the UAV is overlaid on top. The UAV started from the red circle and continued in a south-easterly direction over the road until reaching the red triangle.

The image error is shown in Fig. 13. Note that the error converges to zero only for a short time. This deviation can be explained by two disturbances: the lag in

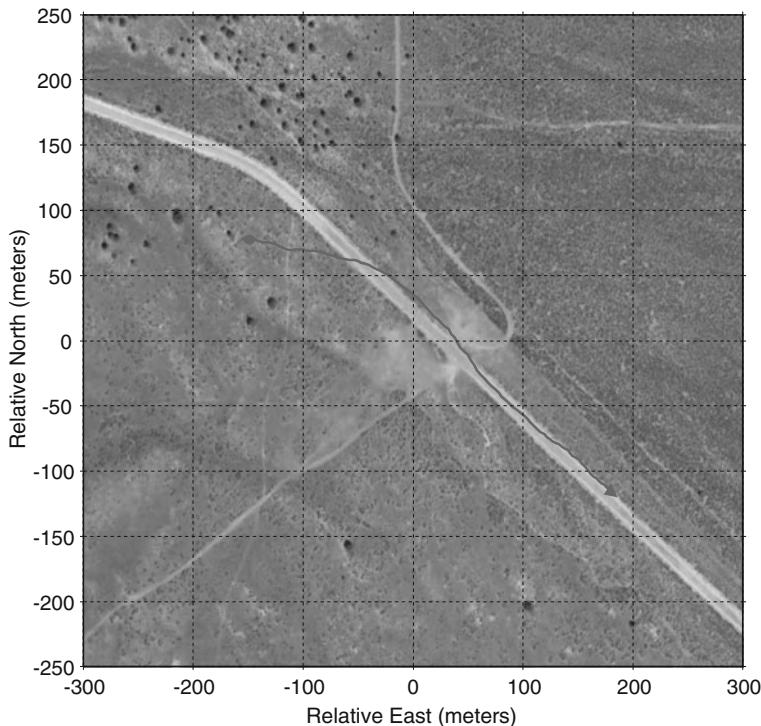


(a) Original Video Frame

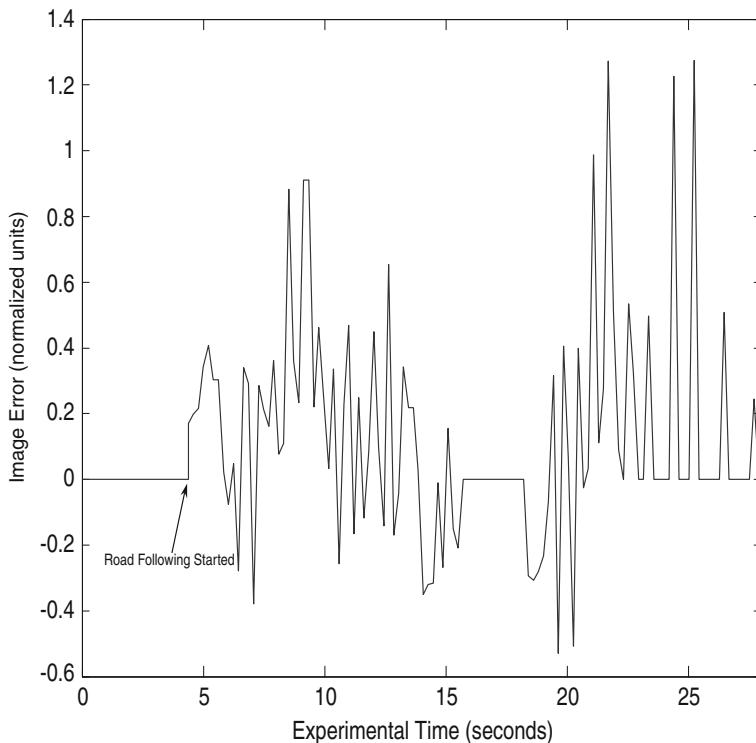
(b) Classified Image

**Fig. 11** A sample from the results of the image-processing algorithm used during the experimental test flights. In **a** is the original image and in **b** is the classified image. The road position determined by the algorithm for this image is indicated by the *red square* at the top of the image. The *green lines* indicate the boundaries of the sections that were processed

the movement of the gimbal and high wind conditions. The lag allows the camera



**Fig. 12** An aerial view of the road used in the flight test with the actual GPS path of the UAV depicted in *red*, starting at the circle and ending at the triangle. The image was obtained from Google Earth



**Fig. 13** The image error recorded during the road-following flight-test

to leave the downward position, causing the road in the image to leave the desired location. During experiments the wind speeds were approximately 50–60% of the vehicle's airspeed, requiring the UAV to compensate by significant rolling motions to counteract the wind. Despite these disturbances, the UAV was still able to track the road.

## 7 Conclusions

In this paper, we have applied proportional navigation to road-following for a UAV using only visual information. We derived the necessary equations and developed a guidance law for both a skid-to-turn model using a strap-down camera and a bank-to-turn model using a gimbaled camera. After performing some analysis, a modification was made to the algorithm to account for road accelerations caused by bends and turns in the road. We showed simulation results that support the feasibility and the effectiveness of the algorithm, as well as the limitations of the image-processing algorithm used. Finally, flight-tests were used to validate the suitability of the algorithm.

## References

1. Casbeer, D.W., Kingston, D.B., Beard, R.W., McLain, T.W., Li, S.M., Mehra, R.: Cooperative forest fire surveillance using a team of small unmanned air vehicles. *Int. J. Syst. Sci.* **37**(6), 351 (2006)
2. Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., Sengupta, R.: Vision-based road-following using a small autonomous aircraft. In: Proceedings of the IEEE Aerospace Conference, vol. 5, pp. 3006–3015 (2004)
3. Rathinam, S., Kim, Z., SoghiKian, A., Sengupta, R.: Vision based following of locally linear structure using an unmanned aerial vehicle. In: Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference, pp. 6085–6090. Seville, Spain (2005)
4. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.: An Invitation to 3-D Vision: From Images to Geometric Models. Springer, New York (2003)
5. Zarchan, P.: Tactical and strategic missile guidance. In: Progress in Astronautics and Aeronautics, 4th edn, vol. 199. American Institute of Aeronautics and Astronautics, Reston (2002)
6. Lin, C.F.: Modern Navigation, Guidance, and Control Processing. Prentice Hall, Englewood Cliffs (1991)
7. Siouris, G.M.: Missile Guidance and Control Systems. Springer, New York (2004)
8. Bryson, A.E., Ho, Y.C.: Applied Optimal Control. Blaisdell, Waltham (1969)
9. Geulman, M.: Proportional navigation with a maneuvering target. *IEEE Trans. Aerosp. Electron. Syst.* **8**, 364 (1972)
10. Geulman, M., Idan, M., Golan, O.M.: Three-dimensional minimum energy guidance. *IEEE Trans. Aerosp. Electron. Syst.* **31**, 835 (1995)
11. Egbert, J.W., Beard, R.W.: Low altitude road following constraints using strap-down EO cameras on miniature air vehicles. In: Proceedings of the American Control Conference (2007)
12. Silveira, G.F., Carvalho, J.R.H., Madrid, M.K., Rives, P., Bueno, S.S.: A fast vision-based road following strategy applied to the control of aerial robots. In: Proceedings of the IEEE XIV Brazilian Symposium on Computer Graphics and Image Processing, pp. 226–231 (2001)
13. Broggi, A.: An image reorganization procedure for automotive road following systems. In: Proceedings of the IEEE International Conference on Image Processing, vol. 3, pp. 532–535 (1995)
14. Jain, A.K.: Fundamentals of digital image processing. In: Chap. Image Analysis and Computer Vision, pp. 407–414. Prentice-Hall, Upper Saddle River (1989)
15. Beard, R.W., Curtis, J.W., Eilders, M., Evers, J., Cloutier, J.R.: Vision aided proportional navigation for micro air vehicles. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference. Paper number AIAA-2007-6609. American Institute of Aeronautics and Astronautics, Hilton Head (2007)
16. Mehrandezh, M., Sela, N.M., Fenton, R.G., Benhabib, B.: Three-dimensional minimum energy guidance. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans* **30**(3), 238 (2000)
17. Stevens, B.L., Lewis, F.L.: Aircraft Control and Simulation, 2nd ed. Wiley, New York (2003)

# A Vision-Based Automatic Landing Method for Fixed-Wing UAVs

Sungsik Huh · David Hyunchul Shim

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 23 October 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** In this paper, a vision-based landing system for small-size fixed-wing unmanned aerial vehicles (UAVs) is presented. Since a single GPS without a differential correction typically provide position accuracy of at most a few meters, an airplane equipped with a single GPS only is not guaranteed to land at a designated location with a sufficient accuracy. Therefore, a visual servoing algorithm is proposed to improve the accuracy of landing. In this scheme, the airplane is controlled to fly into the visual marker by directly feeding back the pitch and yaw deviation angles sensed by the forward-looking camera during the terminal landing phase. The visual marker is a monotone hemispherical airbag, which serves as the arresting device while providing a strong and yet passive visual cue for the vision system. The airbag is detected by using color- and moment-based target detection methods. The proposed idea was tested in a series of experiments using a blended wing-body airplane and proven to be viable for landing of small fixed-wing UAVs.

**Keywords** Landing dome · Fixed-wing UAVs · Visual servoing · Automatic landing

## 1 Introduction

It is well known that the landing is the most accident-prone stage for both manned and unmanned airplanes since it is a delicate process of dissipating the large amount of kinetic and potential energy of the airplane in the presence of various dynamic and operational constraints. Therefore, commercial airliners heavily rely on instrument landing system (ILS) wherever available. As for UAVs, a safe and sound retrieval of

---

S. Huh (✉) · D. H. Shim  
Department of Aerospace Engineering, KAIST, Daejeon, South Korea  
e-mail: hs2@kaist.ac.kr  
URL: <http://unmanned.kaist.ac.kr>

D. H. Shim  
e-mail: hcshim@kaist.ac.kr

the airframe is still a significant concern. Typically, UAVs land manually by either internal or external human pilots on a conventional runway or arresting mechanisms. For manual landing, the pilot obtains visual cue by naked eye or through the live images taken by the onboard camera. Piloting outside of the vehicle needs a lot of practice due to the limited situation awareness. As a consequence, a significant portion of mishaps happen during the landing phase. Nearly 50% of fixed-wing UAVs such as Hunter and Pioneer operated by the US military suffer accidents during landing. As for Pioneer UAVs, almost 70% of mishaps occur during landing due to human factors [1, 2]. Therefore, it has been desired to automate the landing process of UAVs in order to reduce the number of accidents while alleviating the pilots' load.

There are a number of automatic landing systems currently in service. Global Hawk relies on a high-precision differential GPS during take-off and landing [3]. Sierra Nevada Corporation's UCARS or TALS<sup>1</sup> are externally located aiding systems consisting of a tracking radar and communication radios. It has been successfully used for landing of many military fixed-wing and helicopter UAVs such as Hunter or Fire Scout on runways or even on a ship deck. The information including the relative position or altitude of the inbound aircraft is measured by the ground tracking radar and relayed back for automated landing. Some UAVs are retrieved in a confined space using a special net, which arrests the vehicle flown manually by the external pilot. Scan Eagle is retrieved by a special arresting cable attached to the tall boom, to which the vehicle is precisely guided by a differential GPS.<sup>2</sup> These external aids listed above rely on special equipment and radio communication, which are not always available or applicable due to complexity, cost, or limits from operating environment. Therefore, automatic landing systems that are inexpensive, passive, and reliable are highly desired.

Vision-based landing has been found attractive since it is passive and does not require any special equipment other than a camera and a vision processing unit onboard. A vision-enabled landing system will detect the runway or other visual markers and guide the vehicle to the touchdown point. There are a range of previous works, theoretical or experimental, for fixed-wing and helicopter UAVs [4–6]. Notably, Barber et al. [7] proposed a vision-based landing for small fixed-wing UAVs, where visual marker is used to generate the roll and pitch commands to the flight controller.

In this paper, a BWB-shaped fixed-wing UAV is controlled to land automatically by using vision algorithms and relatively simple landing aids. The proposed method is shown to be viable without resorting to expensive sensors and external devices. The overall approach, its component technologies, and the landing aids including the landing dome and recovery net are described in Section 2. The vision algorithms including detection and visual servoing algorithm which enables airplane to land automatically are described in Section 3. Finally, the experiment results of vision-based landing test are shown to validate the algorithm in Section 4. In Section 5, conclusion and closing remarks are given.

<sup>1</sup><http://www.sncorp.com>

<sup>2</sup><http://www.insitu.com/scaneagle>

## 2 System Description

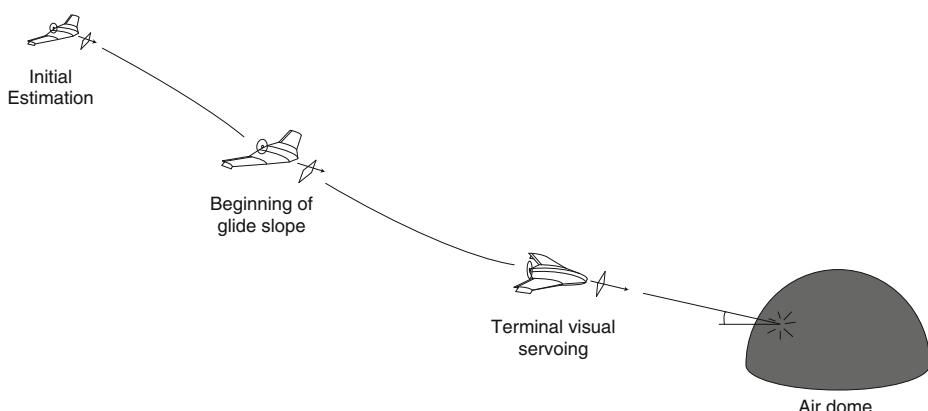
The proposed landing system consists of three major components: an inflated dome as a visual marker, a vision processing unit, and a flight controller using a visual servoing algorithm. In order to develop a low-cost landing system without expensive sensors or special supporting systems, the vision system is integrated with an MEMS-based inertial measurement unit (IMU) and a single GPS. Without a differential correction, the positioning accuracy of GPS is known to be a few meters horizontally and twice as bad vertically. Therefore, even if the position of the dome is accurately known, it will be still difficult to hit the dome consistently with the navigation system that has position error larger than the size of the landing dome.

For landing, with roughly estimated location of the dome, the airplane starts flying along a glide slope leading to the estimated position of the dome. When the vision system locks on the dome, the flight control switches from glide slope tracking to direct visual servoing, where the offset of the dome from the center of the image taken from the onboard front-looking camera is used as the error signals for the pitch and yaw control loops. In the following, detailed descriptions on the proposed approach are presented (Fig. 1).

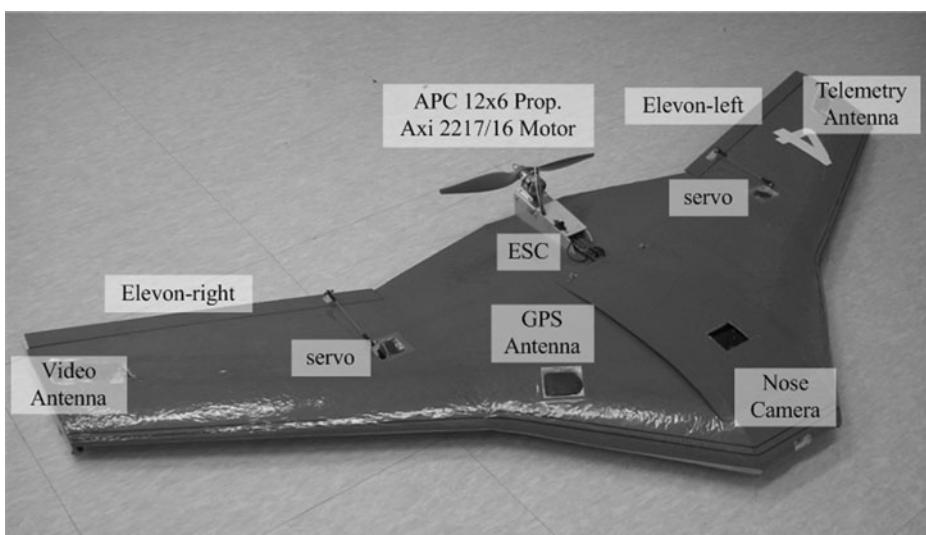
### 2.1 Airframe and Avionics

The platform used in this research is a BWB-based UAV (Fig. 2). BWBs are known to carry a substantially large payload per airframe weight due to the extra lift generated by the airfoil-shaped fuselage. Our BWB testbed is constructed with sturdy expanded polypropylene (EPP), which is reinforced with carbon fiber spars embedded at strategic locations. The vehicle is resilient to shock and crash, a highly welcomed trait for the landing experiments in this paper.

This BWB has only two control surfaces at the trailing edge of the wing, known as elevons, which serve as both aileron and elevator. At the wingtips, winglets are installed pointing down to improve the directional stability. The vehicle has a DC brushless motor mounted at the trailing edge of the airframe, powered by Lithium-polymer cells. The same battery powers the avionics to lower the overall weight. The



**Fig. 1** Proposed dome-assisted landing procedure of a UAV



**Fig. 2** The KAIST BWB (blended wing-body) UAV testbed

large payload compartment ( $30 \times 15 \times 7.5$  cm) in the middle of the fuselage houses the flight computer, IMU, battery, and radio receiver. These parts are all off the shelf products.

The avionics of KAIST BWB UAV consists of a flight control computer (FCC), an inertial measurement unit (IMU), a GPS, and a servo controller module. Flight control computer is constructed using PC104-compatible boards due to their industry-grade reliability, compactness and expandability. The navigation system is built around a GPS-aided INS. U-blox Supersense 5 GPS is used for its excellent tracking capability even when the signal quality is low. Inertial Science's MEMS-IMU showed a robust performance during the flight. The vehicle communicates with the ground station through a high-bandwidth telemetry link at 900 MHz to reduce the communication latency (Table 1, Fig. 3).

The onboard vision system consists of a small forward-looking camera mounted at the nose of the airframe, a video overlay board, and a 2.4 GHz analog video transmitter with a dipole antenna mounted at a wingtip. Due to the payload limit, the image processing is currently done at the ground station, which also serves as the monitoring and command station. Therefore the image processing and vision algorithm results are transmitted back to the airplane over the telemetry link. Image processing and vision algorithms are processed on a laptop computer with Pentium Core2 CPU 2 GHz processor and 2 GB memory. Vision algorithm is written in MS Visual C++ using OpenCV library.<sup>3</sup> It takes 240 ms to process the vision algorithm at an image, then, the visual servoing command is transmitted back to the vehicle at 4 Hz, which is shown to be sufficient for the visual servoing problem considered in this research.

<sup>3</sup>Intel Open Source Computer Vision Library

**Table 1** Specification of BWB testbed

Base platform	StingRay 60 by <i>Haak Works</i> <sup>a</sup> Blended wing-body (BWB)
Dimensions	Wing span: 1.52 m(W) Wing area: 0.52 m <sup>2</sup> Aspect ratio: 4.43
Weight	2.6 kg (fully instrumented)
Powerplant	Axi 2217/16 DC brushless motor 12 × 6 APC propeller Hacker speed controller X-40-SB-Pro Lithium-ion-polymer (3 S: 11.1 V 5,350 mAh)
Operation time	Over 10 min
Avionics	Navigation: single GPS-aided INS GPS: U-Blox Supersense 5 IMU: inertial science MEMS IMU Differential/absolute pressure gauges Flight computer: PC104 Pentium-M 400 MHz Communication: 900 MHz Ethernet Bridge Color CCD camera (70 deg FOV) 2.4 GHz analog transmitter Frame grabber: USB 2.0 analog video capture kit
Vision system	
Operation	Catapult-launch, body landing
Autonomy	Speed, altitude, heading, altitude hold/command Waypoint navigation Automatic landing

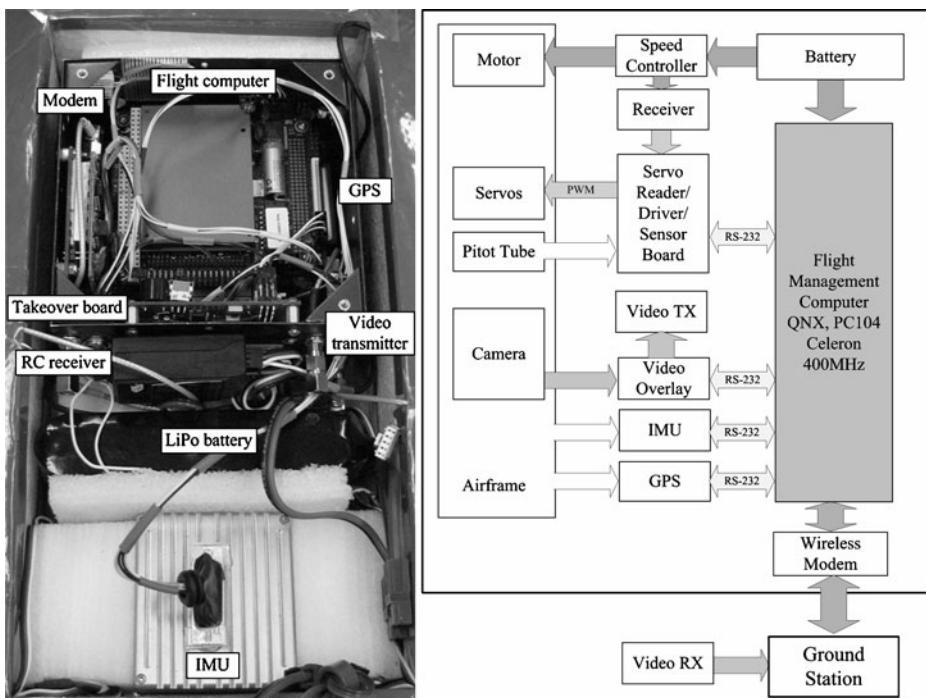
<sup>a</sup><http://www.haakworks.com>

## 2.2 Landing Equipment

In order to land an airplane in a confined space, an arresting device is needed to absorb the kinetic energy in a relatively short time and distance. In this research, we propose a dome-shaped airbag, which offers many advantages for automatic landing scenarios.

The dome is constructed with sturdy nylon of a vivid red color, which provides a strong visual cue even at a long distance. It also absorbs the shock during the landing and therefore the vehicle would not need to make any special maneuver but simply fly into the dome at a low speed with a reasonable incident angle. The vision system runs a fast and reliable color tracking algorithm, suitable for high-rate visual servoing under various light conditions.

For operation, the bag is inflated within a few minutes by a portable electric blower or possibly by a gas-generating charge and transported to anywhere in a compact package after deflated. Using the hooks sewn around the bottom patch, it can be secured to the ground using pegs. Its dome shape allow landing from any direction without reinstallation while conventional arresting nets have to be installed to face the wind to avoid crosswind landing. Since this approach is vision-based, the landing dome is not equipped with any instruments, signal sources, or communication devices. Therefore it does not consume any power or emits energy that can be detected by adversaries.



**Fig. 3** Avionics and hardware architecture

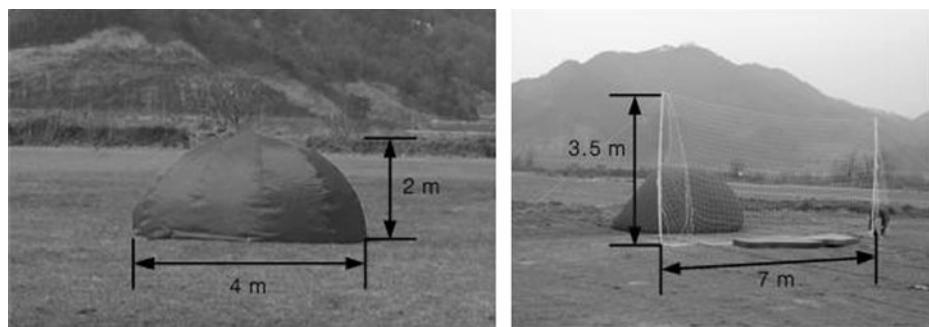
In addition, a recovery net can be used to capture larger vehicles. In this case, the landing dome serves only as a visual cue and the recovery net is responsible for arresting the vehicle.

### 3 Vision Algorithms

The proposed vision algorithm consists of a color-based detection, a moment-based detection, and visual servoing. The color- and moment-based methods perform in a complementary manner to detect the landing dome more reliably. In the terminal stage of landing, the vehicle maneuvers by the visual servoing command only (Fig. 4).

#### 3.1 Color-Based Target Detection

The dome's color is the strongest visual cue that distinguishes the target from other objects in the background. In RGB coding, a pixel at image coordinate  $(x, y)$  has three integers,  $(I_R, I_G, I_B)$ , varying from 0 to 255, respectively. As the pixels that belong to the red dome vary quite large depending on lighting condition, shadow, color balance, or noise, a filtering rule is needed to determine whether a pixel



**Fig. 4** Proposed landing dome and a recovery net

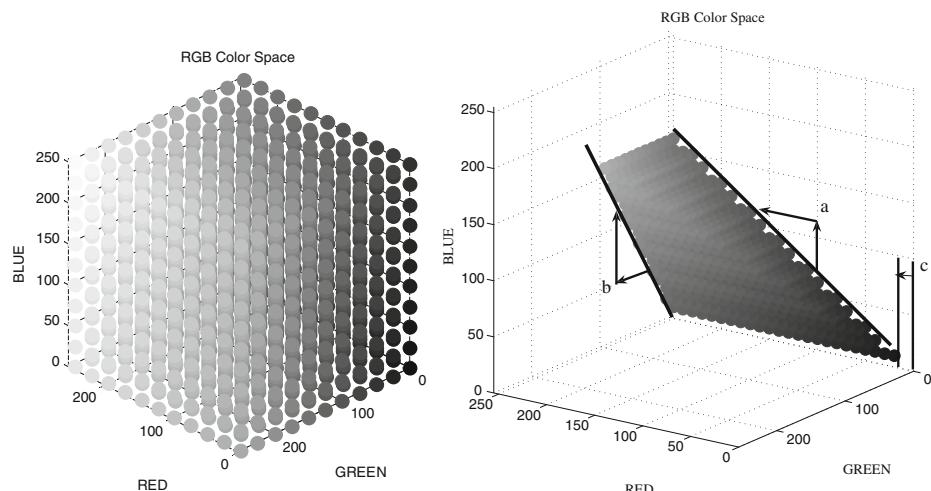
belongs to the dome or not. Based on many images of the dome taken under various conditions, the following filtering rule is proposed:

$$\begin{aligned} aI_B(x, y) < I_R(x, y) \leq 255 \\ bI_G(x, y) < I_B(x, y) \leq 255 \\ 0 \leq c < I_R(x, y) \end{aligned} \quad (1)$$

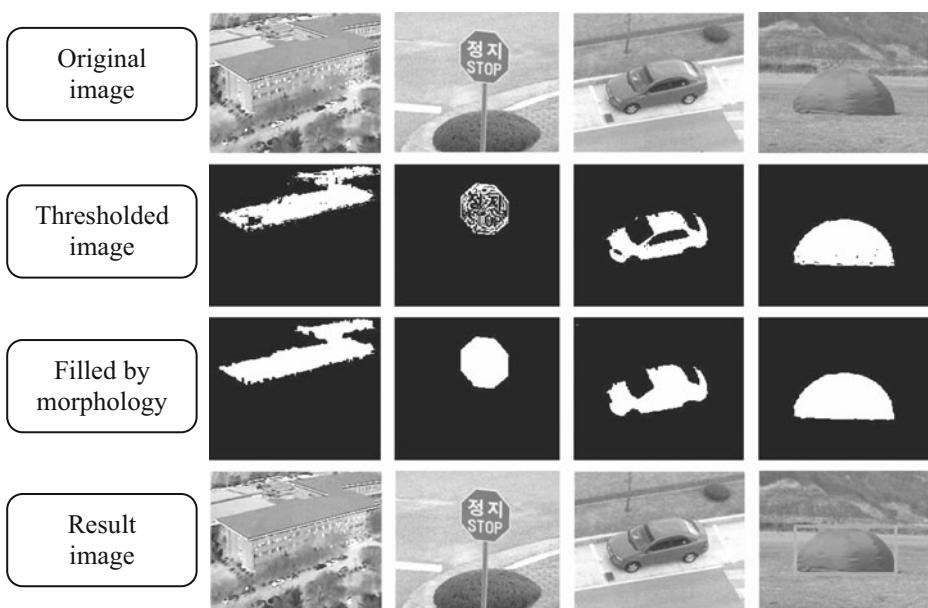
where  $(a, b, c) = (1.5, 1.5, 20)$ . The threshold levels are chosen rather inclusively for detection of wider color range. Figure 5 visualizes the colors that are qualified as a dome using Eq. 1.

### 3.2 Moment-Based Target Detection

When multiple red objects are found by the color-based detection described above in a given image, another classification is needed to determine whether an object is



**Fig. 5** Full RGB color space (*left*) and selected region by filtering rule (*right*)



**Fig. 6** Moment-based image processing procedure

indeed the landing dome or not. For this, an image moment-based filtering is devised. Among many algorithms such as template matching or image moment comparison, Hu's method [8, 9] is used since it is computationally efficient for high-rate detection needed for vision-based landing (Table 1).

Hu's moment consists of seven moment invariants derived from the first three central moments. In a gray image with pixel intensities  $I(x, y)$ , raw image moments  $M_{ij}$  are calculated by

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y). \quad (2)$$

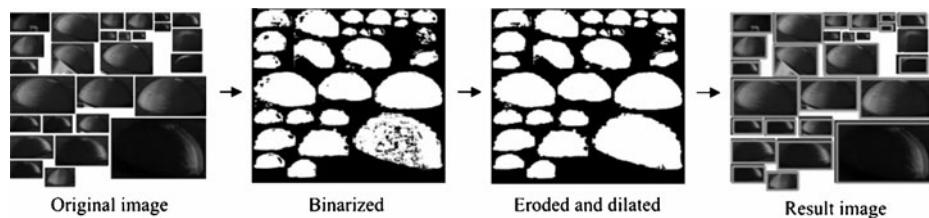
Then simple image properties derived by raw moments include the area of image  $M_{00}$ , and its centroid  $\{\bar{x}, \bar{y}\} = \{M_{10}/M_{00}, M_{01}/M_{00}\}$ .

Central moments for digital image are defined as below

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \quad (3)$$

**Table 2** Hu's moment value of various shapes in Fig. 6

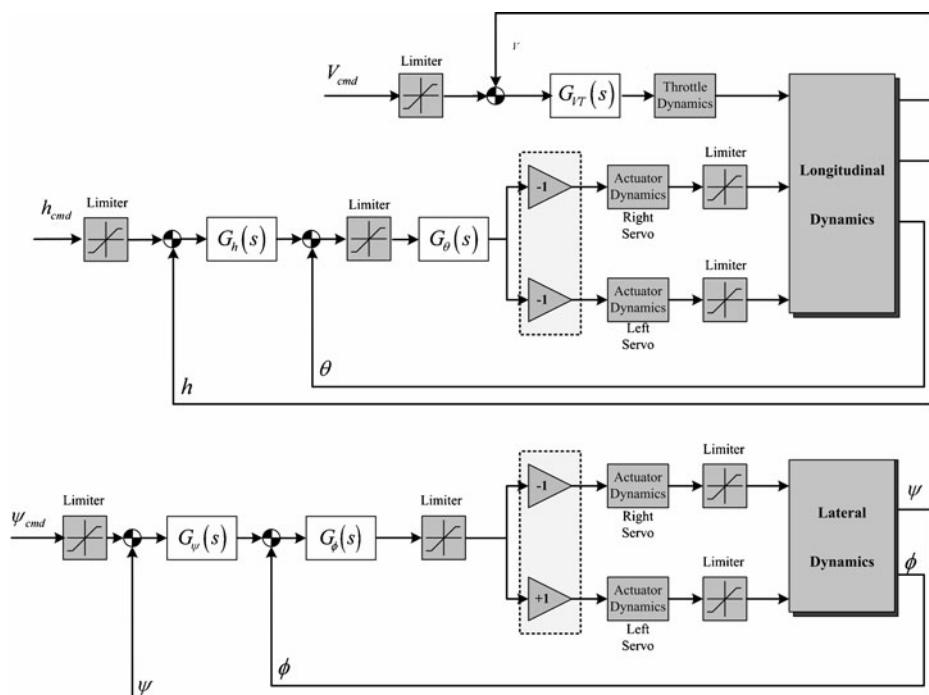
	Roof	Road sign	Automobile	Landing dome
Hu's moment 1	0.48560	0.20463	0.27423	0.21429
Hu's moment 2	0.18327	0.00071	0.04572	0.01806
Hu's moment 3	0.01701	0.00009	0.00193	0.00106
Hu's moment 4	0.00911	0.00028	0.00098	0.00007



**Fig. 7** Application of Hu's method to domes seen from various viewpoints

and the central moments of order up to the third order are given as

$$\begin{aligned}
 \mu_{01} &= \mu_{10} = 0 \\
 \mu_{11} &= M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10} \\
 \mu_{20} &= M_{20} - \bar{x}M_{10} \\
 \mu_{02} &= M_{02} - \bar{y}M_{01} \\
 \mu_{21} &= M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2M_{01} \\
 \mu_{12} &= M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2M_{10} \\
 \mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2M_{10} \\
 \mu_{03} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2M_{01}
 \end{aligned} \tag{4}$$



**Fig. 8** Autopilot loops for longitudinal (top) and lateral dynamics (bottom) of airplane

In order to make the moments invariant to both translation and changes in scale, the following formula is used to divide the corresponding central moment by the properly scaled moment.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(i+j+2)/2}} \quad (5)$$

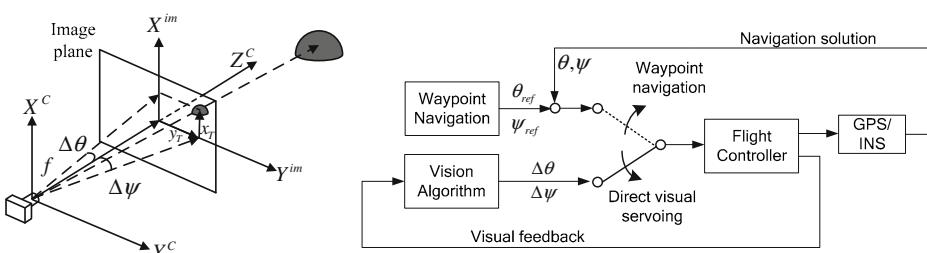
Since the dome is projected to be somewhere between a semicircle to a full circle, its Hu's moments would not vary too much regardless of the distance and the viewing angle. Therefore Hu's moment can be a good invariant characterization of a target shape. Using the second- and third-order moments given in Eq. 5, the following Hu's image moments that are invariant to translation, rotation, and scale are derived:

$$\begin{aligned}\phi_1 &= \eta_{20} + \eta_{02} \\ \phi_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\ \phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\ \phi_4 &= (\eta_{30} - \eta_{12})^2 + (\eta_{21} + \eta_{03})^2\end{aligned} \quad (6)$$

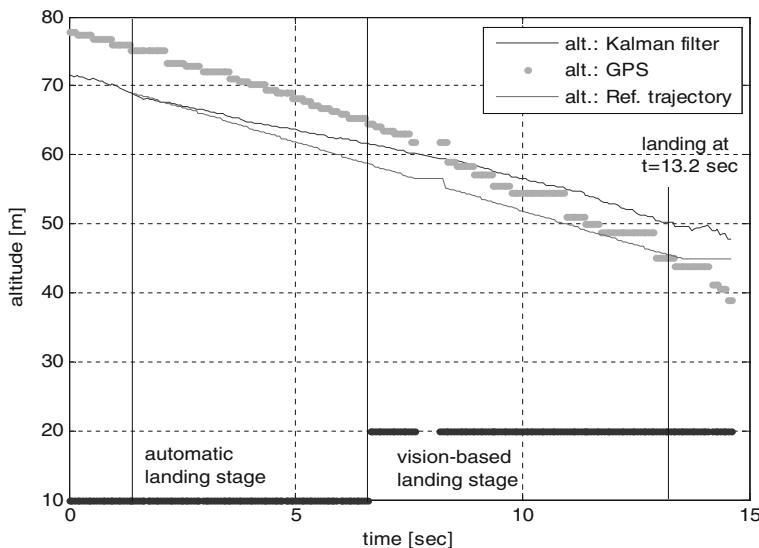
In this paper, only four image moments are used for detection.

Hu's moments approach is applied to various red objects seen from air as shown in Fig. 6 and their moment values are listed in Table 2. The rooftop and the automobile in Fig. 6 has relatively larger Hu's moments while the road sign and the landing dome have comparable first moments due to their round shapes. The higher-order moments of the road sign has much smaller values than those of landing dome since the road sign has a smaller area for given number of pixels. Although the landing dome seen directly above would look much similar to the road sign, such cases are excluded because the camera is mounted to see only the side of the dome during landing.

In Fig. 7, the image processing results of landing domes seen from various viewpoints. Once a database of Hu's moments of these domes was built, the average, standard deviation, and minimum/maximum values of landing dome can be determined. Based on these values, the blob with the largest area in a given image is finally declared as the landing dome. In case of all candidates are not in the proper range of Hu's moments, the algorithm concludes that a landing dome does not exist in that image.



**Fig. 9** Visual servoing scheme



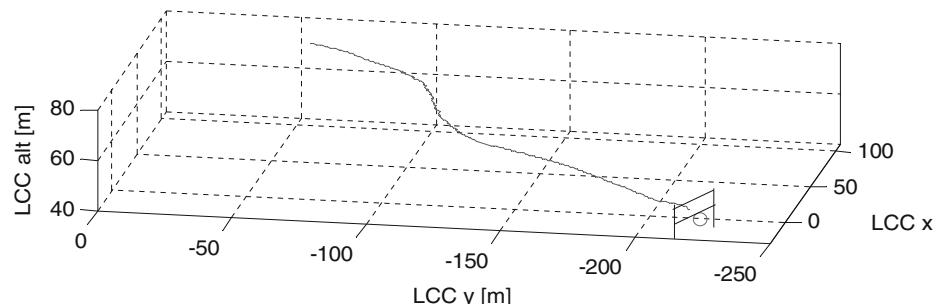
**Fig. 10** Experiment result of vision-based landing: vehicle altitude

### 3.3 Visual Servoing

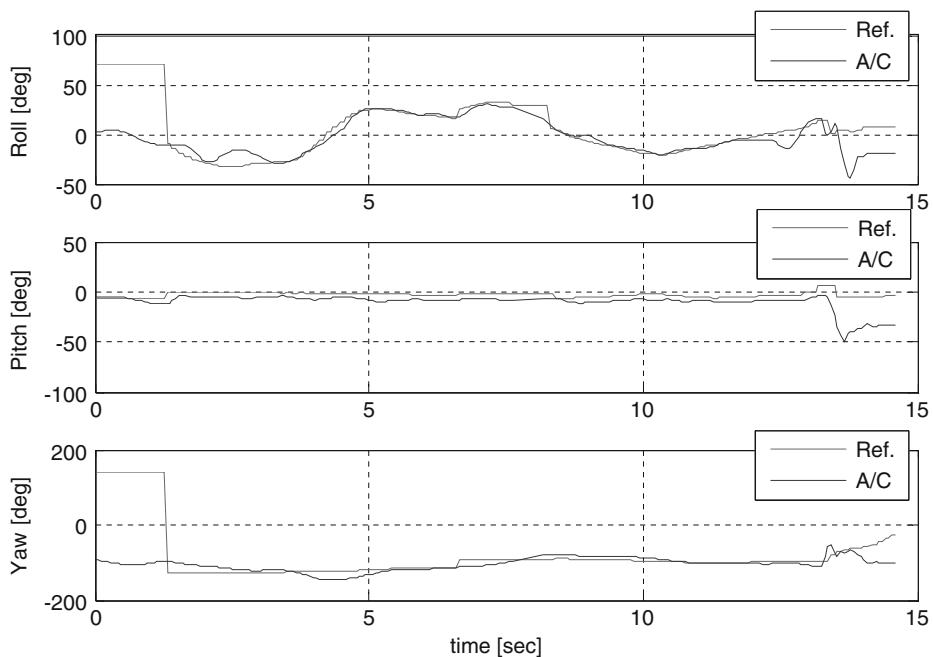
Visual servoing is a method to control a robot using computer vision. The robot is controlled by the error of features between the desired image and current image. The visual servoing method can be applied to the fixed-wing UAV such that the vehicle is controlled by the pitch and roll angle errors estimated from an image as shown in Fig. 8.

As illustrated in Fig. 9, the offset of the landing dome from the center of the image represents the heading and pitch deviation.

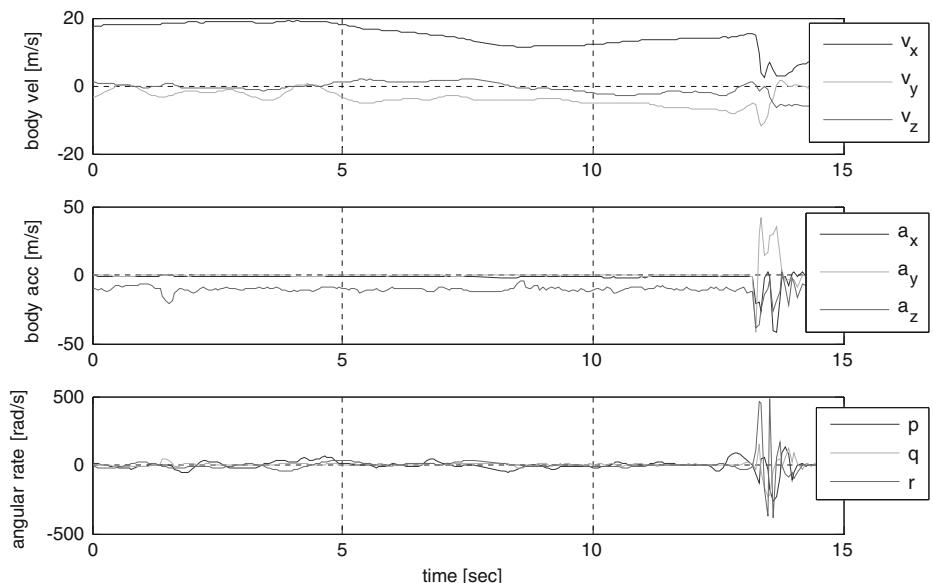
$$\begin{aligned}\Delta\psi &= \arctan(y_T/f) \\ \Delta\theta &= \arctan(x_T/f)\end{aligned}\quad (7)$$



**Fig. 11** Experiment result of vision-based landing: 3D trajectory

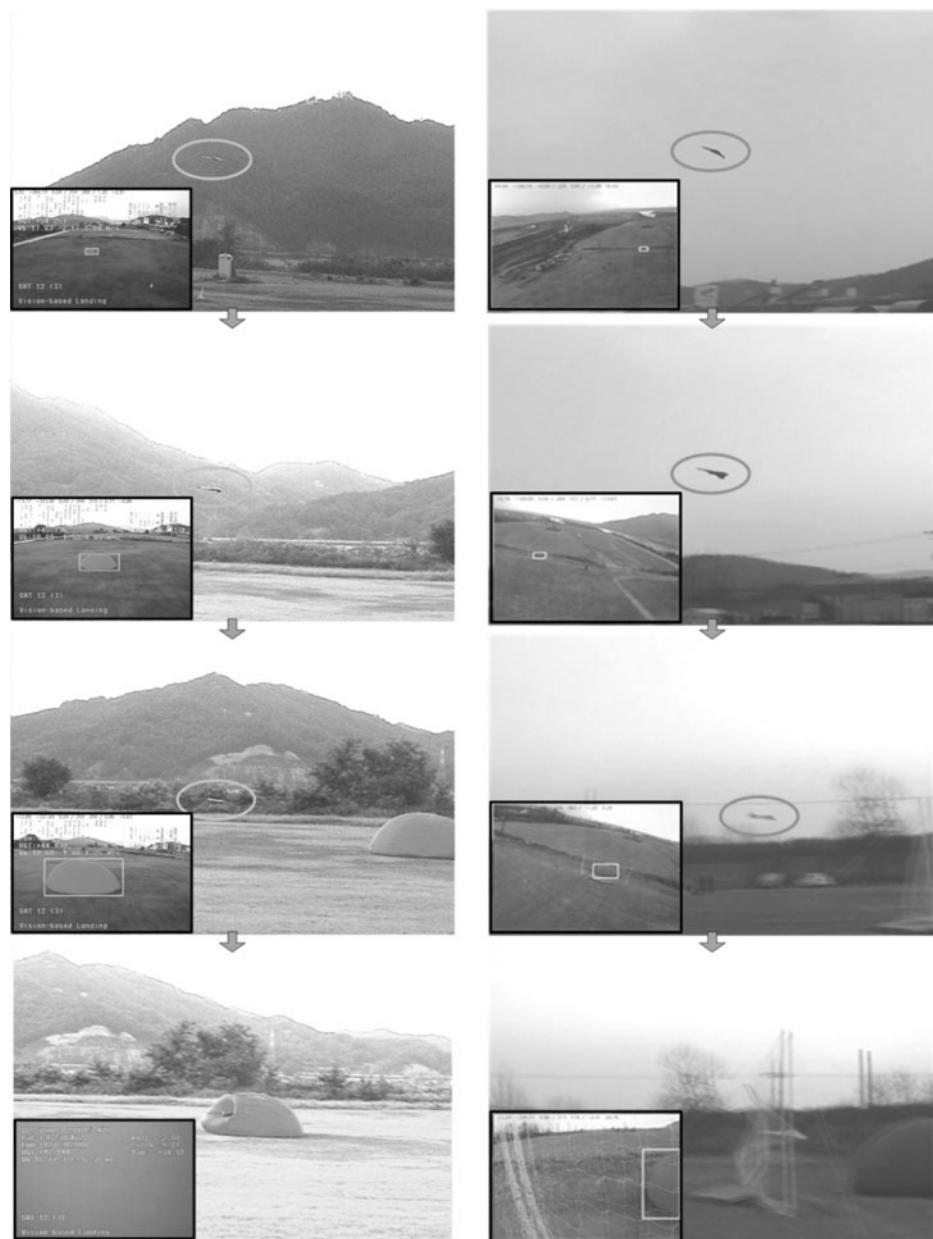


**Fig. 12** Experiment result of vision-based landing: vehicle attitude and heading



**Fig. 13** Experiment result: body velocity, body acceleration and angular rate

When the heading and pitch angle errors are zero, i.e.  $\Delta\psi = \Delta\theta = 0$  (the landing dome appears at the center of the image), the vehicle flies directly to the landing dome.



**Fig. 14** Sequences of images during landing to the dome (*left*) and the recovery net (*right*)

If the camera is mounted at the nose of the airplane so that the  $Z^C$ -axis in Fig. 9 aligns with the direction of flight, any heading and pitch angle deviations indicate that the heading of the vehicle deviates from the target point. Therefore, in order to guide the vehicle to the target point, the controllers should minimize these angles. It can be done simply by sending the heading and pitch angles to the heading and pitch angle controllers. Since the pitch and yaw deviation angles computed from an image are directly sent to the controllers, the INS output is not utilized at all. This approach is simple and robust to any failure in the INS or GPS. This scheme is surprisingly simple but highly effective as will be shown in the experimental result.

#### 4 Vision-Based Landing Experiment Result

Using the experiment setup described in Section 2, the proposed vision-based landing algorithm with landing aids was tested. First, the inflated landing dome was installed in the test field. After the GPS is locked on and the INS is initialized, the vehicle is launched using a catapult at an airspeed excess of 20 m/s. Once airborne, the vehicle is commanded to look for the landing dome in its view. When the dome is located in an image, it generates a glide slope from its current location to roughly estimated location of the dome. During descent, the aircraft flies by visual servoing, and finally, it lands at the dome or at the recovery net safely.

In Fig. 10, vehicle altitude as the results of flight test is shown. Before  $t = 6.6$  s, the vehicle follows the glide slope leading to the roughly estimated location of the landing dome. During  $t = 6.6\text{--}13.2$ , the vehicle flies in the direct visual servoing mode. At  $t = 13.2$  s, the vehicle contacts with the landing net safely.

The reference altitude and the actual one are plotted in Fig. 10. The altitude of the landing dome was initially estimated to be 45 m according to reference trajectory, but it turned out to be 50 m according to the altitude of Kalman filter at the time of impact. This observation advocates our approach to use direct visual servoing instead of following blindly the initial trajectory, which does contain errors large enough to miss the landing net. Figures 11, 12, and 13 show the vehicle's 3D trajectory and states. In Fig. 14, the images taken from the onboard cameras are shown and the green rectangles in onboard images show that the landing dome was reliably detected during flight. The result also shows that the landing method using both landing dome and recovery net are feasible to recover the airplane safely.

#### 5 Conclusion

In this paper, a vision-based automatic landing method for fixed-wing unmanned aerial vehicle is presented. The dome's distinctive color and shape provide unmistakably strong and yet passive visual cue for the onboard vision system. The vision algorithm detects the landing dome reliably from onboard images using color- and moment-based detection method. Since the onboard navigation sensors based on low-cost sensors cannot provide accurate enough navigation solutions, a direct visual servoing is implemented for precision guidance to the dome and the recovery net. The proposed idea has been validated in a series of experiments using a BWB airplane to show that the proposed system is viable approach for landing.

**Acknowledgement** The authors gratefully acknowledge for the financial support by Korea Ministry of Knowledge and Economy.

## References

1. Williams, K.W.: A summary of unmanned aircraft accident/incident data: human factors implications. U. S. Department of Transportation Report, No. DOT/FAA/AM—04/24 (2004)
2. Manning, S.D., Rash, C.E., LeDuc, P.A., Noback R.K., McKeon, J.: The role of human causal factors in U. S. army unmanned aerial vehicle accidents. USAARL Report No. 2004—11 (2004)
3. Loeggering, G.: Landing dispersion results—global Hawk auto-land system. In: AIAA's 1st Technical Conference and Workshop on Unmanned Aerial Vehicles (2002)
4. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: Proceedings of the IEEE International Conference on Robotics & Automation (2002)
5. Bourquard, O., Chaumette, F.: Visual servoing of an airplane for auto-landing. In: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (2007)
6. Trisiripisal, P., Parks, M.R., Abbott, A.L., Liu, T., Fleming, G.A.: Stereo analysis for vision-based guidance and control of aircraft landing. In: 44th AIAA Aerospace Science Meeting and Exhibit (2006)
7. Barber, B., McLain, T., Edwards, B.: Vision-based landing of fixed-wing miniature air vehicles. In: AIAA Infotech@Aerospace Conference and Exhibit (2007)
8. Hu, M.-K.: Visual pattern recognition by moment invariants. In: IRE Transactions on Information Theory (1962)
9. Flusser, J.: Moment invariants in image analysis. In: Proceedings of World Academy of Science, Engineering, and Technology (2006)

# A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks

**A. Cesetti · E. Frontoni · A. Mancini ·  
P. Zingaretti · S. Longhi**

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 21 October 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** In this paper a vision-based approach for guidance and safe landing of an Unmanned Aerial Vehicle (UAV) is proposed. The UAV is required to navigate from an initial to a final position in a partially known environment. The guidance system allows a remote user to define target areas from a high resolution aerial or satellite image to determine either the waypoints of the navigation trajectory or the landing area. A feature-based image-matching algorithm finds the natural landmarks and gives feedbacks to an onboard, hierarchical, behaviour-based control system for autonomous navigation and landing. Two algorithms for safe landing area detection are also proposed, based on a feature optical flow analysis. The main novelty is in the vision-based architecture, extensively tested on a helicopter, which, in particular, does not require any artificial landmark (e.g., helipad). Results show the appropriateness of the vision-based approach, which is robust to occlusions and light variations.

**Keywords** Vision · Navigation and control · Autonomous navigation and landing

---

A. Cesetti (✉) · E. Frontoni · A. Mancini · P. Zingaretti · S. Longhi  
Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione,  
Università Politecnica delle Marche, Ancona, Italy  
e-mail: cesetti@diiga.univpm.it

E. Frontoni  
e-mail: frontoni@diiga.univpm.it

A. Mancini  
e-mail: mancini@diiga.univpm.it

P. Zingaretti  
e-mail: zinga@diiga.univpm.it

S. Longhi  
e-mail: sauro.longhi@univpm.it

## 1 Introduction

A helicopter is a well-suited air vehicle for a wide variety of operations, ranging from search and rescue (e.g., rescuing stranded individuals or launching life buoys at drowning people) to surveillance and inspection missions (e.g., landmine detection or inspection of towers and transmission lines for corrosion and other defects). All these applications demand dangerous flight patterns in close proximity to the ground or to other objects that can attempt to the pilot safety. Additional hazards derive from operations in dangerous or contaminated areas, e.g., inspection after chemical or nuclear accidents. An unmanned helicopter that operates autonomously or is piloted remotely will eliminate these risks and increase the helicopter's effectiveness. Typical missions of autonomous helicopters require flying at low speeds to follow a path or hovering near an object. Positioning equipments such as Inertial Navigation Systems (INSs) or Global Positioning Systems (GPSs) are well-suited for long range, low precision helicopter flights and fall short for very precise, close proximity flights. Manoeuvring helicopters close to objects requires accurate positioning in relation to the objects. Visual sensing is a rich source of data for this relative feedback.

Unmanned Aerial Vehicles (UAVs) constitute a research field that has been extensively explored in the last decade [1]. In literature a wide range of studies on autonomous helicopters has been reported: modelling and identification, simulation, sensor integration, control design and fault diagnosis [2–5].

The use of computer vision as secondary or primary method for autonomous helicopter control has been discussed frequently in recent years, since classical combination of GPS and INS systems can not sustain autonomous flight in any situation [6, 7]. Several studies have demonstrated the effectiveness of approaches based on motion field estimation [8] and feature tracking [9] for visual odometry.

A global view of the main aspects related to the research field of computer vision for UAVs can be found in [10], along with the results of applying these techniques in several applications. In [11] Caballero et al. propose a work on visual odometry based on geometric homography. They use vision to compute a frame by frame odometry for Simultaneous Localization And Mapping (SLAM). A cooperative vision-based system for detecting and tracking objects of interest on or near the ground is presented in [12].

Vision based methods have been proposed even in the context of autonomous landing management: in [13] Merz et al. utilize inertial sensors combined with a single camera and a specially designed landing pad in order to be independent from GPS; in [14] Daquan and Hongyue estimate all the motion parameters of the aircraft are obtained by exploiting images of an airport runway lighting acquired by the airborne camera. The problem of how to land a helicopter in unknown terrain is tackled in [15]. A combination of feature tracking, motion estimation, and multi-frame planar-parallax obtained by a stereo rig is utilized in order to estimate a digital elevation map of the terrain, allowing the determination of a safe landing area of terrain and map waypoints to a desired landing spot.

In general, we can say that the degree of autonomy that an helicopter can achieve depends on factors such as the ability to solve unexpected critical situations, e.g., loss of GPS signal, and the ability to interact with the environment, e.g., using natural landmarks. A vision-based solution for autonomous waypoint navigation and safe

landing on unstructured terrain represents a strong improvement for both these abilities. Several techniques have been implemented, decoupling the problem of locating and tracking a high contrasted, well known landmark, e.g., a classical helipad that can be easily identified by standard image processing techniques [16–18], from the problem of detecting and avoiding natural obstacles, e.g., steep slopes and rocks on a landing area [19–21]. The dependence on fixed, artificial landmarks and on optimal visibility conditions constitutes a strong limit for visual-based navigation in real-environment applications. In some works vision approaches based on moment descriptors are used; they impose no constraints on the design of the landing pad except that it should lie on a two dimensional plane, but artificial landmarks composed by polygons are indispensable [17]. Moreover the weakness of that strategy appears in situations in which natural (e.g., due to debris or leaves) or artificial (e.g., due to engine smoke) occlusions can make the computation of moment descriptors very difficult.

In this paper a vision based approach for guidance and landing of an UAV is proposed. Differently from previous works in this field this paper presents a vision-based system for guidance and safe autonomous landing of a helicopter based on the concept of reuse of local features extracted from the vision system. The vision based architecture, better described in Section 3, is novel and even if presented here in the case of helicopter navigation and landing, can be applied to any UAV, even fixed wing vehicles. The general idea is to guide the UAV using natural landmarks only, which can be selected from aerial images of the operating area. Such images can be already available before or just acquired during the flight. The features extracted from the natural landmarks can be chosen as navigation targets allowing a vision-based feature tracker to compute the references for the flight control.

The main aim of such approach is to provide a remote operator with an intuitive and direct control system, exploiting the benefits deriving from the use of visual information, which constitutes the friendliest interface for an operator to interact with the environment. Through the video stream the operator is able not only to supervise but even to directly lead the operations, assigning the desired navigation targets. Manoeuvres are then managed by the on board control system.

The main advantage of this approach consists in computing the displacement relatively to the target rather than through a coordinate system fixed to the earth. This approach becomes particularly useful in all those situations in which GPS signal is not available (e.g., when operating in urban like environments or in case of fault) or target coordinates are unknown. Moving the UAV using the visual feedback that the operator receives from the on-board camera can be really useful when the path can not be planned before, as during search and rescue or investigation missions.

In the case of landing on unstructured areas the ground is analyzed to verify if the area is flat and proper for landing, since Digital Elevation Map (DEM) resolution can not guarantee the possibility to identify a flat and safe target area. The landing task is managed by the vision system through the localization of the geometrical centre of mass of the matched features. Appropriate feedbacks are given to the autonomous landing control system.

In this paper studies have focused on the problem of natural landmark detection and recognition and safe landing area detection, which, as said before, are the main novelties of the paper; for this reason we focused results on the vision based methods,

leaving a minor space to the validation of the control system. Results show the appropriateness of the vision based approach, which is also robust to occlusions, light variations and scene changes (i.e., reference images grabbed in different days and hours).

The organization of the paper is as follows. In Section 2 the problem of locating, tracking and inspecting an assigned landing area is formulated. In Section 3 the vision approach, a feature-based image-matching algorithm, is presented. Section 4 describes the hierarchical behaviour-based control architecture. Section 5 presents the results achieved. Finally in Section 6 concluding remarks and directions for future improvements are discussed.

## 2 Test-bed and Experimental Task

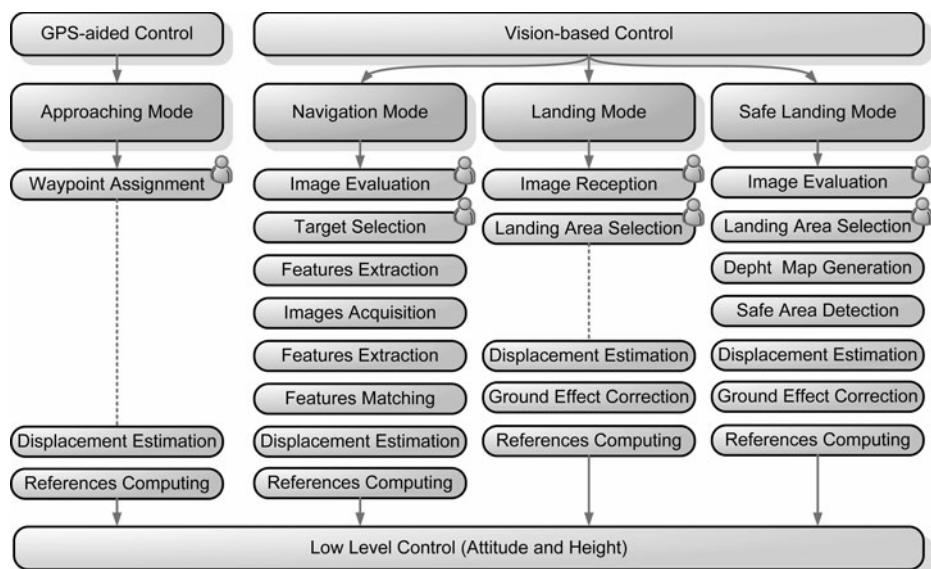
The experimental test-bed HELIBOT is a customized Bergen Twin Observer. It is a twin-cylinder, gasoline-powered radio-controlled model helicopter approximately 1,5 meters in length and capable of lifting approximately 9 kg of payload. Onboard avionics include a PC/104-based computer stack running the QNX RTOS (800 MHz PIII CPU with 256 MB DRAM and 256 MB flash disk), a GPS-41EBF3V receiver with WAAS and EGNOS (available in Europe) corrections and a Microstrain 3DM-GX1 Attitude Heading Reference System (AHRS). The helicopter is also equipped with a downward-pointing Nikon camera. The ground station is mainly constituted by a laptop, used to send high level control commands to the helicopter as well as displaying telemetry from it. Data are sent using an advanced radio modem that transmits and receives on the 868 MHz band. Autonomous flight is achieved using the hierarchical behaviour-based control architecture presented in Section 4.

The testing of algorithms was largely performed using a simulation framework developed by our research group. The framework is provided with a customizable HELIBOT model, allowing an easy switching from simulated to real tasks.

As shown in Fig. 1 four operating modes are proposed for the high level control system. The approach to the operating area is managed with the aid of GPS information. The user gives to the system the waypoint coordinates and the helicopter is autonomously guided to the goal. Then the operating mode is switched and vision system enters into the control loop. The user analyzes a high resolution image of the area and selects a target. The system extracts the appropriate features from the target image to define a natural landmark and starts to search it in the image sequence coming from the onboard camera. Once features are detected and tracked, the system uses the image location of these features to generate image-based velocity references to the low level attitude and height controllers.

In the case of a programmed landing, the vision-based controller pilots the helicopter over the landing area and manages the landing. If the landing area is unknown, an appropriate analysis is performed to select the nearest flat area for a safe landing.

Figure 2 shows the flying area in which experimental tests have been carried out. The vision algorithm is described in the next session.



**Fig. 1** Control system operating modes

### 3 Vision Based Approach

The main difficulty to attain fully autonomous robot navigation outdoors is the fast detection of reliable visual references, and their subsequent characterization as landmarks for immediate and unambiguous recognition. The vision approach presented here following is based on the concept of natural landmark. This system allows a user to control the robot choosing the navigation target in the images received from an on-board camera or from a high resolution aerial or satellite image. This form of navigation control is convenient for exploration purposes or when there



**Fig. 2** The “Gruppo Aeromodellistico Rio Ete” flying area

is no previous map or knowledge of the environment, situations in which systems like GPS, even if available, become useless and can be discarded. The user can decide the next target for the robot and change it as new views of the environment become available. This vision based navigation approach was also used to locate a landing area and to allow autonomous landing by giving feedbacks to the UAV control system.

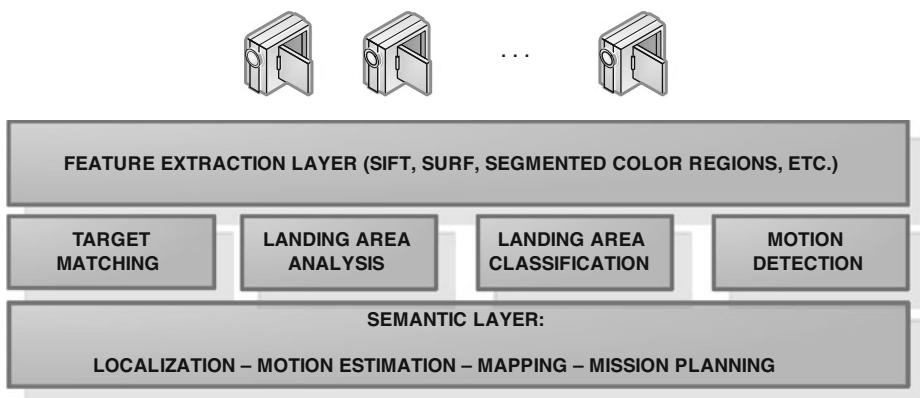
Figure 3 depicts this concept, underlining the key idea that any high level task relies on the same features extracted at the lower layer. This ensures time performances of the whole approach and the chance of choosing different feature extractors, following research developments in computer vision, but not depending on a single approach.

The vision based system performs also a scene analysis to obtain a first rough classification of the safe landing area. All elaborations are based on the same feature set extracted from the current image and elaborated for different purposes: navigation, waypoint identification, landing area detection, safe landing area classification. Processing is based on high level analysis of point features, which in this approach are extracted using SIFT [22], but that could be extracted using any feature extraction algorithm able to detect stable points and to provide a robust descriptor.

### 3.1 Feature Detection for Natural Landmark Tracking

The system should be able to detect natural landmarks in unstructured outdoor environments while the robot is operating. These natural landmarks must provide reliable matching results. To overcome the inherent difficulties with matching images of an object or a scene captured outdoor during motion, the description of landmarks must be based on invariant image features. These features must be distinctive and invariant or partially invariant to translation, rotation, scale, 3D camera view point and outdoor illumination conditions. Furthermore, robustness to image noise, occlusion, background changes and clutter should be granted while maintaining near real-time feature extraction and matching performance.

To solve all the discussed aspects an improved version of the Scale Invariant Feature Transform (SIFT), developed by Lowe [22, 23] is used. SIFT is invariant



**Fig. 3** The vision based architecture

to image translation, scaling and rotation. SIFT features are also partially invariant to illumination changes and affine 3D projection. These features have been widely used in the robot localization field as well as many other computer vision fields. SIFT features are distinctive and invariant features used to robustly describe and match digital image content between different views of a scene. While invariant to scale and rotation, and robust to other image transforms, the SIFT feature description of an image is typically large and slow to compute. Our improvements, better described in [24, 25] consist of two different steps:

1. Divide the image in different sub-images;
2. Adapt SIFT parameters to each sub image and compute SIFT feature extraction, only if it is useful.

During the first step, the image is divided into several sub images according to its resolution. The number of scales to be computed is also decided. The image is doubled at the first step of the SIFT algorithm. During the second step the following threshold value ( $Tr$ ) is computed to define the contrast threshold value of the SIFT algorithm.

$$Tr = k \cdot \frac{\sum_{i,j=0}^{DimX, DimY} |I(x_i, y_j) - \bar{I}(x_i, y_j)|}{DimX \cdot DimY}$$

In the previous expression  $k$  is a scale factor,  $DimX$  and  $DimY$  are the x and y image dimensions,  $I(x,y)$  is the intensity of the gray level image and  $\bar{I}(x, y)$  is the medium intensity value of the processed image. In the Lowe's SIFT implementation the contrast threshold is statically defined and low contrast key points are rejected because they are sensitive to noise. To speed up the feature extraction every sub-image that has a very low threshold value is not computed because of the high probability of not finding features in an almost constant image region (i.e., extreme detection algorithm will fail). Further details of the used approach can be found in [24, 25].

This method is also known to be robust to partial occlusions, such as those caused by engine smoke (Fig. 4) and can obtain time performances comparable with other fast feature extractors; in particular our experiments demonstrated that the system was able to elaborate five frames per second using a  $320 \times 240$  pixel image series.

The matching of a natural landmark in an image is then refined using the computation of the centre of mass  $C$  of a subset of matched features; relying on the assumption that a single landmark is present in each image, we analyze the group of features by calculating the cluster that minimizes the relative distance of the 80% of the total number of features.

This ensure that isolated false matches, usually far from the landmark centre, are deleted in the computation of  $C$ , whose coordinates  $(x_c, y_c)$  are finally used as feedback for the landing control system.

The metric used to compare feature descriptors is a classic Euclidean distance and some heuristics (i.e., discard frames with only few features extracted, etc.).

At the present state of our research we do not perform a real feature tracking, but we simply evaluate feature matching between every frame and the set of way points or landing areas. Here following we often name this methodology landmark detection and tracking.



**Fig. 4** Typical occlusions caused by engine smoke; camera is mounted on the bottom of the UAV and it points the terrain

### 3.2 Optical Flow for Safe Landing Area Classification

Surface reconstruction can be defined as the process of inferring a mesh of interconnected points representing a three-dimensional surface. The surface is often assumed to be rigid and fixed. Computer vision systems ideally would like to be able to reconstruct objects or environments from a sequence of pictures.

In the UAV application presented in this paper we are interested only in depth estimation to determine if the landing area is safe for landing.

To reach this purpose we use two different approaches, based on typical helicopter movements. The first case assumes that images are taken at a constant height and with a pure translational motion; this restriction can be overcome since the control system can assure those flight conditions during landing area inspection. The second approach assumes that the helicopter is approaching the landing phase and is moving maintaining the position and decreasing the altitude to complete a pure vertical landing; also in this case we use the feature-based vision system to inspect the landing area relying on the same set of local features. This motion is quite typical too and can be easily managed by the control system.

In the first case we use an optical flow for depth perception that allows safe autonomous landing. The system is based on feature-based surface reconstruction using an optical flow computation. This approach chooses image features that are stable for large motions. Thus, a sparse set of very confident depth estimates is obtained. If the surface is considered quite stable the area is used for the autonomous landing maneuver.

First, an image sequence of the target landing area is taken by the camera mounted on the UAV and the SIFT features are used to estimate the optical flow in two successive frames; then, structure (in particular depth) information of target landing area relative to UAV is recovered; finally, a simple classifier based on a binary threshold is used to decide if the surface has variable depth or not and, consequently, if it can be used as a landing area.

Experiments using both computer simulated images and real video images demonstrate the correctness and effectiveness of our method. In detail, the optical flow vector field is evaluated using a simplified approach: the optical flow is computed on the base of SIFT features that have been matched in two consecutive frames captured

by the UAV during a translational flight over the area. Translational optical flow is then filtered on the base of the flow vector orientation and, finally, a computation of the variance of the optical flow module is performed to measure surface flatness. A binary threshold is used to define if the surface is suitable or not for safe landing of the UAV. The threshold adopted is now experimentally determined. A specific study is planned to relate it to some parameters to obtain a dynamic estimation.

Feature extraction and matching algorithms as SIFT are useful to estimate the flatness of a surface especially in translation conditions as above introduced. If  $F_k = \{(x_i, y_i, \hat{x}_i, \hat{y}_i)\}, i = 1, \dots, N_k$  is the set of  $N_k$  features at step  $k$  formed by quadruples of matched features, we estimate the flatness of surface applying a simple assumption.

*Assumption 1* If a vehicle is moving with pure translational motion, the flatness of beneath surface is calculated by comparing the variance of distances between two or more sets of matched features; low variance corresponds to safe areas, while great variations are typical of unsafe regions.

We define  $d_k^i = \|(x_i, y_i) - (\hat{x}_i, \hat{y}_i)\|_j$  as the distance (norm  $j$ , with  $j = 2$ ) between two matched features at step  $k$ ; the flatness of “surface” is calculated as:

$$\text{flatness}_k = \text{std} \left( \sum_{n=1}^{N_k} d_k^n \right)$$

where  $\text{std}$  is the standard deviation operator applied to a set of data. The classifier is a binary threshold on the  $\text{flatness}_k$  value, which operates in the following way:

$$\text{flatness}_k = \begin{cases} \text{safe if } < T \\ \text{unsafe if } \geq T \end{cases}$$

The  $T$  threshold is calculated by a supervised training phase, where a human operator classifies a set of well known images as *safe* or *unsafe*; this procedure is simple and requires a little spent of time. Other approaches under development focus on a different calculation of  $T$  using fuzzy classifiers, which allows to represent the uncertainty;  $T$  can be also time-varying, adapting to the number and quality of features extracted.

Some considerations are suitable for clarity. The use of SIFT features is a reasonable proposal since they have been previously extracted and used also for matching. No additional computational load is required. A restrictive hypothesis assumed in the proposed approach concerns the necessity of images taken at a constant height and with a pure translational motion, but this restriction can be overcome since the control system can assure those flight conditions during landing area inspection. The threshold used for classification is now experimentally determined. A specific study is planned to relate it to some parameters in order to obtain a dynamic estimation.

In the second case (vertical landing) we compute a graph of distances between every feature couple and compare, frame by frame, the variation of this array of distances, trying to evaluate the presence of a constant increase of variation, stable over the entire image. This computation is based on the hypothesis that geometric linear zooming over a flat surface brings to constant distance variations. For this

reason we compute the variance of the variation vector as a measure of the previous explained heuristic. High variances bring to the classification of unsafe areas, while very low variance gives us a feedback of the flatness of the landing area. From another point of view this method evaluates a kind of radial movement of features while the helicopter is landing vertically; it is the same case of a zoom aligned with the centre of the image that enlarges every object in the figure. This heuristic evaluator is very quick, even if not exact, due to the possible presence of false feature matches.

An assumption also made in this heuristic surface classification method is the fact that a quite large number of features is extracted almost uniformly from the current helicopter view; the assumption is not restrictive in any of the feature extraction methods here proposed because, usually, they are able to extract hundreds of features.

Here following a short meta code description of the approach is proposed.

### 3.2.1 Slope Detection Algorithm

- search for feature matchings in a pair of consecutive images;
- calculation of distances among features in the first image (D1 matrix);
- calculation of distances among features in the second image (D2 matrix);
- detection of wrong matchings (if  $d_{1,i,j} > d_{2,i,j}$ );
- filtering of inconsistent data;
- calculation of normalized variations in distances among features;
- calculation of mean and the variance of the previous results.

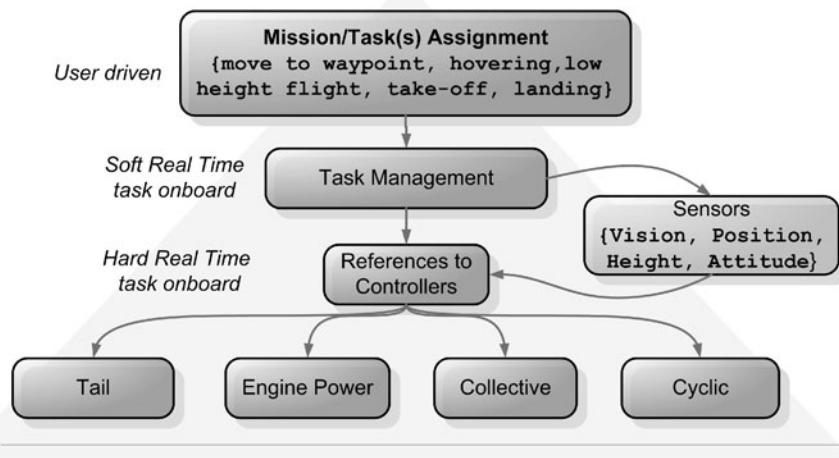
The two methods bring to a correct and conservative classification of the area, choosing only flat surfaces as safe areas. Results, even if preliminary, show the feasibility of the proposed method. Time performances of the algorithm are good because it does not add time consuming evaluations to the feature extraction process.

## 4 Control Strategy

Unmanned Aerial Vehicles, in particular helicopters as HELIBOT, are complex flying machines. The mathematical model of a reduced scale helicopter has been deeply studied during last years [26]; the various proposed approaches require a large set of well-known parameters (e.g., geometrical, aerodynamic, mechanical) that in the real case are estimated or measured [27]. Simulation environments are useful tools that support the design process of control laws and task management [27]. Given an accurate model of system, simulations assure to save time and money, due to the high costs of UAVs. The introduction of a vision sensor that aids the autonomous navigation and landing requires the development of a hierarchical control architecture. The use of a hierarchical control structure is a widely adopted approach in UAV [28–30], that allows to de-couple the two main aspects:

- High-level control, represented by strategy and task management,
- Low-level control, which translates the assigned behaviour to actuator controllers.

Each level is supported by a set of sensors and/or actuators. In Fig. 5 a graphical representation of hierarchical structure implemented in HELIBOT is shown. The

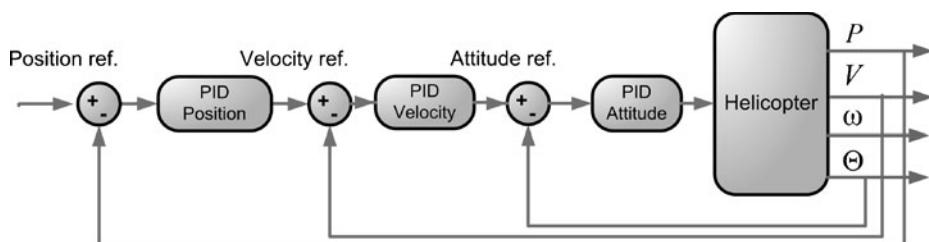


**Fig. 5** The hierarchical structure allows to de-couple problems, solved at different level by specialized modules

vision sensor is used to track features of interest in the field of view. For improving the robustness to loss/degradation of GPS signal, information retrieved by the computer vision algorithms is fused with position data obtained by GPS (with EGNOS corrections) and inertial data by AHRS.

The top level is represented by Mission/Task(s) Assignment module, where the interaction between UAV and human operator is maximum. Task management is a part of Flight Management System (FMS) [27] and generates the path to accomplish a specific task. The last level manages the control of actuators. The complexity of mathematical model of helicopter justifies the use of model-free controllers. A triple nested PID control scheme has been implemented for HELIBOT as shown in Fig. 6.

As stated in the previous section, where the vision-based approach has been presented, to locate the landing area or to explore a zero-knowledge area, common approaches can fail due to non-robust techniques (e.g., scale, rotation, change of light conditions, point of view). The use of landmarks, natural or artificial, gives the Task



**Fig. 6** The nested PID controllers scheme separates the problem of minimizing position error assuring moderate speed when error is high and low speed when the error tends to be small

Management module the capability to control the vehicle even in presence of faults, such as loss/degradation of GPS signal. The question is when and how to switch from the standard behaviour, basically driven by standard sensors, to the vision aided behaviour. The challenge is to ensure the stability during the switching. Simulation stages are useful to predict the dynamics of helicopter during the switch.

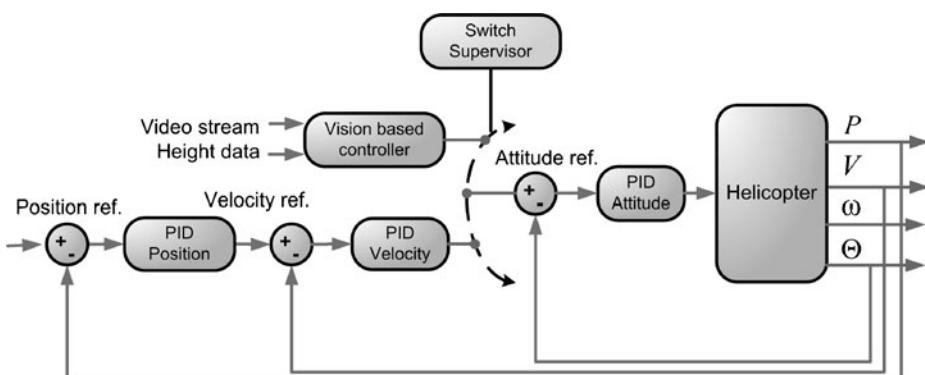
#### 4.1 Switching Control Strategy by Behaviour

Switching control is a common approach when one controller is not sufficient to cover the dynamics of a plant/process. A set of controllers tuned on a particular operational condition often guarantees better performance in terms of precision to external references and robustness to change of parameters. In the considered case, the switch occurs when there is a change of task. Standard controllers used for navigation are not suitable for fine tuning, as required in the landing task. In fact, during navigation high accuracy and precision in terms of position, velocity and attitude are not required, while in a critical task as landing a high accuracy on position is required and the use of dedicated sensors as vision systems and range finders (e.g., laser or sonar) becomes necessary.

In Fig. 7 the proposed switching architecture for attitude is shown. The structure of the tail controller and the engine power is similar, but is not reported here for the sake of brevity. The main difference from the control scheme of Fig. 6 is the introduction of a supervisor and a vision based controller. The supervisor acts switching the controller, taking into account that a switching condition must be satisfied to avoid the loss of stability. References from the vision based controller are imposed without velocity loop because the speed in this phase is low.

*Switching Condition* A switching at time  $k$  can occur if and only if speed (translational and angular) tends to zero.

The condition is a bit conservative, but it is reasonable in the context of landing or low speed feature tracking. The advantage of low speed during transition is the reduced magnitude of bump between old references and new ones. The vision based



**Fig. 7** Switching architecture for fine tuning tasks as landing or low speed features tracking

controller provides fine planar alignment between helicopter and features found in the landing area and height over ground is also regulated. The vision based controller acts as a quasi-proportional controller, as shown in Fig. 8, where a dead zone is introduced to overcome chattering problems induced by small oscillations around the equilibrium. Saturator levels, both for maximum and minimum changes, avoid to impose dangerous references (e.g., high values in collective can cause the overturn of helicopter).

An integral action with saturation can be added to increase the steady state precision in terms of position error. A more complex scheme that makes use of fuzzy rules is under development.

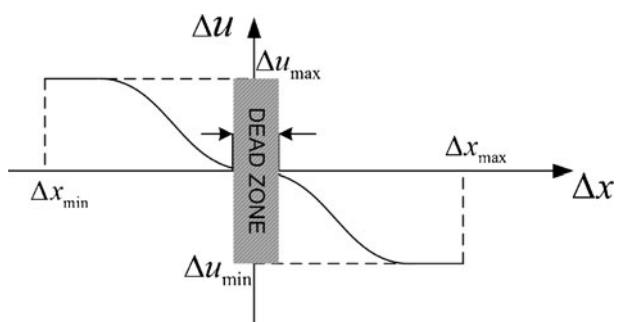
#### 4.2 Ground Effect Estimation during Low Height Flights and Landing Flights

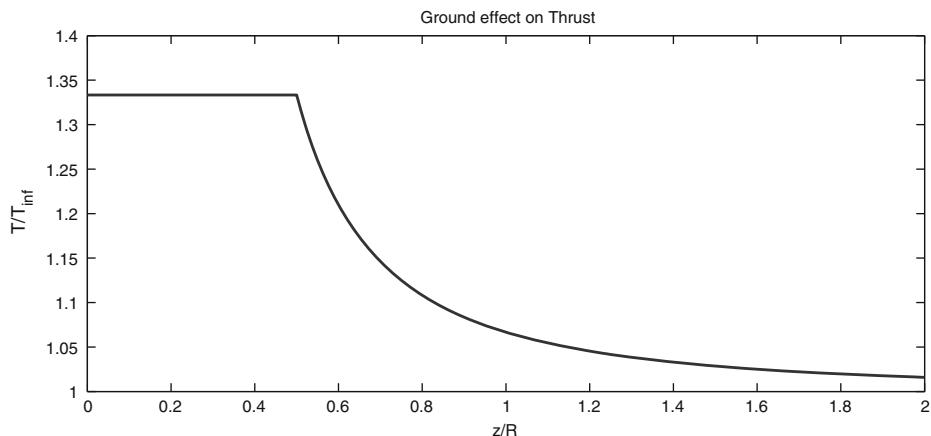
Ground effect is an aerodynamical effect that turns significant when the helicopter flights at altitudes that are comparable with the main rotor span. The main effects related to this phenomenon are a significant increase of lift and a reduction of induced velocity. The problem is well known and deeply investigated especially after First World War, when the military forces understood the tactical importance of helicopters due to the hovering feature of these vehicles.

During the last period many approaches have been proposed to precisely model the unsteady aerodynamics of rotorcraft In Ground Effect (IGE). Several dependences (e.g., blade loading, blade aspect ratio, twist) are weak and not significant for control purposes. Well-known simple models bind the rotor span normalized altitude (height / rotor span) with an increase of thrust or, alternatively, a reduction of induced velocity. In Fig. 9 the behaviour of  $T/T_{inf}$  ratio is reported, where  $T_{inf}$  is the thrust out of the ground effect.

Taking into account the ground effect, the altitude controller is improved introducing the dynamics of thrust when the helicopter is approaching to ground. Following the control scheme previously introduced (Fig. 6), two controllers must be revised: the collective controller and the engine gas controller. They are responsible of varying the helicopter altitude when the attitude is proximal to zero. The gas controller tends to maintain the same main rotor speed  $\Omega_{MR}$  and the collective controller is varied to control the thrust. In the case of IGE, a gain scheduling

**Fig. 8** Transfer function between input from vision algorithms (difference  $\Delta x$  in position along an axis between helicopter and target) and variation  $\Delta u$  of attitude reference





**Fig. 9** The considered ground effect on thrust

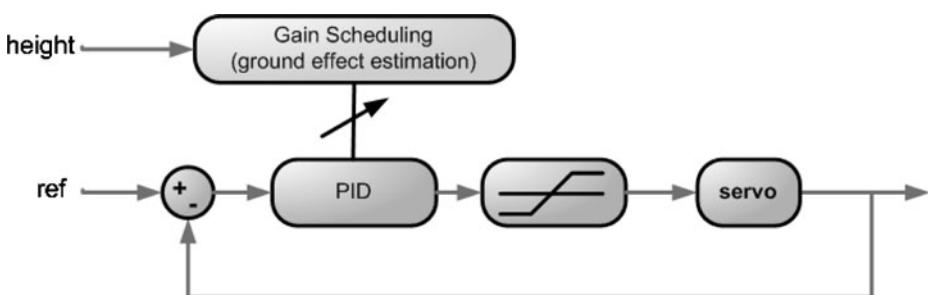
approach is used to tune in real time the controller gain. In this condition  $\Omega_{MR}$  is lightly reduced owing to the increase of lift. Gain is expressed by:

$$\hat{K} = K(\alpha k_G) \quad 0 \leq \alpha \leq 1$$

where  $K$  is the nominal gain value in the case of Out of Ground Effect (OGE),  $\alpha$  a weighting factor and  $k_G$  is related to the variation of lift, calculated at each step by the following formula:

$$k_G = \frac{1}{0.9926 + 0.03794 \left( \frac{2R}{z} \right)^2}$$

A graphical representation of gain scheduling approach in Fig. 10 is shown. Gain scheduling can be applied to both collective and gas controllers.



**Fig. 10** Gain scheduling applied to PID that controls the collective to contrast the ground effect

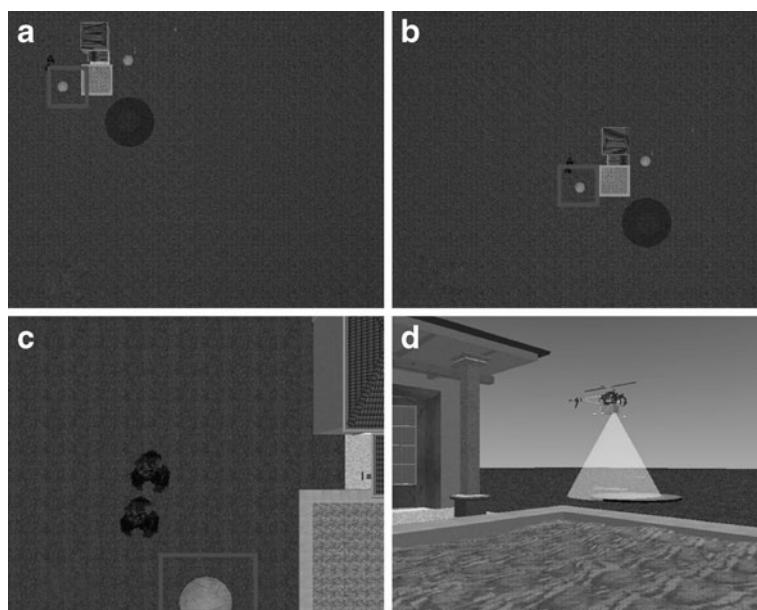
## 5 Experimental Tests

This section is organized as follows: first, results about simulations of the control system using the visual feedback as input in a virtual scenario are presented. Then, simulation results of ground effect during low height flights and landing flights are described. Finally, the visual feedback is evaluated using a real scenario and presenting results about the navigation and landing, taking into account also the safe landing area detection.

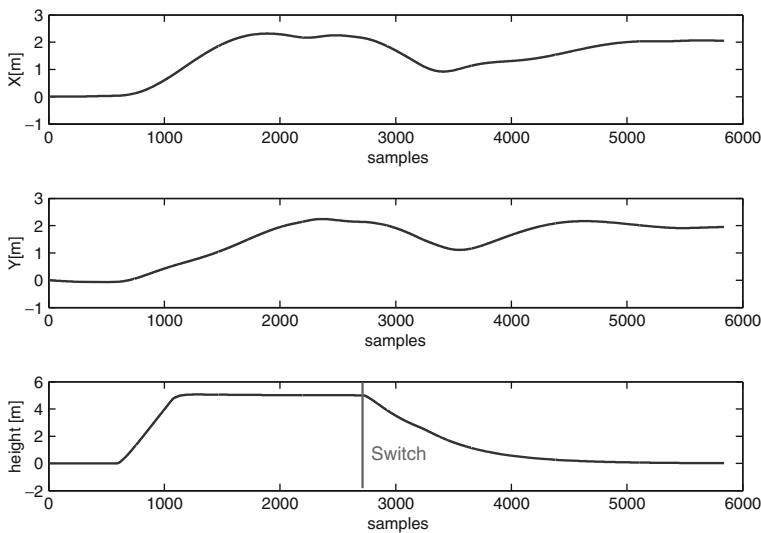
### 5.1 Simulation of Vision Sensor Feedback in a Virtual Scenario

As mentioned above, simulation is needed to evaluate the behaviour of the helicopter in transition stages. All the simulations have been carried out on a complete MATLAB/Simulink model. The virtual scenario shown in Fig. 11, developed in VRML, allows to increase the reality of simulation.

In this case, it is possible to simulate dynamically a vision sensor exploiting the camera property of VRML. More complex scenarios with real textures can be also used to stress the system more. All other sensors as GPS and AHRS are integrated in the developed simulator, allowing a full test of the system. Vision algorithms, based on feature extractors and matching, run synchronously with the simulation of helicopter's dynamics.



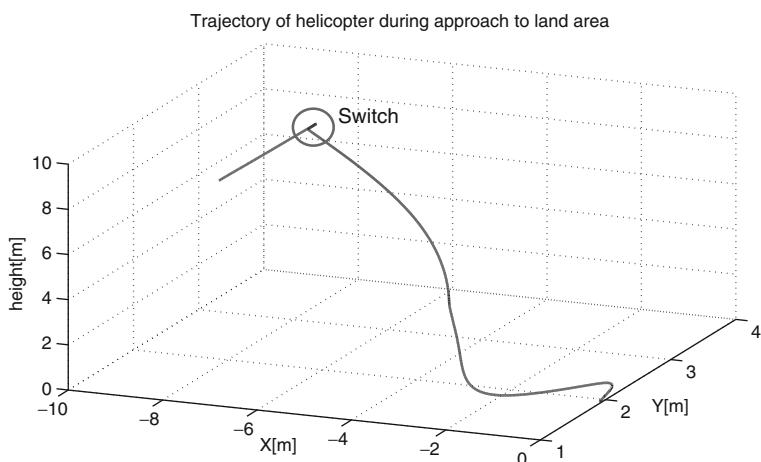
**Fig. 11** **a** the initial frame when the vision system detects features of interest as helipad or other known structures; **b** tracking is active and the helicopter performs fine tuning to reach the target; **c** the helicopter attempts to reduce height over ground; **d** simulated scenario in VRML



**Fig. 12** Graphical representation of position along  $x, y, z$  axes

In Figs. 12 and 13 the results of a simulation in the developed multi-agent simulator are shown. The simulation shows a switch from normal behaviour (waypoint following) to landing in a safe area detected by the vision algorithm.

Switching is not critical because the relevant change involves the heading of the helicopter as shown in Fig. 13; a low speed transition, which guarantees stability, is



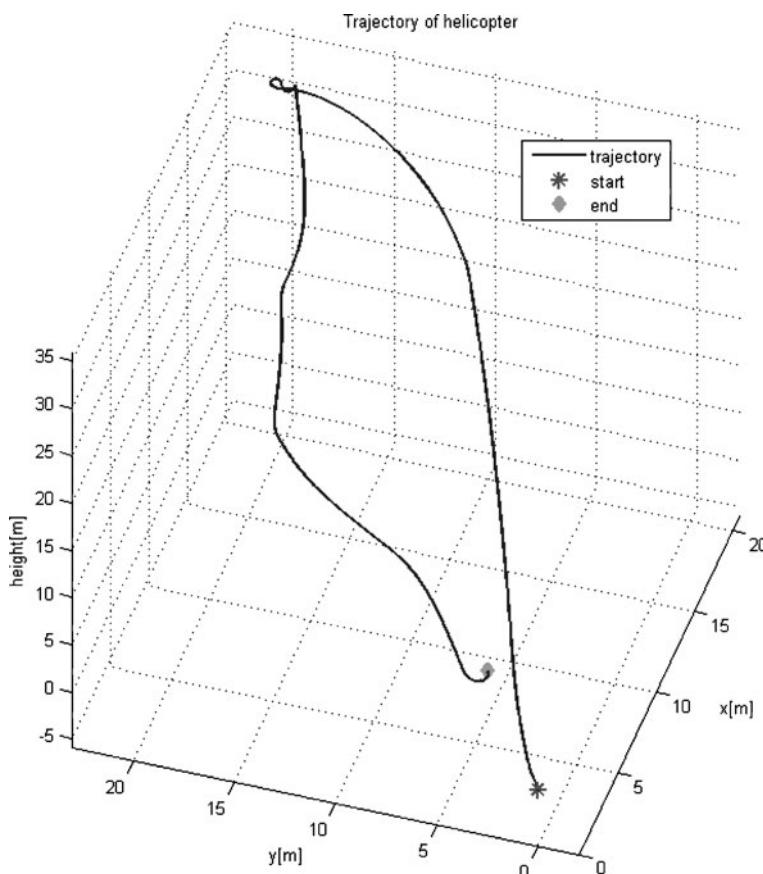
**Fig. 13** 3D graph of trajectory followed by the helicopter during the mission; the switch introduces a singularity only in the change of orientation along the yaw axis, which does not imply a loss of stability

maintained as required. Switch is triggered by the vision controller that detects the landing area; this high level information is then used by the supervisor, which takes the decision to switch from standard to vision-based control.

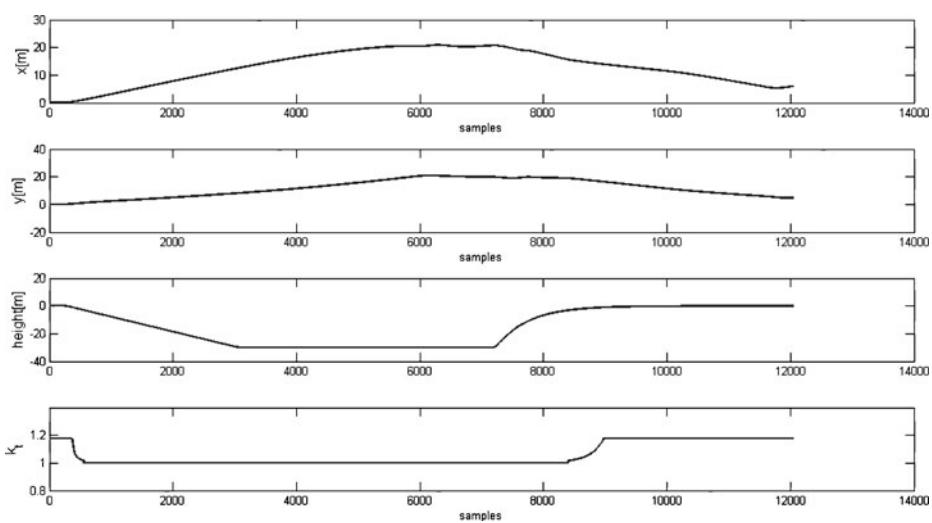
## 5.2 Simulation of Ground Effect during Low Height Flights and Landing Flights

In Figs. 14, 15 and 16 results for an interesting scenario are presented. The scenario is characterized by the presence of a switch that simulates a trigger from the vision sensor system to start an emergency/urgent landing in a safe area. Then the helicopter quickly reduces the altitude and receives the command to flight at very low altitude (0.25 m) to test the performance of the controller during the IGE hovering.

The helicopter switches from standard navigation (as shown with red line in Fig. 16) to landing task; in the last phases the rate of descent is low according to desired behaviour. Ground flights are suitable for special tasks as mine detection, where the helicopter is equipped with a Synthetic Aperture RADAR (SAR); in



**Fig. 14** Trajectory of helicopter during entire mission



**Fig. 15** Diagram of positions (horizontals and vertical); bottom graph reports the trend of corrective factor to contrast the ground effect

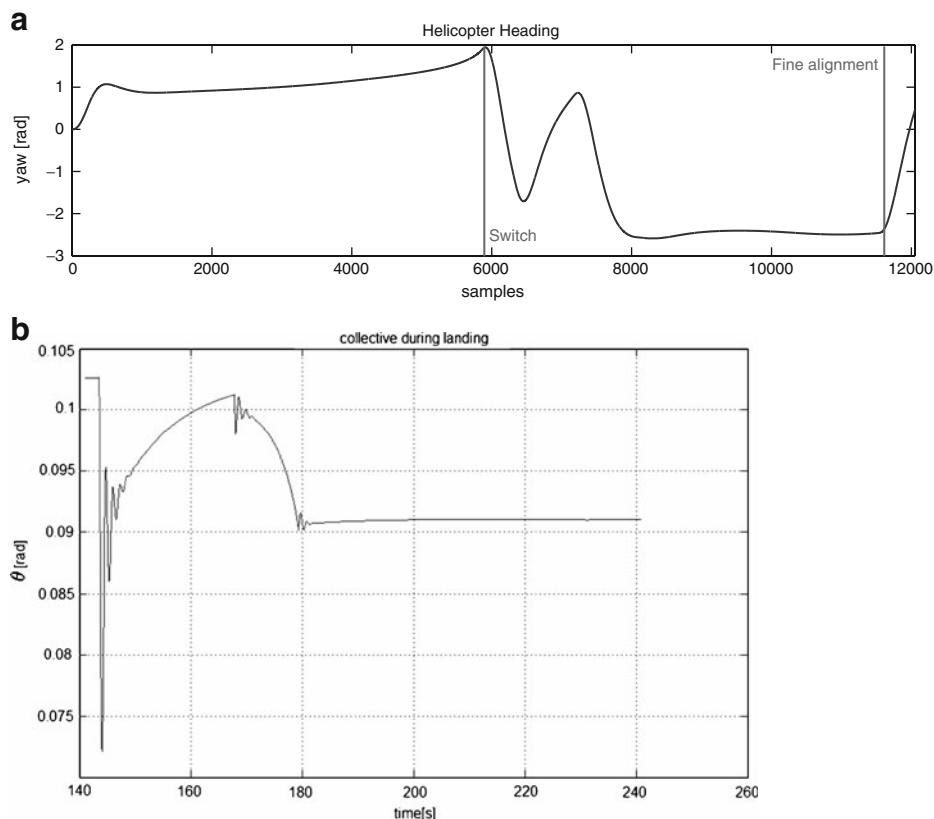
this case it is essential that the distance (also speed) between antenna and ground is minimal.

### 5.3 Vision Results in Real Scenario

For the vision part, images with  $320 \times 240$  pixels and shot by a Nikon camera are used. These images can be processed at five frames per second. In the case of autonomous landing this is also the frequency of the feedback input to the control system. The vision system has been tested in different situations: the tracking of a natural landmark for waypoint based navigation, the landing using a natural landmark and giving a feedback to the control system, the detection of a safe landing area.

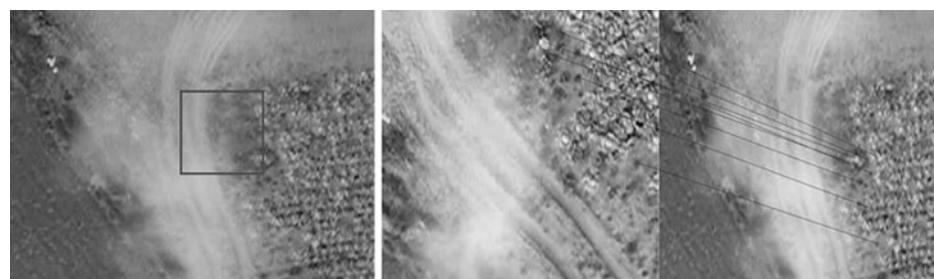
Figures 17 and 18 show the matching between the frames while the UAV is flying over natural landmarks. This test was performed to show the robustness of the vision system when the UAV is far from the landmark and is moving at high speed. Blue lines are the results of the feature matching used to track the waypoints. In Fig. 17 is also visible the smoke produced by the engine of the UAV, which causes several partial occlusions; the system behaves well also in these cases, due to the high number of features being tracked. In Fig. 19 an example to show the rotational invariance of SIFT matching is presented.

The next test shows the feasibility of the feature tracking system to guide an autonomous vision based landing. Data collected from this test are also used for a simulation of the safety analysis of the landing area, at the end of this section. The altitude feedback is given to the control system evaluating the centre of mass of matched features with respect to the centre of the actual view. Figure 20 shows a sequence of a landing using always the same feature-based approach and a natural landmark.

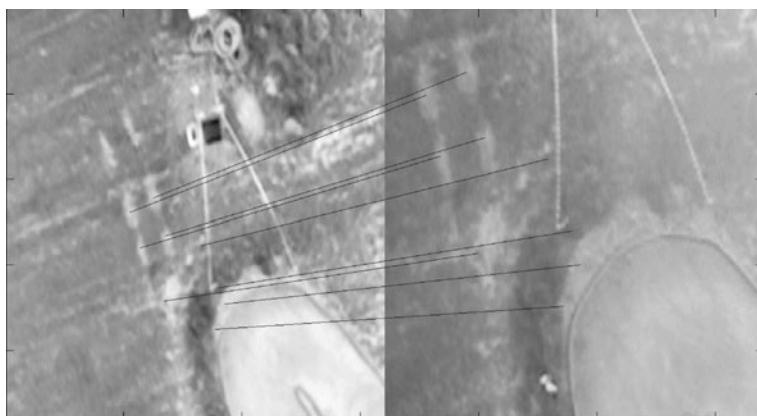


**Fig. 16** Heading of helicopter during mission; red lines evidence significant instants (switch and fine alignment)

The final tests were performed to evaluate the optical flow approach for a safe landing area classification system on the real UAV. Firstly, images taken during linear flights at constant height over the ground (first case described in Section 3)



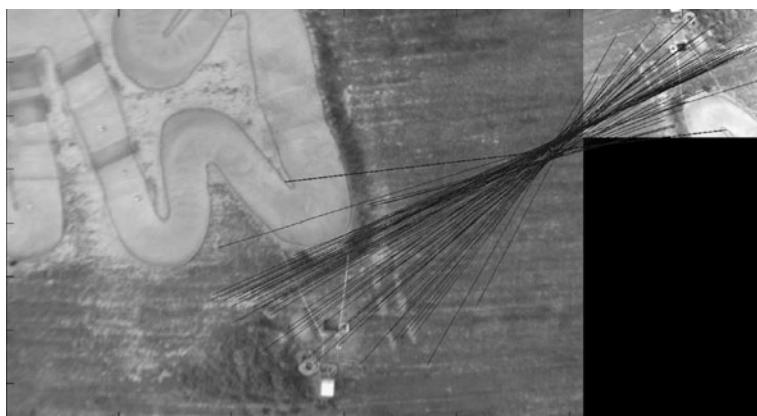
**Fig. 17** Tracking of the landing area with respect to the given one, using a natural landmark and dealing with smoke occlusions



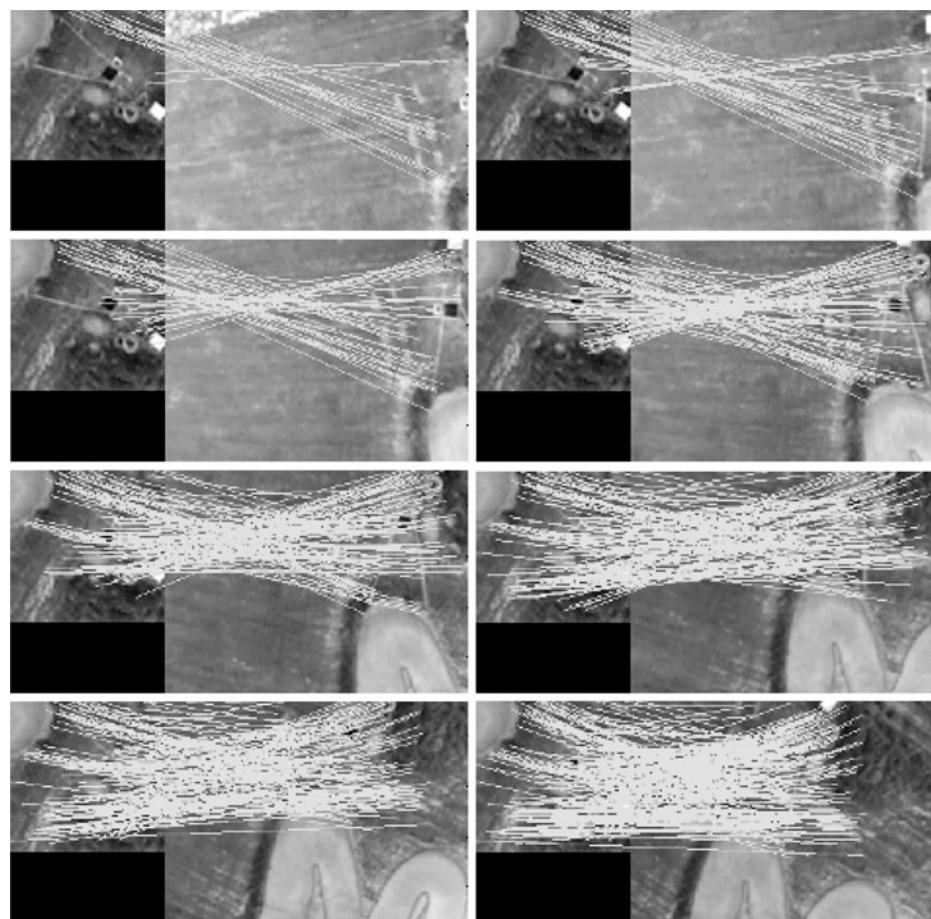
**Fig. 18** Matching between two frames using a natural landmark for landmark localization and tracking

have been analyzed. Here following two different cases of depth estimation and surface classification are reported. In Fig. 21 two illustrative sequences of images taken flying over an unsafe area (Fig. 21a) and a safe (Fig. 21b) area are shown. Every pair of consequential images was analyzed. A variance analysis of geometrical characteristics of the blue lines obtained from SIFT feature matching was performed and a representative value computed. Figure 22 shows two different situations: the first matching (unsafe area) presents a variance parameter almost hundred times greater than second one (safe area). The variance parameters in Fig. 22a and b are 382.06 and 2.78 respectively, with an experimentally determined threshold of 50.

A second test (case of vertical landing described in Section 3) was also performed in a real scenario. The radial movement of features, shown in Fig. 23, is the base of the heuristic that evaluates the kind of surface and, according to an experimental



**Fig. 19** An example to show the rotational invariant SIFT matching

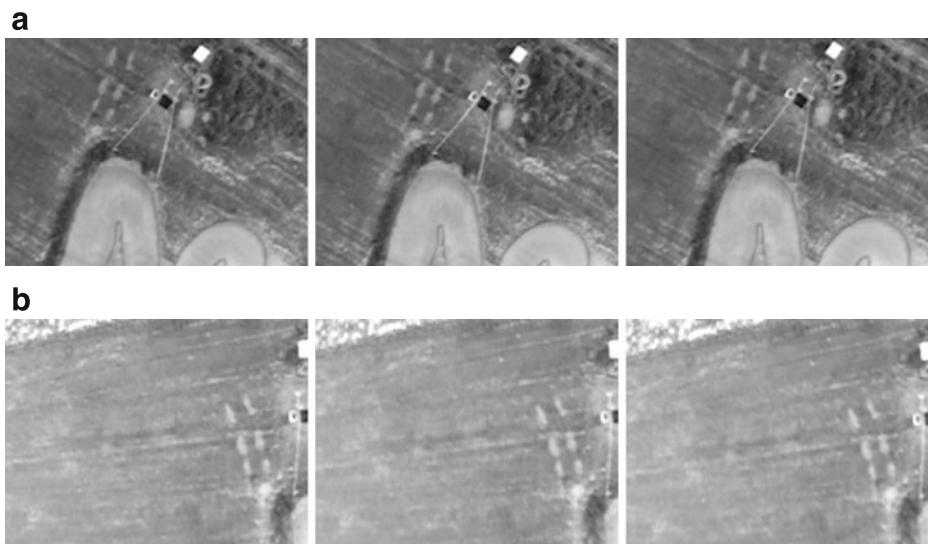


**Fig. 20** A sequence of target identification and tracking used for autonomous navigation and landing

threshold, classifies the safe landing area. Figure 23 shows the case of the helicopter approaching a safe (Fig. 23a) and an unsafe (Fig. 23b) landing area. In the first case the control system allows the helicopter to land. In the second case the safe landing area classification gives back to the supervisor the command to leave the area because it is unsafe.

Figure 24 is the comparison between parameters of the safe and unsafe area. The evaluation of the variance of the distances between features among all extracted features and frame by frame during the vertical landing, is a good and fast methodology. The heuristic does not add computational time to the process with respect to the feature extraction process, which is much bigger than all the post processing here proposed and presented. This enforces the general idea of re-using the same features extracted for landing area detection and tracking for other purposes, such as the safe landing area classification.

Other preliminary results were obtained using SURF and other variants of these approaches, previously cited. All bring to very similar results and here only SIFT



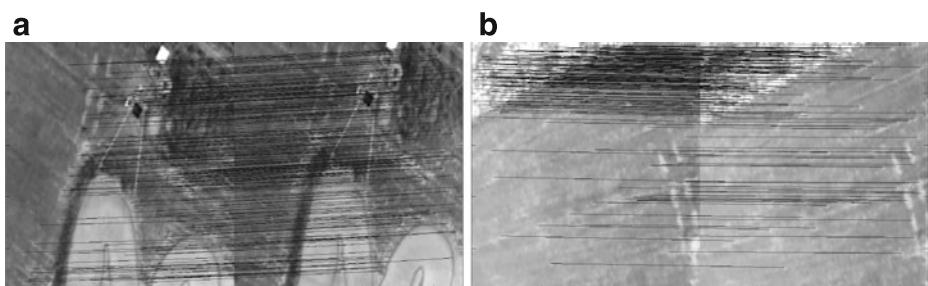
**Fig. 21** Two illustrative sequences of images taken flying over an unsafe area (**a**) and a safe area (**b**), respectively

ones are presented. The proposed approach is totally general and applicable to any kind of local feature extractor.

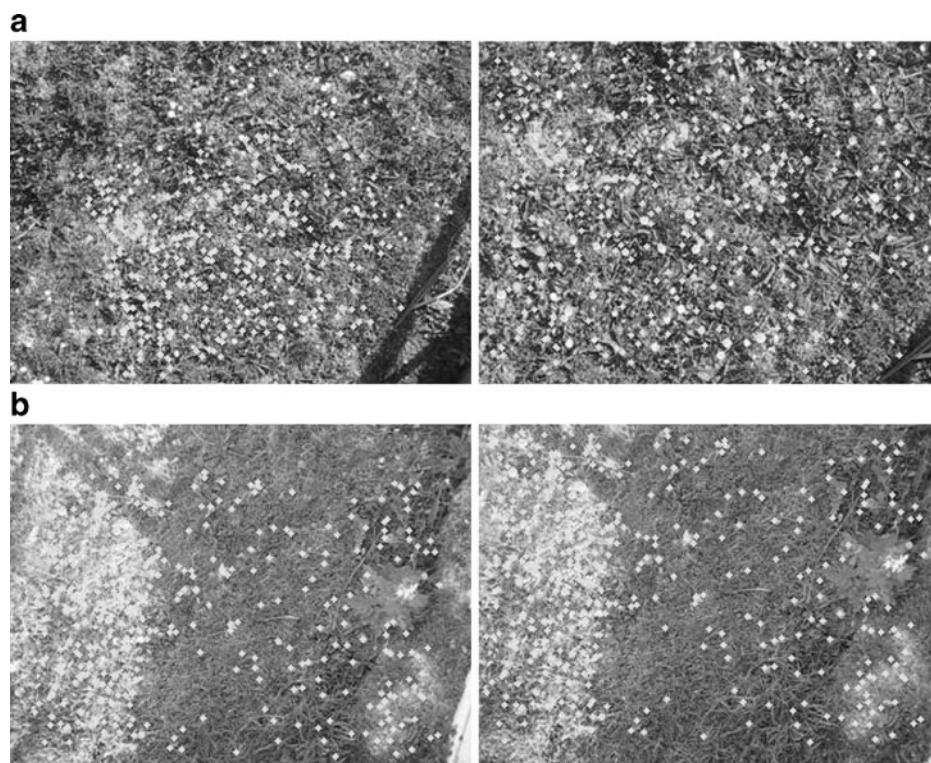
The real scenario results obtained with the use of HELIBOT are encouraging and current works are on the extension of this approach for motion analysis of the landing area (e.g., avoiding streets or flat areas used by people).

## 6 Conclusions and Future Works

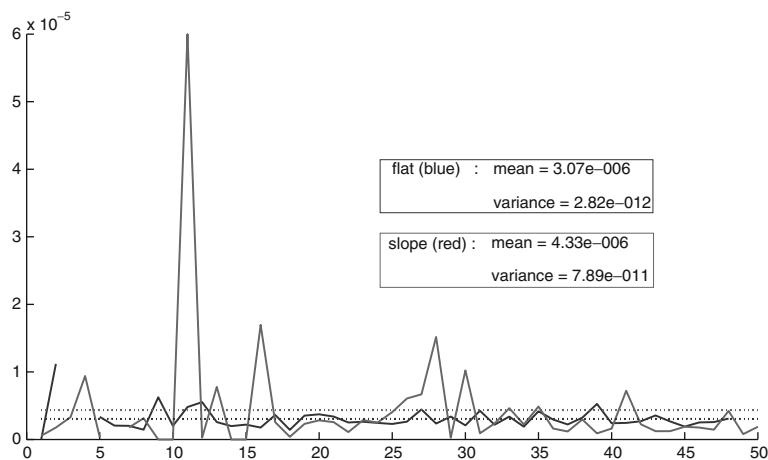
In this paper the design and implementation of a vision-based landing and navigation algorithm for an autonomous helicopter has been presented. A hierarchical behaviour-based controller exploits alternately GPS and vision input data, allowing the helicopter to navigate from an initial position to a final position in a partially known environment, to locate a landing target and to land on it. The vision system



**Fig. 22** SIFT Features matching analysis. Variance parameters are 382.06 (**a**) and 2.78 (**b**)



**Fig. 23** Frame by frame matching of a safe (**a**) area and of an unsafe landing region (**b**). Evaluated parameters are reported in next figure



**Fig. 24** Mean (dashed lines) and Variance (continuous lines) parameters used for the classification; blue continuous line represents a flat area, while the red one is the variance of the slope, that is an unsafe area for the landing

allows to define a target area from a high resolution aerial or satellite image, hence to define the waypoints of the navigation trajectory or the landing area. Two algorithms for safe landing area detection are also proposed, based on the optical flow analysis. Results show the appropriateness of the vision based approach that does not require any artificial landmark and is robust to occlusions and light variations.

Future works are oriented to improve robustness of the whole system and specifically some important aspects of the landing task concerning with safety. The possibilities related to the optical flow techniques presented in the last part of the paper will be explored to obtain not only the simple evaluation of the landing area flatness, but even the capacity to actively search a safe zone. The first objective will be the detection of safe and unsafe areas from the same image.

The applications of such a system are notable; from exploration and rescue to target tracking and data acquisition.

**Acknowledgements** The authors gratefully acknowledge Antonio Pinelli, Biagio Ambrosio and Alberto Cardinali for their essential support.

## References

1. Valavanis, K.: Advances in unmanned aerial vehicles: state of the art and the road to autonomy. *Intelligent Systems, Control and Automation: Science and Engineering* **33** (2007)
2. Bejar, M., Ollero, A., Cuesta, F.: Modeling and control of autonomous helicopters, advances in control theory and applications. *Lect. Notes Control Inf. Sci.* **353** (2007)
3. Lee, D., Jin Kim, H., Sastry, S.: Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *Int. J. Control Autom. Syst.* **7**(3), 419–428 (2009)
4. Bernard, M., Kondak, K., Hommel, G.: Framework for development and test of embedded flight control software for autonomous small size helicopters. *Embedded Systems – Modeling, Technology, and Applications*, pp. 159–168 (2006)
5. Monteriù, A., Asthana, P., Valavanis, K., Longhi, S.: Model-based sensor fault detection and isolation system for unmanned ground vehicles: theoretical aspects (part i and ii). In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2007)
6. Conte, G., Doherty, P.: An integrated UAV navigation system based on aerial image matching. In: *IEEE Aerospace Conference*, pp. 1–10 (2008)
7. Luo, P., Pei, H.: An autonomous helicopter with vision based navigation. In: *IEEE International Conference on Control and Automation* (2007)
8. He, Z., Iyer, R.V., Chandler, P.R.: Vision-based UAV flight control and obstacle avoidance. In: *American Control Conference* (2006)
9. Mondragon, I.F., Campoy, P., Correa, J.F., Mejias, L.: Visual model feature tracking for UAV control. In: *IEEE International Symposium on Intelligent Signal Processing, WISP* (2007)
10. Campoy, P., Correa, J.F., Mondragón, I., Martínez, C., Olivares, M., Mejías, L., Artieda, J.: Computer vision onboard UAVs for civilian tasks. *J. Intell. Robot. Syst.* **54**(1–3), 105–135 (2009)
11. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Vision-based odometry and SLAM for medium and high altitude flying UAVs. *J. Intell. Robot. Syst.* **54**(1–3), 137–161 (2009)
12. Bethke, B., Valenti, M., How, J.: Cooperative vision based estimation and tracking using multiple UAVs. In: *Advances in Cooperative Control and Optimization*. *Lect. Notes Control Inf. Sci.*, vol. 369, pp. 179–189 (2007)
13. Merz, T., Duranti, S., Conte, G.: Autonomous landing of an unmanned helicopter based on vision and inertial sensing. *Experimental Robotics IX, Springer Tracts in Advanced Robotics*, vol. 21, pp. 343–352 (2006)
14. Daquan, T., Hongyue, Z.: Vision based navigation algorithm for autonomic landing of UAV without heading & attitude sensors. In: *Proceedings of the Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pp. 972–978 (2007)
15. Meingast, M., Geyer, C., Sastry, S.: Vision based terrain recovery for landing unmanned aerial vehicles. In: *43rd IEEE Conference on Decision and Control (CDC)*, vol. 2, pp. 1670–1675 (2004)

16. Shakernia, O., Vidal, R., Sharp, C.S., Ma, Y., Sastry, S.S.: Multiple view motion estimation and control for landing an unmanned aerial vehicle. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2793–2798 (2002)
17. Saripalli, S., Montgomery, J., Sukhatme, G.: Visually-guided landing of an unmanned aerial vehicle. *IEEE Trans. Robot. Autom.* **19**(3), 371–381 (2003)
18. Saripalli, S., Sukhatme, G.S.: Landing a helicopter on a moving target. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2030–2035 (2007)
19. Garcia-Padro, P.J., Sukhatme, G.S., Montgomery, J.F.: Towards vision-based safe landing for an autonomous helicopter. In: *Robotics and Autonomous Systems*, vol. 38, no. 1, pp. 19–29(11). Elsevier (2002)
20. Johnson, A., Montgomery, J., Matthies, L.: Vision guided landing of an autonomous helicopter in hazardous terrain. In: Proceedings of the IEEE International Conference on Robotics and Automation (2005)
21. Templeton, T., Shim, D.H., Geyer, C., Sastry, S.: Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1349–1356 (2007)
22. Se, S., Lowe, D., Little, J.: Vision-based mobile robot localization and mapping using scale-invariant features. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2051–2058 (2001)
23. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004)
24. Frontoni, E., Zingaretti, P.: Adaptive and fast scale invariant feature extraction. In: Second International Conference on Computer Vision Theory and Applications, Workshop on Robot Vision (2007)
25. Frontoni, E., Zingaretti, P.: Feature extraction under variable lighting conditions. CISI (2006)
26. Bramwell, A.R.S., Done, G., Balmford, D.: *Bramwell's Helicopter Dynamics*, 2nd edn. Butterworth Heinemann (2001)
27. Mancini, A., Cesetti, A., Iuale', A., Frontoni, E., Zingaretti, P., Longhi, S.: A framework for simulation and testing of UAVs in cooperative scenarios. In: International Symposium on Unmanned Aerial Vehicles (UAV'08) (2008)
28. Montgomery, J.: Learning helicopter control through “teaching by showing”. Ph.D. Thesis, School of Comp. Sci., USC (1999)
29. Mataric, M.J.: Behavior-based control: examples from navigation, learning and group behavior. *J. Exp. Theor. Artif. Intell. (Special Issue on Software Architecture for Physical Agents)* **9**(2–3), 67–83 (1997)
30. Shim, D.H., Kirn, H.J., Sastry, S.: Hierarchical control system syntesys for rotorcraft-based unmanned aerial vehicles. In: AIAA Guidance, Navigation and Control Conference and Exhibit (2000)

## Autonomous Vision-Based Helicopter Flights Through Obstacle Gates

Franz Andert · Florian-M. Adolf · Lukas Goormann ·  
Jörg S. Dittrich

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 21 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** The challenge for unmanned aerial vehicles to sense and avoid obstacles becomes even harder if narrow passages have to be crossed. An approach to solve a mission scenario that tackles the problem of such narrow passages is presented here. The task is to fly an unmanned helicopter autonomously through a course with gates that are only slightly larger than the vehicle itself. A camera is installed on the vehicle to detect the gates. Using vehicle localization data from a navigation solution, camera alignment and global gate positions are estimated simultaneously. The presented algorithm calculates the desired target waypoints to fly through the gates. Furthermore, the paper presents a mission execution plan that instructs the vehicle to search for a gate, to fly through it after successful detection, and to search for a proceeding one. All algorithms are designed to run onboard the vehicle so that no interaction with the ground control station is necessary, making the vehicle completely autonomous. To develop and optimize algorithms, and to prove the correctness and accuracy of vision-based gate detection under real operational conditions, gate positions are searched in images taken from manual helicopter flights. Afterwards, the integration of visual sensing and mission control is proven. The paper presents results from full autonomous flight where the helicopter searches and flies through a gate without operator actions.

---

F. Andert (✉) · F.-M. Adolf · L. Goormann · J. S. Dittrich  
Institute of Flight Systems, Unmanned Aircraft, German Aerospace Center (DLR),  
Lilienthalplatz 7, 38108 Braunschweig, Germany  
e-mail: franz.andert@dlr.de

F.-M. Adolf  
e-mail: florian.adolf@dlr.de

L. Goormann  
e-mail: lukas.goormann@dlr.de

J. S. Dittrich  
e-mail: joerg.dittrich@dlr.de

**Keywords** Unmanned aerial vehicle · Autonomous helicopter flight · Image processing · Target recognition · Sensor fusion · Obstacle avoidance

## 1 Introduction

An auto-piloted unmanned vehicle with environmental sensors needs the ability of situational awareness that is essential when autonomous operations in unknown urban terrain are aspired. Missions beyond line of sight or with limited ground control station access require capabilities of autonomous and safe navigation and necessitate a continuous extension of existing and potentially outdated information about the environment.

For example, current research on aerial robots covers obstacle avoidance behaviors [19, 22], mapping [2] and map-based path planning [11, 17], autonomous landing [18], detection and tracking of specific targets [15] or following moving ground vehicles [4].

Safe and precise navigation close to obstacles are a matter of particular interest. This paper addresses two problems that are often negligible in terrain with a lot of free space, but of major importance when flying into narrow passages, under bridges or through windows.

The first problem is that a safe passage is nearly impossible with only a geo-referenced map and GPS-based vehicle localization which cannot guarantee the demanded accuracy. The vehicle must “see” the environment to increase knowledge about true object positions and to get its own position relative to them.

Environmental sensing leads to the second problem. Flight instructions based on environmental sensors require a very accurate relative orientation between sensor and navigation solution. Navigation sensor uncertainties, drift, and imprecise calibration lead to biases of the estimated vehicle position and attitude angles. Additional sensors like cameras aligned with the vehicle are not aligned with the navigation system which leads to errors in sensor-based navigation instructions. Beside that, the orientation between sensors and vehicle is manually measured and imprecise as well. Hence, accurate sensor fusion between visual and inertial data is required [7, 20].

The following example scenario acts as a methodology demonstration that copes with both problems. Gates are assembled on a flight field, and an unmanned helicopter shall fly through them without any operator interaction. The obstacle sizes and appearances are previously known as well as their approximate global position in GPS-coordinates. Gate width and height are approximately 6 m and therefore a safe passage requires to hit the gate center with a deviation of much less than three meters since the vehicle size must also be considered.

A calibrated monocular color camera is used for object detection. While flying towards the initially given gate position, the position relative to the camera is estimated with image processing techniques. At the same time, sensor misalignment is estimated by comparing the vehicle trajectory from navigation with visual odometry during flight. The presented algorithms enable these tasks without the need of separate calibration flights. Output is a gate position and orientation estimation relative to the navigation reference. This position is used to guide the vehicle through

the gate on a determined heading. Due to drift errors in relative navigation, it is recommended to completely repeat the gate localization and sensor misalignment estimation process if another gate shall be passed.

The approach has similarities with simultaneous localization and mapping techniques [6, 8, 21] where environmental measurements are used not only to locate object positions, but also to re-locate the vehicle so that actual sensing information aligns with a map built from previous measurements. In the presented scenario, the relative vehicle position in the world is of major importance without regarding the global world map consistency. Hence, the *world* is re-located so that the sensor readings match with the map while the vehicle position is kept. An advantage is that no changes of vehicle navigation and guidance are needed. It is easy to readjust this image-based application for other vehicles, including those with proprietary auto-piloting systems having limited modification options.

## 2 Flight Hardware

The test vehicle is an auto-piloted model helicopter (Fig. 1) with all computational power on-board. It is developed within the ARTIS (Autonomous Rotorcraft test bed for Intelligent Systems) project doing aeronautic and robotic research on unmanned helicopters [1, 9]. Avionics payload comprises GPS, IMU and magnetometer for navigation, sonar altimeter for landing, wireless data links, manual R/C and video data links, a telemetry module and a flight control computer. The image acquisition and processing equipment consists of an industry color camera (Imaging Source DFX 31F03, 1024 × 768 pixels, max. 30 fps, lens: 60° × 48° field of view) and a separate vision computer (Intel Pentium 4 Mobile, 1.8 GHz).

Basis for robotic applications are an EKF-based navigation filter providing the actual flight state, especially the absolute position and attitude angles, and a flight controller that allows precise vehicle guidance and the input of high-level commands from external applications. Autonomous flights are accomplished with a loosely-coupled approach where the flight controller is not modified. The algorithms

**Fig. 1** Research UAV

ARTIS: a combustion-engine-powered autonomous helicopter, 1.9 m rotor diameter, 12 kg gross weight



presented in this paper act as the application that triggers waypoint position instructions to the autopilot.

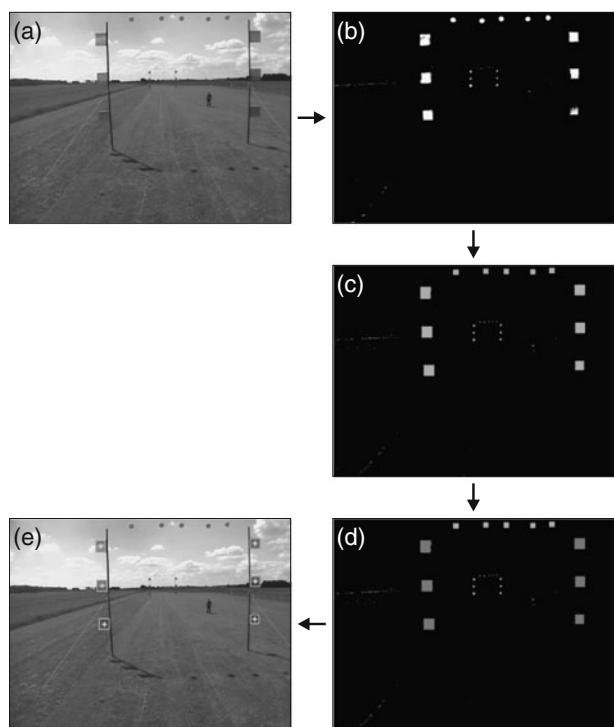
### 3 Vision-Based Target Finding

This section describes obstacle detection with monocular vision to estimate the relative gate position and orientation for this specific scenario. In the later sections, the paper deals with other fundamental concerns required for a successful solution for narrow passages, and a simple solution for a limited case is presented here. The approach of this paper can be extended to more general scenarios by modifying this target finding method, for example by using laser range finders or stereo vision. Like the presented method, general approaches have to determine the sensor-relative orientation of a waypoint that leads through a passage.

#### 3.1 Image Processing

Each gate is constructed with three flags at each post for automatic recognition. Figure 2 shows the main steps used to locate the flag positions that will be the basis for locating the vehicle relative to the gate. The image processing approach to find and locate the gate flags consists of the following steps:

**Fig. 2** Vision-based gate recognition. Camera image (a), segmented image (b), potential objects (c), chosen flag objects (d), and projection of this pattern into the camera image (e)



**Segmentation:** The image is segmented to produce a binary image (Fig. 2b) with pixels that may represent parts of the flags (white) and those who do not (black). Segmentation is achieved with a pixel-based color filter (HSV color space) where all of the following criteria must be fulfilled to generate a white output pixel:

- hue: close to red (between  $300^\circ$  and  $25^\circ$ ),
- saturation: above a fixed threshold ( $\approx 30\%$ ),
- value: above a fixed threshold ( $\approx 45\%$ ).

The parameters can be modified to detect flags of different appearance, e.g. a different color.

**Clustering:** Neighboring white pixels of the binary image are merged into clusters. For each cluster, store size (number of connected pixels), centroid (center of gravity, i.e. average pixel position) and bounding box (minimum and maximum row and column, i.e. also width and height).

**Cluster Filtering:** Clusters are filtered out that are

- too small or too big (size  $< 30$  or size  $> 3500$  pixels here), and
- far away from squarish shape (width  $< 0.5 \cdot \text{height}$  or width  $> 2 \cdot \text{height}$ ).

Remaining clusters are marked in Fig. 2c.

**Cluster Relationship:** Use the centroids of the remaining clusters to find

- lines with exactly three clusters on it (or close to them) and that are almost vertical in the image, and out of these
- similar line couples where the segment lengths between the clusters are nearly equal.

If more than one couple is found, take the one with larger clusters and a higher distance between the clusters. This line couple will probably represent the nearer gate. An example is shown in Fig. 2d. By projecting the clusters and their centroids back to the camera image, the correspondence between gate flags and the clusters is visible (Fig. 2e).

The cluster centroids mark six feature positions that can now be used to locate the vehicle relative to the gate. Ambiguities are eliminated by giving unique identifiers to the six features by their position in the image, e.g. beginning with the upper left.

**Feature Tracking:** There is no need to completely repeat the flag detection process for each image while processing sequences. Once all six flags have been found, they are observed using image feature tracking algorithms. For example, the classic Lucas-Kanade algorithm [14] is very fast and performs a good tracking if the flag corners are significantly distinguishable from the background. Alternatively, there is a variety of new powerful algorithms like SIFT [13] suitable for this task. If the feature tracking is not successful or is of low confidence, features get lost during the image sequence and the flag detection process starts again from the beginning.

**Sequence Cutting:** Gate position estimation is based on either a single image, or an image sequence without ambiguities between multiple gates. They are distinguished

by analyzing feature tracking results. If detected gate flags get lost by tracking and a new flag detection provides significantly different flag positions, a cut is done and a new sequence is started. Within this paper, an image sequence is defined that all included images refer to the same gate.

### 3.2 Estimation of the Gate Position

Now, the orientation of the gate relative to the camera is estimated for images where all six flags have been found. It is solved with camera resectioning techniques as used in photogrammetry. A preceding calibration estimates the constant internal camera parameters such as image center, focal length, and lens distortion from the close-range model [5]. It is done with *OpenCV* image processing library methods applied to images of a typical chessboard calibration pattern. For the gate position estimation, normalized image coordinates and the pinhole camera model are used to estimate the external orientation of the camera with respect to the gate.

Figure 3 shows how the image feature points are related to the real flag positions. The perspective transformation for corresponding points  $\mathbf{p}_j \in \mathbb{R}^3$  and  $\mathbf{q}_j \in \mathbb{R}^4$  in homogenous coordinates is

$$\lambda_j \mathbf{p}_j = [\mathbf{R}_{\mathbf{q}_i} \mathbf{t}_{\mathbf{q}_i}] \mathbf{q}_j \quad (1)$$

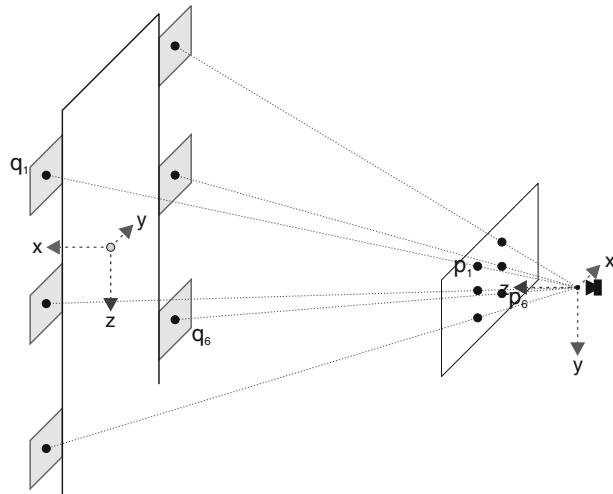
with a scale factor  $\lambda_j$  and the camera orientation given by rotation  $\mathbf{R}_{\mathbf{q}_i}$  and translation  $\mathbf{t}_{\mathbf{q}_i}$  of the  $i$ -th image. Following [18], this equation can be transformed into

$$(\mathbf{p}_j \mathbf{e}_3^T - I)[\mathbf{R}_{\mathbf{q}_i} \mathbf{t}_{\mathbf{q}_i}] \mathbf{q}_j = 0 \quad (2)$$

with  $\mathbf{e}_3 = (0, 0, 1)^T$ . This equation is valid for all corresponding points in an ideal case. Due to noise, there are uncertainties which leads to the problem finding  $\mathbf{R}_{\mathbf{q}_i}$  and  $\mathbf{t}_{\mathbf{q}_i}$  that minimizes the error

$$E(\mathbf{R}_{\mathbf{q}_i}, \mathbf{t}_{\mathbf{q}_i}) = (\mathbf{p}_j \mathbf{e}_3^T - I)[\mathbf{R}_{\mathbf{q}_i} \mathbf{t}_{\mathbf{q}_i}] \mathbf{q}_j. \quad (3)$$

**Fig. 3** Pinhole camera model to estimate the relative gate position



A solution to this problem is presented in [18]. It assumes a minimum of four coplanar points  $\mathbf{q}_j$  (here:  $j = 1, \dots, 6$ ) and consists of a linear initialization step that produces approximated results of  $\mathbf{R}_{\mathbf{q}_i}$  and  $\mathbf{t}$ , followed by a non-linear multivariate Newton-Raphson iteration that produces a more accurate result by minimizing the reprojection error. Since scaling is not obtained with vision-based estimation algorithms, the size of the gate is given by the flag positions with respect to the gate center.

The origin of the gate coordinate system marks the point where the vehicle shall pass the gate in  $x$ -direction. Hence,  $\mathbf{R}_{\mathbf{q}_i}$  and  $\mathbf{t}_{\mathbf{q}_i}$  give a transformation between the actual camera orientation and the desired target.

## 4 Calculating the Global Gate Position

This section describes the process to calculate the global position of the gate. It is based on the camera position relative to the gate as described above and uses information about camera assembly and vehicle navigation required for the coordinate transformations.

### 4.1 Preliminary Transformations

Let  $\mathbf{R}_{\mathbf{g}_i}$  and  $\mathbf{g}_i$  denote the helicopter orientation relative to the gate estimated using the  $i$ -th image. With known camera alignment  $\mathbf{R}_{hc}$  and  $\mathbf{t}_{hc}$ , it is

$$\mathbf{R}_{\mathbf{g}_i} = \mathbf{R}_{hc} \mathbf{R}_{\mathbf{q}_i}, \quad \text{and} \quad \mathbf{g}_i = \mathbf{R}_{hc} \mathbf{t}_{\mathbf{q}_i} + \mathbf{t}_{hc}. \quad (4)$$

Using the corresponding helicopter orientation  $\mathbf{R}_{\mathbf{h}_i}$  and position  $\mathbf{h}_i$  from the navigation filter, the global target  $\mathbf{R}$ ,  $\mathbf{t}$  is determined by

$$\mathbf{R} = \mathbf{R}_{\mathbf{h}_i} \mathbf{R}_{\mathbf{g}_i}, \quad \text{and} \quad \mathbf{t} = -\mathbf{R}_{\mathbf{h}_i} \mathbf{g}_i + \mathbf{h}_i. \quad (5)$$

Based on this rotation, the desired direction  $\alpha$  to fly through the gate is determined by the rotation matrix elements. It is

$$\alpha = \text{atan2}(\mathbf{R}_{21}, \mathbf{R}_{11}). \quad (6)$$

Ideally, the flight path azimuth  $\chi$  is equal to this direction angle  $\alpha$  while the gate is crossed. Since the vehicle is not advised to fly sideways, its ideal heading  $\Psi$  is also equal to  $\alpha$ .

Experiments have shown that this estimated gate position and direction is only roughly valid, see the results presented in Section 6.1. While the helicopter approaches the gate, the estimated gate position moves dependent on the helicopter position. This strongly suggests systematic errors since the gate does not move in real. Primary error sources are vision-based, for example the relative orientation estimation accuracy dependent on the viewing perspective, alignment-based which means that biases in the camera orientation are existent, and navigation-based which refers to the measured geodetic vehicle orientation.

These errors are not large with few meters and may be negligible when navigating in large canyons. In the case of narrow passages as represented by the obstacle gates, it has become indispensable to perform a calibration of the external camera orientation with respect to the vehicle localization. Especially rotational errors must

be minimized since they induce an incorrect gate orientation and too large position errors when viewing the gate from far distances.

#### 4.2 Improvement with Multiple Measurements

Since multiple image and navigation measurements are taken while the vehicle approaches towards a gate, it is a straightforward procedure to integrate these measurements to improve the gate detection. The presented improvements focus on the alignment and navigation errors.

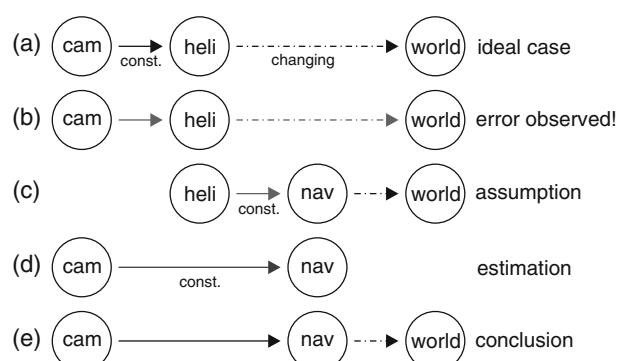
Figure 4 illustrates the approach how the gate positions are obtained anyhow. Case (a) illustrates the initial configuration. The camera is fixed at the helicopter whose orientation is changing due to vehicle movements. Both camera (*cam*) and helicopter (*heli*) orientation are assumed to be known without error, and the transformation of the gate into geodetic world coordinates is done as given in the Eqs. 4 and 5. Since the gate is not moving, its estimated position should only be influenced by noise. This was not observed as illustrated by (b). Camera alignment as well as helicopter localization can be incorrect.

To compensate for these shortcomings, (c) shows the *world* regarded as the geodetic system where the helicopter orientation is absolutely known due to its navigation filter (*nav*). There will be an orientation error between the true helicopter position and its localization by the navigation filter. This error may change over time due to drift errors, but is assumed to be constant for a short time period, i.e. for a single gate passage. Differences between the world used here and real world coordinates, e.g. from obstacle maps, are assumed to be static.

Now, sensor fusion methods are applied, see (d). Both camera and navigation filter provide localization and with that, relative movements in the same reference system. The following Sections 4.3 and 4.4, respectively, develop two approaches to compensate the camera misalignment in order to get an accurate gate localization.

The final conclusion is shown in (e) where the estimated orientation between camera and navigation coordinate system, and the orientation between navigation and world coordinates are successively used to get the camera position in the world and with that, the gate position in the world. It is not referenced to a true world of

**Fig. 4** Compensating errors in camera and helicopter alignment by estimating the transformation between camera and navigation solution

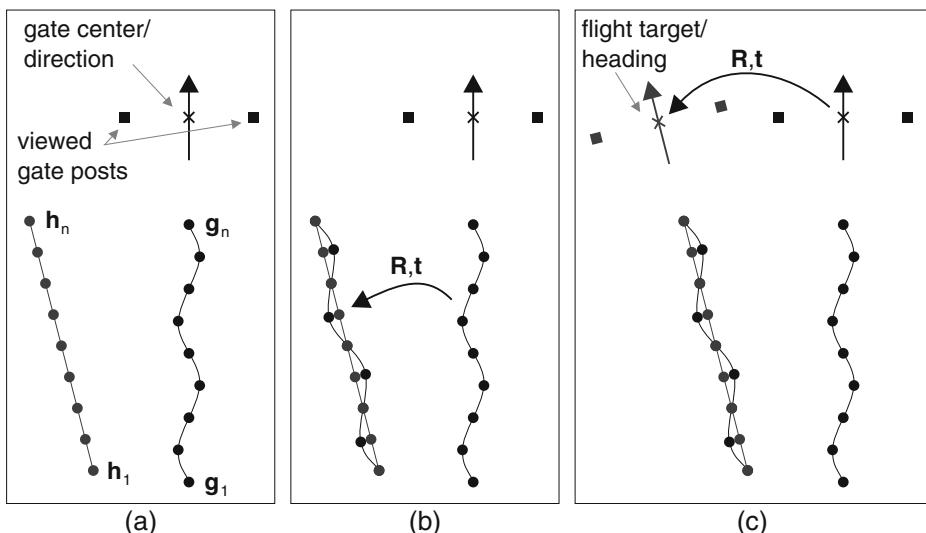


previously known maps, but to the navigation coordinate system, which is necessary to fly through the gate. The true helicopter position is never used since all commands refer to the navigation coordinate system.

#### 4.3 Trajectory Fitting

The first camera misalignment compensation approach compares the camera-based and navigation-based movement trajectories to get the relative orientation between camera and navigation system. After taking a couple of images and determining corresponding helicopter positions, vehicle and image-based movements are used to get the gate position in geodetic coordinates and eventually the position and heading to fly through the gate. Since the helicopter is instructed to fly to coordinates in its own navigation coordinate system, the position is assumed to be exact and it is not necessary to know the absolute error of the geodetic helicopter position with respect to “ground truth” of the real world. However, this error is assumed to be constant during the rather short period of time where the gate position is estimated. Figure 5 illustrates the approach to find the target to fly through the gate.

During flight towards the gate and after taking  $n$  images, gate positions are estimated from each acquired image as described in Section 3.2 (Fig. 5a). In a coordinate system with the gate as origin, the relative vehicle position based on the  $i$ -th image ( $i = 1, \dots, n$ ) is denoted as  $\mathbf{g}_i = (g_{ix}, g_{iy}, g_{iz})^T$ . For each image, corresponding geodetic helicopter positions  $\mathbf{h}_i = (h_{ix}, h_{iy}, h_{iz})^T$  based on navigation filter output are taken. Relative vehicle movement is known independently from image-based gate position estimation, as well as from navigation filter data.



**Fig. 5** Approach to find the geodetic gate position. Take navigation and image-based positions (a), find the orientation between both trajectories (b), and project the gate into the navigation coordinate system (c)

After estimating the relative orientation  $\mathbf{R}$ ,  $\mathbf{t}$  between the point sets  $\{\mathbf{h}_i | i = 1, \dots, n\}$  and  $\{\mathbf{g}_i | i = 1, \dots, n\}$  (Fig. 5b), the searched gate position can be obtained by transforming the gate position into geodetic coordinates using the estimated orientation (Fig. 5c).

The estimation of  $\mathbf{R}$  and  $\mathbf{t}$  is done with least-squares fitting based on the algorithm presented in [3]. The presented approach takes additionally into account that the points  $\mathbf{g}_i$  are more accurate if the vehicle is closer to the gate. Hence, weights  $w_i$  for the  $i$ -th image are introduced, which are dependent on the estimated distance to the gate:

$$w_i = \frac{1}{\|\mathbf{g}_i\|}. \quad (7)$$

Using these weights, nearer gate positions receive a higher relevance. The weights are normalized using

$$w'_i = \frac{w_i}{\sum_{i=1}^n w_i} \quad (8)$$

so that the sum of all ratios is unity  $\sum_{i=1}^n w'_i = 1$ .

Further, the orientation estimation problem is reduced to only rotation estimation by subtracting the weighted centers of gravity  $\bar{\mathbf{g}}$  and  $\bar{\mathbf{h}}$  of the point sets from each point. It is

$$\bar{\mathbf{g}} = \sum_{i=1}^n w'_i \mathbf{g}_i \quad \text{and} \quad \bar{\mathbf{h}} = \sum_{i=1}^n w'_i \mathbf{h}_i \quad (9)$$

and yield to the translated points

$$\mathbf{g}'_i = \mathbf{g}_i - \bar{\mathbf{g}} \quad \text{and} \quad \mathbf{h}'_i = \mathbf{h}_i - \bar{\mathbf{h}}. \quad (10)$$

As presented by [16], only a rotation has to be estimated to fit the pre-translated point sets  $\{\mathbf{g}'_i\}$  and  $\{\mathbf{h}'_i\}$ . With a least-squares approach, the rotation  $\mathbf{R}$  that minimizes

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^n \|\mathbf{h}'_i - \mathbf{R} \mathbf{g}'_i\|^2 \quad (11)$$

is determined. The minimization problem is solved by calculating a correlation matrix  $\mathbf{H} \in \mathbb{R}^{3 \times 3}$ :

$$\mathbf{H} = \sum_{i=1}^n w'_i \mathbf{h}'_i w'_i \mathbf{g}'_i{}^T = \sum_{i=1}^n w'_i{}^2 \mathbf{h}'_i \mathbf{g}'_i{}^T \quad (12)$$

and its singular value decomposition

$$\mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^T. \quad (13)$$

Finally, the rotation matrix is given by

$$\mathbf{R} = \mathbf{V} \mathbf{U}^T. \quad (14)$$

There is no translation between the point sets  $\{\mathbf{g}_i'\}$  and  $\{\mathbf{h}_i'\}$  due to their identical averages. Between the original point sets  $\{\mathbf{g}_i\}$  and  $\{\mathbf{h}_i\}$ , the estimated translation  $\mathbf{t}$  is now

$$\mathbf{t} = \bar{\mathbf{h}} - \mathbf{R}\bar{\mathbf{g}}. \quad (15)$$

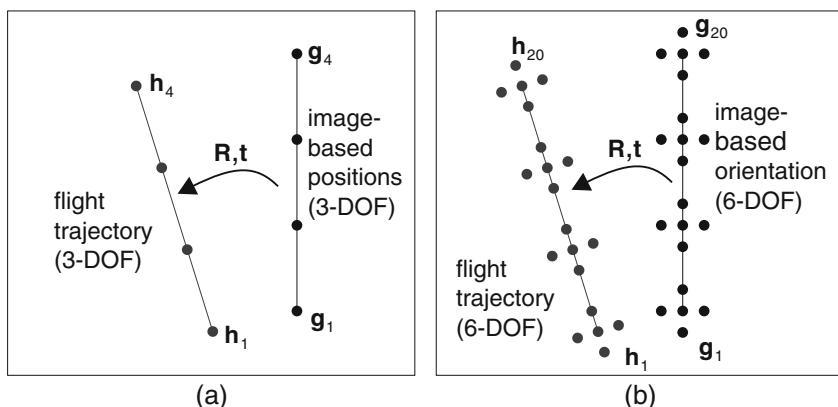
Beside typical noise, image-based position estimation has to count on failures that may result in completely wrong gate positions. The RANSAC algorithm [10] has turned out to be suitable for detecting and removing outliers, and it is used here to improve camera and navigation trajectory fitting.

Furthermore, transformation estimation using the presented method or alternative ones, e.g. depicted in [12], will provide rather poor results if the point set elements are very close to each other or nearly collinear. This can occur if the vehicle hovers or flies straightly towards the gate without curves. To cope with this problem, additional points are taken by using knowledge about the vehicle's attitude angles as illustrated by Fig. 6.

Based on a single vehicle position  $\mathbf{h}_i$  and its attitude  $\mathbf{R}_{\mathbf{h}_i}$  that is available through the navigation filter output, six additional points in a specified distance  $d$  above, below, left to, right to, in front of and behind the vehicle are taken into account. A single vehicle position and attitude yields to seven points added to the point set, they are

$$\begin{aligned} & \mathbf{h}_i, \\ & \mathbf{h}_i \pm d \cdot \mathbf{R}_{\mathbf{h}_i}^T (1, 0, 0)^T, \\ & \mathbf{h}_i \pm d \cdot \mathbf{R}_{\mathbf{h}_i}^T (0, 1, 0)^T, \\ & \mathbf{h}_i \pm d \cdot \mathbf{R}_{\mathbf{h}_i}^T (0, 0, 1)^T. \end{aligned}$$

In an analogous way, image-based points relative to the gate are generated using the positions  $\mathbf{g}_i$  and the corresponding attitude angles  $\mathbf{R}_{\mathbf{g}_i}$  known through the gate position estimation.



**Fig. 6** Using only the vehicle positions (a) versus using vehicle positions and additional points by considering attitude angles (b)

These larger point sets, for ease of use also denoted as  $\{\mathbf{g}_i\}$  and  $\{\mathbf{h}_i\}$  with their cardinality  $n$  (Fig. 6b), are now the input for the transformation estimation. Especially in straight flight trajectories, better results are expected due to the eliminated risk of collinearities.

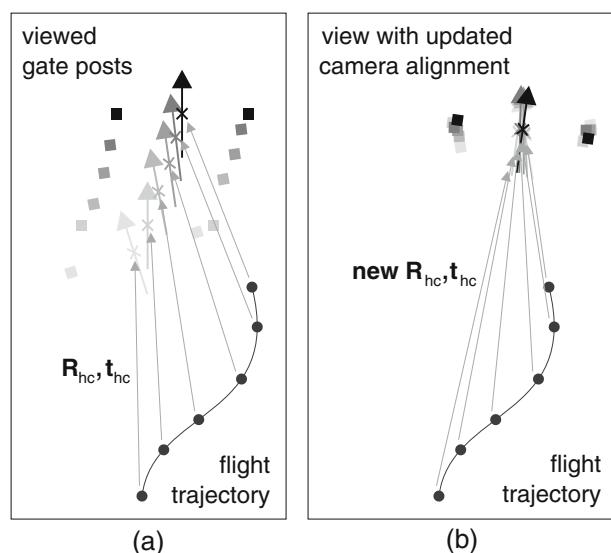
#### 4.4 Target Fitting

The second approach is based on the idea that an ideal camera alignment would lead to a projection of all estimated gate positions from different vehicle locations to the same global coordinate as illustrated in Fig. 7. The problem is solved by searching for a camera alignment  $\mathbf{R}_{hc}, \mathbf{t}_{hc}$  that minimizes the deviation of global gate positions taken so far from different locations.

After taking  $n$  valid images from different vehicle positions, a point cloud with  $n$  points is determined using the global gate translation from Eqs. 4 and 5 for each point. The error vector, denoted by  $\sigma = (\sigma_x, \sigma_y, \sigma_z)^T$ , represents the standard deviations of the point cloud in geodetic  $x$ ,  $y$  and  $z$ -direction.

Let the camera alignment be determined by three position elements  $x$ ,  $y$ ,  $z$  and three rotation angles  $\omega$ ,  $\phi$ ,  $\kappa$ . For fixed gate measurements  $\{\mathbf{R}_g, \mathbf{g}_i | i = 1, \dots, n\}$  and their corresponding vehicle positions  $\{\mathbf{R}_h, \mathbf{h}_i | i = 1, \dots, n\}$ , the point cloud of resulting gates is only dependent on the camera alignment, i.e.  $\sigma : \sigma(x, y, z, \omega, \phi, \kappa)$ . Tests have shown that the initial alignment translation ( $x, y, z$ ) can be measured with comparatively high accuracy and that the camera roll angle ( $\omega$ ) is of low importance to locate the gate. The gate position mainly depends on the camera pitch ( $\phi$ ) and yaw angle ( $\kappa$ ). Only they are optimized, and it is  $\sigma : \sigma(\phi, \kappa)$ . The optimal camera alignment will result in a minimum of  $\sigma$ , which is approximated using a multivariate Newton-Raphson iteration.

**Fig. 7** Minimizing the gate position estimation uncertainty (a) by finding the correct camera alignment (b)



The algorithm uses the Jacobian matrix  $\mathbf{J}_\sigma$  of  $\sigma$

$$\mathbf{J}_\sigma = \begin{pmatrix} \frac{\partial \sigma}{\partial \phi} & \frac{\partial \sigma}{\partial \kappa} \end{pmatrix} = \begin{pmatrix} \frac{\partial \sigma_x}{\partial \phi} & \frac{\partial \sigma_x}{\partial \kappa} \\ \frac{\partial \sigma_y}{\partial \phi} & \frac{\partial \sigma_y}{\partial \kappa} \\ \frac{\partial \sigma_z}{\partial \phi} & \frac{\partial \sigma_z}{\partial \kappa} \end{pmatrix}, \quad (16)$$

where the derivatives of single values  $\phi$  and  $\kappa$  are approximated using the difference quotients

$$\frac{\partial \sigma}{\partial \phi} \approx \frac{\sigma(\phi + t, \kappa) - \sigma(\phi, \kappa)}{t}, \quad \text{and} \quad \frac{\partial \sigma}{\partial \kappa} \approx \frac{\sigma(\phi, \kappa + t) - \sigma(\phi, \kappa)}{t}, \quad (17)$$

with a small  $t = 0.0001$ .

Now, the iterative approximation starts with  $\phi_0$  and  $\kappa_0$  from initially given camera alignment angles, and the  $n$ -th iteration is

$$\begin{pmatrix} \phi_{n+1} \\ \kappa_{n+1} \end{pmatrix} = \begin{pmatrix} \phi_n \\ \kappa_n \end{pmatrix} - \mathbf{J}_{\sigma,n}^\dagger \sigma(\phi_n, \kappa_n) \quad (18)$$

where  $\mathbf{J}_{\sigma,n}^\dagger$  is the Moore-Penrose pseudo inverse of the Jacobian, with the partial derivatives calculated for the single values  $\phi_n$  and  $\kappa_n$ . Termination conditions are a maximal number of iterations or  $\|\mathbf{J}_{\sigma,n}^\dagger \sigma(\phi_n, \kappa_n)\| < \varepsilon$  with a threshold parameter, e.g.  $\varepsilon = 0.001$ .

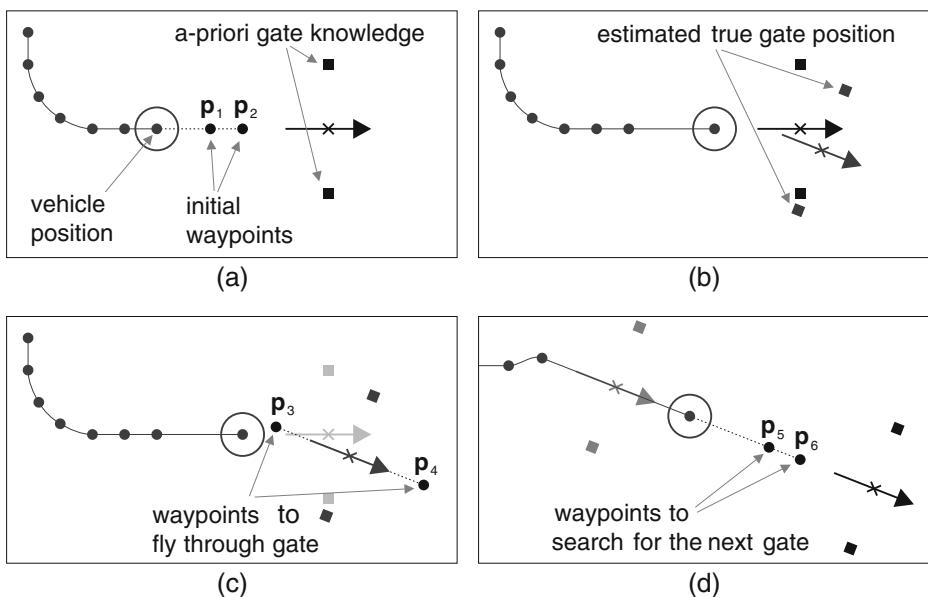
Finally, a new camera alignment  $x, y, z, \omega, \phi_{n+1}, \kappa_{n+1}$  is used to calculate updated  $\mathbf{R}_{hc}$  and  $\mathbf{t}_{hc}$  and a target position as done in the Eqs. 4 and 5.

## 5 Flight Mission Execution

The flight is started with approximate knowledge about gate positions from an obstacle map. To cope with requirements from typical scenarios where it is assumed that a vehicle is moving on a pre-planned path from a higher-ranking task plan, the presented approach avoids separate calibration flights or the generation of specified flight paths in order to get a better view on the obstacle or to improve vision-based tracking results as proposed in [23]. Here, the vehicle shall detect the gate within a normal flight, for example while flying straight ahead towards it with some translational and rotational deviation. Hence, the basic maneuver is to fly to a waypoint next to an expected gate, to re-locate the gate with the described method and to fly through. Complex object scanning flights are not required. To handle multiple gates, the maneuver can be repeated. Another aspect concerns the used hardware. When the vehicle is nearby the gate, its posts are out of view because of the limited field of view. With that, the best gate crossing path has to be calculated before the vehicle approaches too close.

Figure 8 illustrates the process in detail. The pictures show the scenario from bird's eye view with the vehicle, its already flown path, waypoints, and the gate positions. Each gate is represented by its two posts, center and pass direction.

Two initial waypoints  $\mathbf{p}_1$  and  $\mathbf{p}_2$  are placed in front of a gate with different distances (Fig. 8a). For the used vehicle and camera configuration, these waypoints are 30m and 10m in front of the gate where a good visibility is expected. The vehicle is instructed to fly to  $\mathbf{p}_1$  and approach to the gate up to the nearer waypoint  $\mathbf{p}_2$ . On this way, images are processed to search for the true gate position (Fig. 8b).



**Fig. 8** Mission execution plan to fly through gates. Initial path (a), detecting the true gate position (b), generating fly-through waypoints (c), continuing with the next gate (d)

If the gate is visible and its position is estimated with a high confidence, two new waypoints  $p_3$  and  $p_4$  are added at a distance of  $d = 10m$  in front of and behind the gate. They will guide the vehicle blindly through the gate (Fig. 8c). Based on a estimated gate position  $\mathbf{t}$  and direction  $\alpha$  (Eq. 6), it is

$$\mathbf{p}_{3/4} = \mathbf{t} \pm d \cdot \begin{pmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{pmatrix}. \quad (19)$$

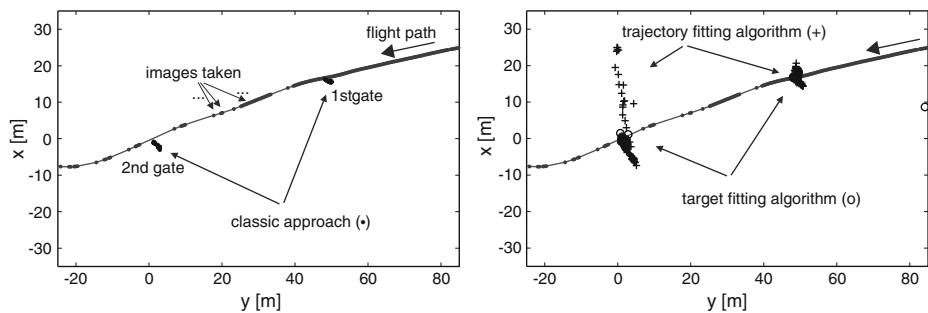
If the search is not successful, another attempt is designated with a flight back to  $p_1$ .

After a successful attempt, the mission is either completed or the next gate is searched (Fig. 8d). The initial map is transformed by using the deviation between the initial and the estimated positions of the first gate. New waypoints  $p_5$  and  $p_6$  are set and the mission continues with the same procedure used for the first gate.

## 6 Flight Test Results

### 6.1 Preliminary Manual Flights

Image processing, gate position and camera alignment estimation is developed and optimized using image sequences taken from manual helicopter flights. Different lighting conditions, camera shaking on the vehicle, navigation uncertainties and the realistic setup provide a case close to full autonomous flight. The scenario contains of two gates placed on a flight field with a distance of 50 m. The quality of gate detection

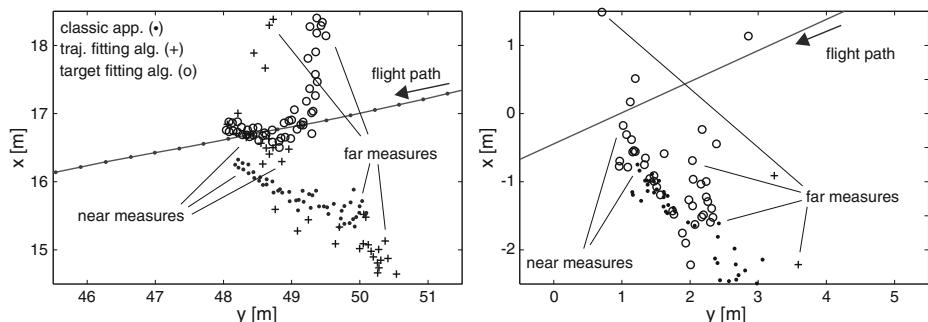


**Fig. 9** Flight trajectory (blue) and estimated gate positions ( $x, y$ ) using a known camera alignment (left, dots), trajectory fitting (right, crosses) and the target fitting algorithm (right, circles)

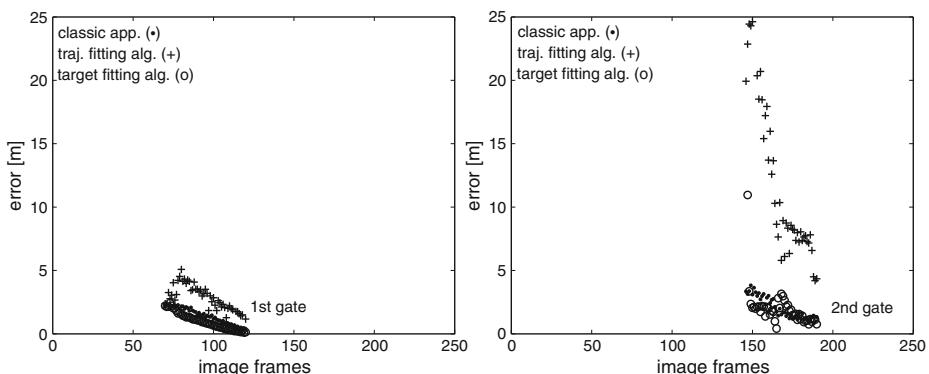
methods is determined with respect to the flight trajectory assuming that the vehicle flew through the gate centers.

The evaluation focuses on the accuracy of gate orientation estimation at any time. The paper presents processing results for most of the images where a gate is recognized. Figures 9–12 show plots of different measure values that are required for gate detection. Results are produced as follows:

- Flight speed is between 4 and 8 m/s, manually controlled. The camera settings are  $1024 \times 768$  pixels, 24 bit color, 15 frames per second.
- Both gate positions are calculated independently since a sequence cut is automatically performed when the first gate leaves field of view during flight. The results are projected to frame numbers of the overall image sequence with both gates.
- For each gate, sequences with a minimum number of 20 frames are used to estimate the gate position. For each new image, an updated gate position is estimated.
- Maximal sequence size is 50. Older images are removed from the sequence and are not taken into account for position estimation.

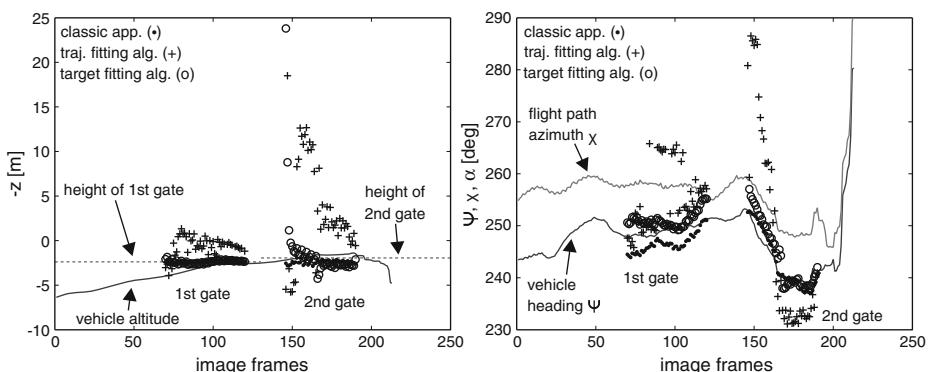


**Fig. 10** Extract of Fig. 9 by focusing on the first gate (left) and second gate (right), respectively. Both plots show all three algorithm results



**Fig. 11** Gate estimation errors of the first (*left*) and second (*right*) gate during the image sequence, i.e. Euclidean distance between estimated gate position and the closest trajectory point

Figure 9 shows a bird's eye view on the flight trajectory and the estimated gate positions using the different algorithms. Flight direction is from the right to the left. The left plot shows the processing results when using the initially given camera alignment. All estimated gate positions are located left to the flight path, but this error decreases while approaching to the gate as later shown. The right plot shows the results when applying the improved algorithms. While the first gate was detected with good quality with both algorithms, the performance of the trajectory fitting algorithm is highly reduced at the second gate (plus sign markers). Figure 10 shows the same results with a higher resolution, focusing on both gates separately. The position estimation is better at the first gate, which is supposed to be due to a larger number of valid images taken in the vicinity of the gate, which means at a distance between 7 and 10 m. Measuring the gate from lower distances, the estimation gets more accurate and nearly converges to the flown trajectory until it gets out of view. Both trajectory and target fitting algorithms have a good performance with an error of less than 0.5 m to the reference flight path. In contrast to that, this was not achieved when



**Fig. 12** Flight altitude ( $z$ ) and the estimated gate heights over the image sequence (*left*). Vehicle heading ( $\Psi$ ), flight path azimuth ( $\chi$ ) and estimated gate heading ( $\alpha$ ) during the image sequence (*right*)

using known camera alignment. Especially the target fitting algorithm provides stable results from image to image, which leads to the best qualification for autonomous flights. The performance of all algorithms is decreased at the second gate. Here, the target fitting algorithm is also best qualified compared to the others. The position estimation errors in all three translational degrees of freedom are summarized in Fig. 11. From this point of view, the target fitting algorithm is only slightly better than the classic approach while the trajectory fitting approach is significantly worse.

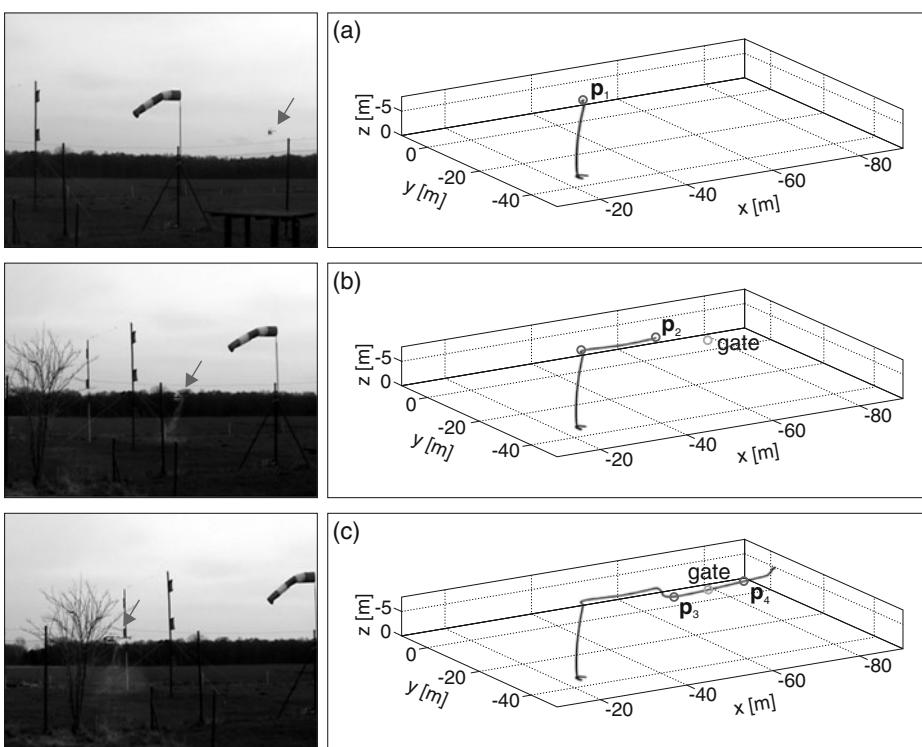
Figure 12 shows estimation charts of height and gate direction. The results support the hypothesis that trajectory fitting has rather poor performance compared to the other approaches. Height estimation is not suitable for flights through a gate using this algorithm. Contrary to that, the desired flight altitude is estimated very precisely when only using the initially known camera alignment or the target fitting algorithm. As a drawback, the estimated gate direction, i.e. the desired flight path azimuth, has always some errors since the ideal case would produce a straightly horizontal spread of the markers. The observed deviation of five degrees for the first gate should be unproblematic. Especially at the first gate, the estimated direction is almost constantly increased by five degrees when comparing the target fitting algorithm with the classic approach. This means that the filter detects a horizontal camera alignment error of five degrees which is considered in the improved gate position and orientation estimation.

As a final conclusion of this preliminary evaluation, a target calculation works fairly for an accurately hand-measured camera alignment and would be at least sufficient to test autonomous flights. To be independent on precise manual measurements and to cope with alignment changes over time due to inertial sensor drift, a camera misalignment compensation becomes useful. As presented, misalignment compensation can minimize target errors and especially increase the rotational accuracy. The trajectory fitting algorithm works in some cases, but has a high risk of failures and will not be used. The target fitting algorithm has turned out to be best qualified and is installed at the onboard image processing computer to be used for all autonomous flights presented in this paper. However, for practical use, it might be sufficient to perform this calibration only from time to time in order to compensate drift.

## 6.2 Autonomous Flights

Afterwards, autonomous helicopter flights are performed. The setup includes flight control, waypoint navigation, and the described gate crossing approach. To increase safety, the helicopter is advised to fly slowly with a speed of 1.5 m/s during the automated gate approach and crossing phase. Camera settings are not changed in comparison to the manual flights. Figure 13 illustrates exemplary how an execution of the flight mission works. The helicopter is directed to the first waypoint (a) in order to start the gate search. After successful search (b), the vehicle is directed to adjust the desired flight path by inserting the two waypoints in front of and behind the gate. Afterwards, the vehicle is precisely guided through these waypoints (c) which results in gate crossing.

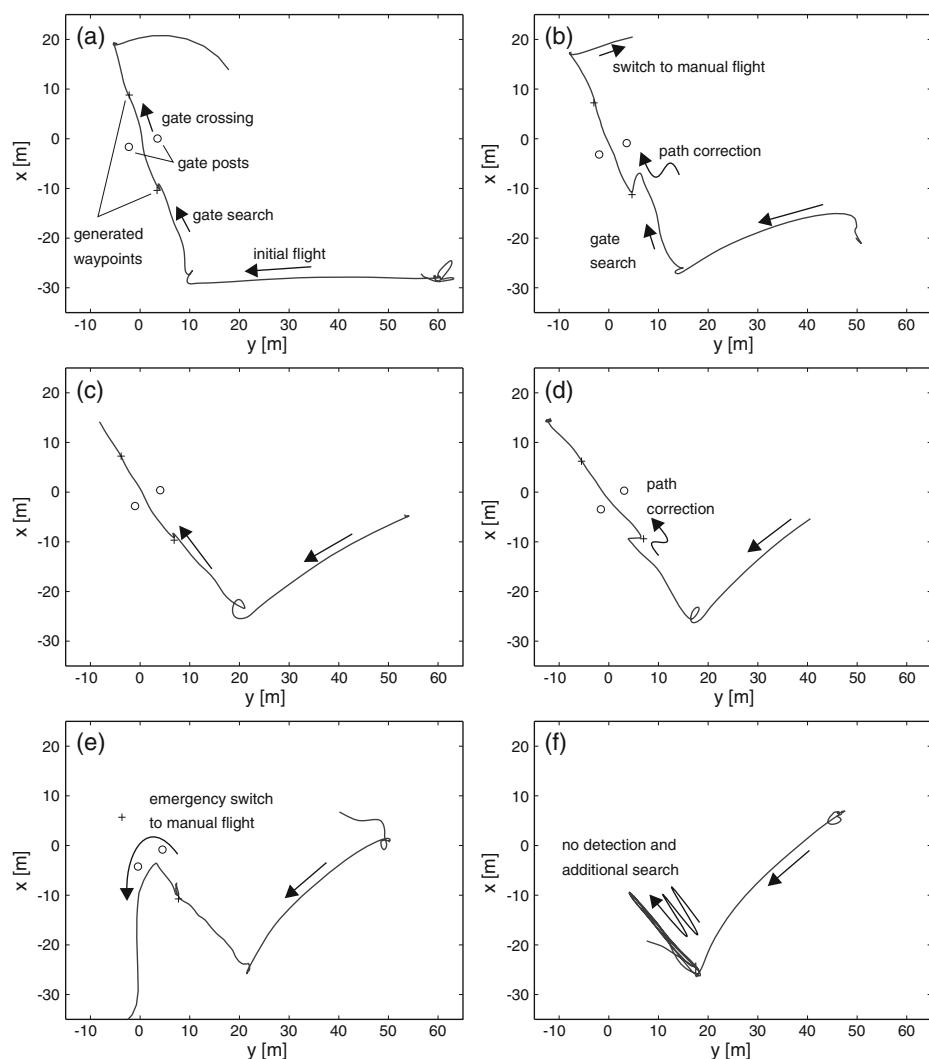
Additional two-dimensional plots in Fig. 14 show the quality of gate detection from six flights with different translation and rotation to the gate while approaching. It shows flight trajectories and the gate posts and waypoints based on the estimated



**Fig. 13** Snapshots of an autonomous mission with pictures of the helicopter at different time steps and a corresponding flight path. Flight up to the first waypoint to start gate search (**a**), reaching the second waypoint (**b**) and after gate passage (**c**)

target position. The navigation solution is initialized while the helicopter is standing at the ground at the center between the two posts, heading perpendicular to the gate. Initial navigation coordinate is  $(x, y, z, \alpha) = (0, 0, 0, 340^\circ)$ . Since the helicopter shall pass the gate in a height of three meters, the optimal gate crossing waypoint is  $(x, y, z, \alpha) = (0, 0, -3, 340^\circ)$  in the ideal case of no sensor drift. In addition to the plots, the determined waypoint coordinates are listed in Table 1 below.

The graphics from (a) to (d) show for successful flights with different a-priori settings for the initial gate position. Plot (a) shows the case where the optimal waypoint is given. As visible, the vision-based detection method provides nearly the same coordinate, and little path corrections are made before the vehicle crosses the gate. Note that an exact measurement of the gate center can be different from the initial given optimum  $(x, y, z, \alpha) = (0, 0, -3, 340^\circ)$  due to navigation sensor drift. In the flight presented in (b), the a-priori waypoint is displaced by three meters to the right. Hence, larger corrections are made to guide the vehicle through the gate. The flight path azimuth is similar to the direction of plot (a). As visible and contrary to plot (a), a flight without gate detection would hit the right gate post. Plot (c) shows flight results where the a-priori given waypoint is rotated by 15 degrees, and in plot (d), the initial waypoint is rotated and translated by three meters to the left. Gate crossing was successful, but as visible, the measured gate direction differs significantly



**Fig. 14** Plots of vehicle trajectories, and estimated gates, illustrated by calculating the gate pole coordinates and the waypoints guiding the vehicle through the gate. Successful crossings (a) to (d), and error cases (e) and (f)

from the results in (a) and (b). This direction is incorrect and the helicopter flies transversely through the gate. Errors in the gate direction estimation become visible.

Two other typical errors are presented in (e) and (f). In (e), the gate position was estimated incorrectly but treated as correct by the system. Here, the desired flight height is too large. The autonomous flight is interrupted by the safety pilot to avoid a possible crash. The plot in (f) does also show no completed gate crossing, but this case can be treated as a success in general. No confident gate position is obtained, presumably due to bad lighting conditions, wrong looking direction or frame drops. In this case, the mission execution plan guides the vehicle back to the starting point

**Table 1** Gate coordinates from flights shown in Fig. 14

No.	x [m]	y [m]	z [m]	$\alpha$ [deg]
(a)	-0.79	0.62	-2.46	343.67
(b)	-2.02	0.83	-3.22	335.71
(c)	-0.93	2.00	-3.88	327.46
(d)	-1.58	0.75	-2.13	321.30
(e)	-2.53	2.05	-5.69	325.26
(f)	n/a	n/a	n/a	n/a

and starts additional attempts to search the gate. Since no gate is detected, the flight is canceled automatically after a maximum number of attempts.

Several autonomous flights are performed in addition to those from the presented graphics. The final result is that from 16 flights, 13 gate crossings were successful. In two cases, no gate was detected and the mission was automatically canceled. In only one case, the vehicle had to be stopped manually as presented.

The results show that autonomous flights through gates can be performed with the presented image processing and mission execution strategy. Image processing functionality is shown by the results of Section 6.1. Additionally, the overall integration of image processing, mission execution and flight control is proven with the presented autonomous flights.

## 7 Conclusion and Future Work

To address obstacle avoidance problems that appear for example in urban environments with limited free space, this paper presents a method to guide an unmanned aerial vehicle through narrow gates. The approach is designed to run on auto-piloted aerial robots and consists of vision-based gate detection, camera misalignment compensation, global navigation-referenced gate localization, and flight mission execution steps.

The algorithms run completely onboard the vehicle and provide key enabling techniques for autonomous navigation in complex environments. Gate recognition is proven with image and navigation data of outdoor helicopter flights. Testing focus are camera misalignment compensation methods developed throughout this paper. The overall approach is integrated into the onboard system and tested with autonomous flights that show the functionality. Results are successful gate crossings without collision. The flight had to be canceled manually in one out of 16 flights. Hence, the functionality is shown from a scientific perspective, but the failure rate is obviously too high for applied use. The robustness has to be increased in future applications, for example with additional sensors like laser range finders or multiple camera systems. Although a successful gate crossing is shown, some other errors concern the estimated gate direction, presumably due to bad perspective transformations of the specific planar pattern. Photogrammetric algorithms may provide more accurate results on objects with more feature points or non-planar three-dimensional objects with a “depth” where the perspective is better visible.

The presented approach requires previous knowledge about expected obstacles. Hence, upcoming research is going to extend the presented algorithm to handle a gate, window, or other small passages where only few attributes are given. For example, it is a challenge to fly through a large variety of rectangular passages

where orientation and flight direction are obtained through the visible perspective projection. Furthermore, absolute obstacle sizes are not required if not only rotation and translation, but also scaling between navigation and image-based trajectories can be estimated accurately. This case does not require range sensors, and monocular vision will be capable of performing basic obstacle recognition.

The used hardware with its specific camera setup implicates that obstacles get out of view when the vehicle comes closer to them. Hence, the presented algorithms are designed to allow blind flights through the passage after measuring it. For best results especially to measure obstacles precisely from near distances, a horizontal field of view of 180 degrees is required. Special camera lenses, multiple cameras or other sensors like the mentioned laser scanners are qualified for that task. For future work, this improved sensor setup can decrease error-proneness. Since the gate will be always visible, this concept allows the implementation of visual servoing concepts with a tightly-coupled vision-based control approach. With that, the vehicle could change its movements while approaching to the obstacle and to fly directly through a passage without previous stopping. The goal is to enable agile flight maneuvers through cluttered environments using helicopters and proceedingly with fixed-wing aircraft also.

## References

1. Adolf, F., Andert, F., Lorenz, S., Goermann, L., Dittrich, J.: An unmanned helicopter for autonomous flights in urban terrain. In: Kröger, T., Wahl, F.M. (eds.) *Advances in Robotics Research*, pp. 275–285. Springer, Berlin (2009)
2. Andert, F., Goermann, L.: A fast and small 3-D obstacle model for autonomous applications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2883–2889 (2008)
3. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(5), 698–700 (1987)
4. Bernatz, A., Thielecke, F.: Navigation of a low flying vtol aircraft with the help of a downwards pointing camera. In: AIAA Guidance, Navigation and Control Conference (2004)
5. Brown, D.C.: Close-range camera calibration. *Photogramm. Eng.* **8**, 855–866 (1971)
6. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Vision-based odometry and slam for medium and high altitude flying uavs. *J. Intell. Robot. Syst.* **54**(1–3), 137–161 (2009)
7. Corke, P., Lobo, J., Dias, J.: An introduction to inertial and visual sensing. *Int. J. Rob. Res.* **26**(6), 519–535 (2007)
8. Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and mapping (slam) problem. *IEEE Trans. Robot. Autom.* **17**(3), 229–241 (2001)
9. Dittrich, J.S., Bernatz, A., Thielecke, F.: Intelligent systems research using a small autonomous rotorcraft testbed. In: 2nd AIAA Unmanned Unlimited Conference, Workshop and Exhibit (2003)
10. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing* **24**(6), 381–395 (1981)
11. Hrabar, S.: 3D path planning and stereo-based obstacle avoidance for rotorcraft uavs. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 807–814 (2008)
12. Lorusso, A., Eggert, D.W., Fisher, R.B.: Comparison of four algorithms for estimating 3-D rigid transformations. In: British Machine Vision Conference, pp. 237–246 (1995)
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
14. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence, pp. 674–679 (1981)

15. Ludington, B., Johnson, E., Vachtsevanos, G.: Vision based navigation and target tracking for unmanned aerial vehicles. In: Valavanis, K.P. (ed.) *Advances in Unmanned Aerial Vehicles*, pp. 245–266. Springer, Dordrecht (2007)
16. Nüchter, A., Lingemann, K., Hertzberg, J., Surmann, H.: 6D slam with approximate data association. In: 12th International Conference on Advanced Robotics, pp. 242–249 (2005)
17. Scherer, S., Singh, S., Chamberlain, L., Saripalli, S.: Flying fast and low among obstacles. In: IEEE International Conference on Robotics and Automation, pp. 2023–2029 (2007)
18. Sharp, C.S., Shakernia, O., Sastry, S.S.: A vision system for landing an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation, pp. 2793–2798 (2002)
19. Shim, D.H., Chung, H., Sastry, S.S.: Conflict-free navigation in unknown urban environments. *IEEE Robot. Autom. Mag.* **13**(3), 27–33 (2006)
20. Strelow, D., Singh, S.: Optimal motion estimation from visual and inertial data. In: IEEE Workshop on Applications of Computer Vision, pp. 314–319 (2002)
21. Thrun, S.: Robotic mapping: A survey. In: Lakemeyer, G., Nebel, B. (eds.) *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann, San Francisco (2002)
22. Watanabe, Y., Calise, A.J., Johnson, E.N.: Vision-based obstacle avoidance for uavs. In: AIAA Guidance, Navigation and Control Conference and Exhibit (2007)
23. Watanabe, Y., Fabiani, P., Mouyon, P.: Research perspectives in uav visual target tracking in uncertain environments. In: Workshop on Visual Guidance Systems for Small Autonomous Aerial Vehicles, IEEE IROS Conference (2008)

# Exploring the Effect of Obscurants on Safe Landing Zone Identification

Keith W. Sevcik · Noah Kuntz · Paul Y. Oh

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 28 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** Manned rotorcraft are often employed in harsh environments and difficult terrain that are inaccessible to other craft. Conversely, robotic rotorcraft are operated in controlled settings, often at safe, high altitudes. Missions such as cargo delivery, medevac and fire fighting are unachievable because of unpredictable adverse environmental conditions. To enable UAVs to perform these missions, the effects of obscurants on UAV sensor suites and algorithms must be clearly understood. This paper explores the use of a laser range finder to accomplish landing zone identification in unknown, unstructured environments. The ability to detect a landing zone in environments obscured by smoke is investigated. This is accomplished using a design methodology of testing and evaluating in a controlled environment followed by verification and validation in the field. This methodology establishes a concrete understanding of the sensor performance, thereby removing ambiguities in field tests.

**Keywords** Evaluating guidance algorithms · Verifying performance · Navigation and control

## 1 Introduction

Helicopters and other manned rotorcraft often perform missions that can not be accomplished by other craft. Their ability to fly and hover allows helicopters to access

---

K. W. Sevcik · N. Kuntz · P. Y. Oh (✉)  
Department of Mechanical Engineering and Mechanics, Drexel University,  
3141 Chestnut Street, Philadelphia, PA, USA  
e-mail: paul@coe.drexel.edu

K. W. Sevcik  
e-mail: kws23@drexel.edu

N. Kuntz  
e-mail: nk752@drexel.edu

remote terrain amongst obstacles like buildings, poles and trees. Pilots are able to perform missions such as cargo delivery, search and rescue, and fire fighting even when faced with thick smoke and brown-out conditions.

Conversely, robotic rotorcraft are often confined to well structured, safe environments. Missions such as surveillance are performed at high altitudes, safe from the threat of obstacles on the ground. In other tasks such as building inspection, the rotorcraft is often controlled by a pilot who has a constant line of sight to the aircraft. When these tasks are made autonomous, the ambient conditions are often chosen to be idyllic without adverse weather conditions. To enable UAVs to perform the missions carried out by manned helicopters, the effects of obscurants on UAV sensor suites and algorithms must be clearly understood.

In evaluating this problem, a baseline capability must be identified for testing. Landing zone identification is an essential component to many rotorcraft missions. The problem extends to tasks such as cargo delivery, medevac and search and rescue among others. Landing zone identification is a thoroughly researched problem, and many well developed solutions already exist. Furthermore, the capabilities necessary for landing zone detection are extensible to other core capabilities, such as obstacle avoidance and mapping.

Landing zone identification requires a sensor suite capable of mapping the ground beneath the helicopter. Selecting a sensor suite for this task is complicated by the nature of the environments in which these robots operate. Areas of interest such as urban landscapes are cluttered with obstacles. Large structures like buildings can be easily detected by many different types of sensors. Small obstacles like wires and sparse obstacles such as trees and bushes can be more difficult to detect. These pose great risk to rotorcraft because the exposed rotor is easily damaged, even by small objects. This potential hazard demands a sensor suite capable of resolving small obstacles at far ranges.

Furthermore, the robot may be subject to adverse ambient conditions. Particles from debris or smoke from fires can obscure the field of view of a sensor suite. When a rotorcraft is landing, dust is often scattered, at times creating a brown out effect. To ensure that performance is robust and reliable, these effects must be quantified when addressing the issue of landing zone identification.

As a Future Combat Systems: One Team member, we have gained extensive experience designing sensor suites for robots flying in near-Earth environments. The Future Combat Systems (FCS) Class II program focused on building a UAV to fly missions in near-earth environments such as urban terrain and forests. This project identified a few fundamental requirements for sensor suites.

The sensor must detect a wide range of obstacles. In urban terrain, object size and composition can vary drastically, from buildings to telephone poles to thin wires and clothes lines. In particular, sparse objects such as trees and bushes are troublesome to detect.

The UAV will also encounter a variety of adverse environmental conditions, such as the scenario depicted in Fig. 1. Smoke from fires or dust from down-wash can hinder the performance of the UAVs sensor suite. Other environmental factors such as rain, fog, and varied lighting can further degrade performance. The selected sensor must adequately address these issues.

Our experiences in sensor suite design revealed that scanning laser range finders are the best suited sensor to meet these criteria. Preliminary experiments against the

**Fig. 1** The SR100 robotic rotorcraft hovers over an unsafe landing site. A laser range finder mounted to the bottom of the craft is tested for its performance in smoke



criteria stated above showed them to out perform common sensors such as sonar, computer vision and optic flow.

The biggest attraction of these sensors is their high fidelity and wide field of view. Their range is comparable if not better than many traditional sensors. Laser range finders are also able to clearly detect many different objects including sparse objects such as trees and bushes. Additionally, they are robust to varied lighting, encountering difficulties only in extreme conditions such as direct sunlight measuring over 10,000 lux.

The major drawback to laser range finders is their sensitivity to obscurants present in the air. Rain can cause reflections that appear to be thin obstacles. Additionally, particulate matter such as fog, smoke and dust attenuate the laser beam and cause back-scatter, making some obstacles undetectable. This detriment can hinder the operational capabilities of the UAV.

Extensive work has been done to model the effect of obscurants. A model for the effect of different kinds of smoke and various other obscurants is presented in [1]. Brinkworth [2] defines general equations for determining attenuation and back-scatter from a light source. Such models can be used to correct for obscurants in laser range measurements.

Other researchers have directly measured the effects of fog and smoke on lasers. Arshinov et al. [3] conclude that back-scatter from lasers is more affected by the amount of smoke and fog than the wavelength of the laser. Even though these effects are well documented and measured, there has been relatively little work towards applying the results to the problem of building terrain maps and identifying safe landing zones.

Laser range finders have proven their capability for mapping cluttered terrain. In [4] a laser range finder mounted to a rotorcraft is used to map buildings, bushes and trees. Previous work has also demonstrated the ability to navigate rotorcraft based on these maps. A LADAR sensor was used to map urban terrain in [5]. Obstacles as small as 6mm diameter wires were successfully detected. This map was then

used to autonomously guide the craft through the environment. These experiments noticed effects from dust, however the research did not extend to characterizing and correcting for these effects. Also, these experiments did not explore using the LADAR generated maps to land the vehicle.

Landing rotorcraft autonomously has been accomplished before. Saripalli et al. [6] successfully demonstrated landing a rotorcraft utilizing GPS information and a vision based approach. The same authors demonstrated the ability to land a rotorcraft on a moving target in [7] using similar techniques. Both of these results rely on predetermined GPS locations where well marked fiducials are placed on the ground. Such well structured scenes would not necessarily be available in a perch-and-stare or cargo delivery missions.

Algorithms for safe landing zone identification have also been presented and tested. Both [8] and [9] present machine vision based approaches for identifying safe landing zones. While the fundamental approach can be applied to any terrain map, the specific algorithms assume images, which could be degraded by obscurants. In [10], an algorithm is presented for identifying safe landing zones using a laser range finder. This body of research shows that the individual components exist to detect landing zones while accounting for adverse conditions.

This paper examines the effects of obscurants on identifying safe landing zones. A method for determining a safe landing zone using a LADAR sensor on board a robotic rotorcraft is presented. This method is comprised of a step in which the terrain map is reconstructed from laser and position data, followed by a step in which the safest landing zone in the terrain map is identified. A SICK LMS200 laser range finder was tested and evaluated for varying densities of smoke inside a UAV testing facility. The sensor model with and without obscurants was determined from these experiments. Verification and validation was performed on board an SR100 robotic helicopter. Results from these experiments are presented.

Section 2 describes the approach used to generate terrain maps and identify landing zones. Section 3 describes the testing and evaluation of the SICK laser utilizing SISTR, a Systems Integrated Sensor Test Rig. The robotic platform used for verification and validation is described in Section 4. Preliminary experimental results are given in Section 5. Finally, conclusions and future work are discussed in Section 6.

## 2 Algorithms

To detect a safe area to land, the robot must first generate a map of its environment. This terrain map is generated using the laser scans and pose measurements of the aircraft. Vibrations from the helicopter and inherent noise/drift in the sensors can seriously degrade the quality of the terrain map. To recover a usable map of the terrain, a mapping algorithm must be applied that considers noise in both measurements. The resulting terrain map is often comprised of large flat areas with both large and small obstacles. The landing zone algorithm must find flat, obstacle free terrain with a large enough area to fit the rotorcraft. The following sections describe the algorithms utilized to accomplish these steps.

## 2.1 Mapping

To generate a terrain map, laser scans must be fused with relatively noisy pose measurements. This is accomplished using an implementation of the process presented in [4]. The fundamental concepts and their application are presented here in brief.

This algorithm produces a 3D map of the environment given noisy pose and terrain measurements. To find the corrected pose, a probabilistic model is constructed. This model is comprised of: the probability of the pose measurement, the probability of differential pose measurements, and the probability of consecutive scan alignment.

The probability of pose measurement is modeled as the probability of measuring the pose given the corrected pose. The system is taken to be 6 degrees of freedom, namely the 3 Cartesian coordinates and rotations about those axes. Their measurement at the current time step is the vector  $y_t$ , while the algorithm solves for the corrected pose  $x_t$ . Given the measurement covariance  $A$ , the probability of  $y_t$  given  $x_t$  as presented in [4] is then:

$$p(y_t|x_t) \propto \exp\left(-\frac{1}{2}(y_t - x_t)^T A^{-1}(y_t - x_t)\right) \quad (1)$$

The method also utilizes a differential model. Typically, the sensors onboard an aircraft measure rotational and translational rates. The pose is recovered through integration, making it susceptible to drift. The differential model is less affected by this error. Given  $D$ , the covariance of differential measurements, the differential model as derived in [4] is:

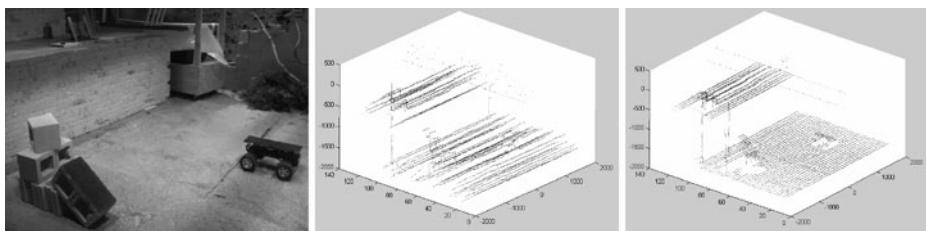
$$p(\Delta y_t|\Delta x_t) \propto \exp\left(-\frac{1}{2}(\Delta y_t - \Delta x_t)^T D^{-1}(\Delta y_t - \Delta x_t)\right) \quad (2)$$

where  $\Delta y_t = y_t - y_{t-1}$  and  $\Delta x_t = x_t - x_{t-1}$ . As differential measurements are more accurate than absolute measurements, the covariance matrix  $D$  should represent a Gaussian with smaller standard deviation than  $A$ .

The final portion of the model is a representation of the likelihood of a scan. Rather than representing individual features as states as in traditional SLAM, the implementation in [4] models the consistency between consecutive scans as:

$$\begin{aligned} p(z_t|x_t, x_{t-1}, z_{t-1}) \\ \propto \prod_i \exp\left(-\frac{1}{2} \min\left[\alpha, \min_j (z_t^i - f(z_{t-1}^j, x_{t-1}, x_t))^T B^{-1} (z_t^i - f(z_{t-1}^j, x_{t-1}, x_t))\right]\right) \end{aligned} \quad (3)$$

The goal of this model is to align points from the current scan with points from the previous scan. A point in the current scan  $z_t^i$  is compared to all points from the previous scan  $z_{t-1}$ . The function  $f$  maps a point from the previous scan  $z_{t-1}^j$  into the local coordinate system of the current scan  $z_t$ . The inner minimization identifies a point from the previous scan that is closest to the point from the current scan. The outer minimization thresholds this alignment to allow for local inconsistencies such as those from sparse objects. The matrix  $B$  is the measurement covariance.



**Fig. 2** The sensor was scanned through a mock urban environment (*left*). Gaussian noise was added to the pose measurement to simulate data gathered from a helicopter (*center*). The terrain map was then reconstructed using the algorithm described (*right*)

Equations 1, 2 and 3 can be combined to form the probabilistic model for the entire problem [4]:

$$p(y_t|x_t)p(\Delta y_t|\Delta x_t)p(z_t|x_t, x_{t-1}, z_{t-1}) \quad (4)$$

The map and pose are recovered by finding the pose that maximizes this likelihood, or by minimizing the negative log likelihood given by [4]:

$$\begin{aligned} & \text{const} + \frac{1}{2}((y_t - x_t)^T A^{-1}(y_t - x_t) + (\Delta y_t - \Delta x_t)^T D^{-1}(\Delta y_t - \Delta x_t) \\ & + \sum_i \min[\alpha, \min_j (z_t^i - f(z_{t-1}^j, x_{t-1}, x_t))^T B^{-1}(z_t^i - f(z_{t-1}^j, x_{t-1}, x_t))]) \end{aligned} \quad (5)$$

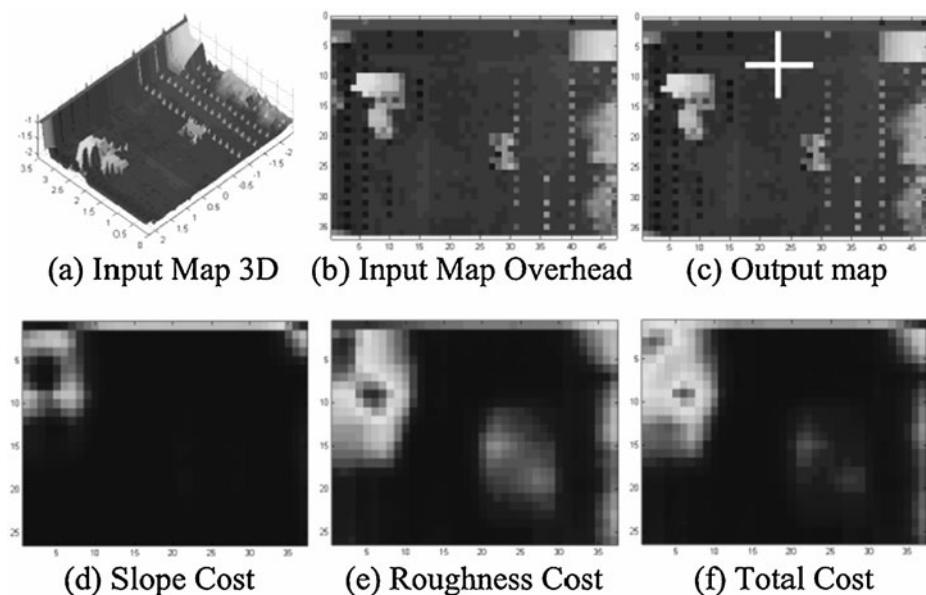
This minimization is found by first minimizing to associate points from the current scan with those from the previous scan, and then performing hill-climbing to determine the pose that minimizes the negative log likelihood. These steps can be iterated until the negative log likelihood falls within a threshold.

This algorithm was tested on a data set gathered inside a mock urban environment. A test was conducted in which the laser scanner was suspended approximately 2m above the ground and oriented to face the ground. The sensor was then traversed through the environment using a robotic gantry. The position of the laser was measured from the gantry's encoders. These conditions were well controlled and the measurements were very accurate, unlike those of a rotorcraft. To simulate noisy pose measurements from a helicopter, Gaussian noise was added to the position data. The results are shown in Fig. 2.

As can be seen, the algorithm successfully recovers the terrain map in the form of a point cloud. The algorithm is able to line up scans and correct the small deviations between measurements. At the same time, it ignores the large deviations, recognizing the discontinuity is actually the ledge of a building. Obstacles such as the truck and cinder blocks are clearly recovered. Even small features are resolved, such as the ridge in the floor from overlapping floor mats. This terrain map can now be used to detect a safe landing zone.

## 2.2 Safe Landing Zone ID

The algorithm presented in [10] provides robust detection of a safe landing zone based on the input of a point cloud terrain map from a LADAR scanner (Fig. 3).



**Fig. 3** A terrain image is generated from the LADAR point cloud map. A cost map is then calculated based on the slope of the terrain and the local roughness. The safe landing zone, marked with a cross, is determined as the lowest cost area that fits the helicopter rotor diameter. **a** Input map 3D. **b** Input map overhead. **c** Output map. **d** Slope cost. **e** Roughness cost. **f** Total cost

This algorithm parameterizes a safe landing zone based on the slope of the landing area and the surface roughness. Costs are assigned to the terrain based on these factors, and the lowest cost area which fits the helicopter rotor is selected. Our implementation of this algorithm is described below.

Because of the design of the laser scanner, the resulting point cloud is an irregularly spaced sampling of the scanned surface. A safe landing zone algorithm that uses this data would be intrinsically complicated and resource intensive. To simplify the implementation, the point cloud map is first converted into an image containing regularly spaced pixels. First the size of the grid cells must be determined. The width of each grid cell,  $C_w$ , is based on the angular resolution of the scanner  $\theta$  and the average range to the surface  $R$ . The total width of the grid,  $G_w$ , is based on the field of view of the scanner  $f$  and the average range to the surface. The width of each cell, the total width of the grid and the total number of cells  $n$  is then:

$$C_w = 2R \tan(\theta/2) \quad (6)$$

$$G_w = 2R \tan(f/2) \quad (7)$$

$$n = G_w / C_w \quad (8)$$

The cell height was taken to be the same as the cell width. The total grid height  $G_h$  is determined from the distance traversed in the direction perpendicular to the scan

plane. This formulation ensures there will not be multiple points per grid cell. After determining the cell and grid sizing, the  $(x, y)$  coordinates of data points in the point cloud must be transformed to  $(r, c)$  coordinates in the grid. This is accomplished using the relation [10]:

$$(r, c) = (y/C_w + G_h/2, x/C_w + G_w/2) \quad (9)$$

The value of each grid cell is based on the z-coordinate of the points. Interpolation is used to define cells that fall in-between points. The resulting 2D array is analogous to a grayscale image who's pixel values correspond to the height of the terrain. This image is the raw elevation map.

To perform safe landing zone identification, the elevation map is separated into a surface roughness map and a landing incidence angle map. Both these maps require that an underlying smoothed surface first be determined. This surface is formed by fitting square planes the size of the helicopter rotor diameter to the terrain map. Planes are represented as [10]:

$$\mathbf{n} \cdot \mathbf{x} + d = 0 \quad (10)$$

Where the fitted plane at cell  $\mathbf{x} = (x, y, z)$  is described by  $(\mathbf{n}, d) = (n_x, n_y, n_z, d)$ . These planes are fitted with an increment of 1/8 the rotor diameter between planes. The resolution for the position of the chosen landing zone is therefore 1/8 that of the helicopter rotor diameter. Smaller increments could be chosen to make this position more precise. This would come at the cost of processing speed. Due to error in the accuracy of the helicopter's pose measurement, the chosen resolution is believed to be sufficiently accurate.

The landing incidence angle  $\alpha$  is calculated using the fitted planes and the geodetic normal of the surface  $n_g$  [10]:

$$\alpha = \cos^{-1}(\mathbf{n} \cdot \mathbf{n}_g / \|\mathbf{n}\| \|\mathbf{n}_g\|) \quad (11)$$

These fitted planes are also used to calculate the smoothed elevation map, where the smooth elevation  $z_s$  is given by [10]:

$$z_s = -(n_x x + n_y y + d)/n_z \quad (12)$$

Once the smoothed surface is generated, the roughness map can be determined. The roughness map  $R(r, c)$  is calculated by subtracting the smoothed elevation map  $Z_s(r, c)$  from the original elevation map  $Z(r, c)$  and taking the mean [10]:

$$R(r, c) = |Z(r, c) - Z_s(r, c)| \quad (13)$$

A safe landing zone is chosen from a cost map. A cost is calculated for a region based on a weighted sum of the roughness and the slope. The weightings are chosen such that areas with a high roughness (and therefore obstacle rich) are avoided first. The remaining areas are then avoided if the landing incidence angle is too high. These weights are chosen based on the requirements of the platform.

By applying these algorithms successively, the rotorcraft can generate a map and locate a place to land. However, it is still unclear how this process will be effected by environmental conditions. A robust and reliable solution must consider the effect of obscurants. The performance of these algorithms is directly dependent on the ability of the sensor to measure the environment. To gain insight into how the algorithms

will be effected, the sensor was tested and evaluated to characterize its performance in smoke.

### 3 Obscurant Testing and Evaluation

The path for evaluating UAV algorithms developed in the lab is to perform flight tests. While flight tests are necessary to ensure the validity of the algorithm, unpredictable conditions can often lead to inconclusive results. We choose to gain a full understanding of the performance of our sensors and algorithms by introducing acclimation conditions in a controlled environment. This process of testing and evaluation gives a clear understanding of how the sensor and algorithms operate. The flight test is then a verification of the results measured in the lab.

#### 3.1 SISTR

Assessing the performance of the sensor requires a testing facility capable of repeatably and controllably simulating realistic environments. SISTR, shown in Fig. 4, is a National Science Foundation funded UAV testing facility that provides this capability. SISTR measures 19 ft. x 18 ft. x 20 ft. enclosing a mock urban environment constructed at full scale with actual materials such as plywood, brick and cinder blocks. The environment can be augmented and reconfigured with other features such as poles, wires and trees to test robustness to varying obstacles.

As described in [11], the facility is surrounded by a six degree-of-freedom computer controlled gantry. Using the math model that describes the flight dynamics of an aircraft, the gantry can be programmed to mimic the flight of a vehicle. Table 1 displays the maximum and minimum velocities achievable by the gantry. While these velocities do not represent the full range of velocities achievable by UAVs, they

**Fig. 4** Systems Integrated Sensor Test Rig (SISTR). SISTR provides a stage for testing and evaluation of hardware in simulated urban environments and disaster scenarios. Effects such as varied lighting, rain, fog and smoke can be introduced in a controlled and repeatable fashion



**Table 1** SISTR velocities

Axis	Velocity range
X	0.012–0.61m/s
Y	0.019–0.61m/s
Z	0.021–0.61m/s

do encompass a portion of the operating range for rotorcraft. The position in all translational axes of the gantry can be controlled to within  $\pm 0.5\text{cm}$ .

Sensor packages can be mounted on SISTR and virtually flown through an environment. Sensor data is collected in real time by the same control algorithms and software that would be used in flight. The control commands are fed into a mathematical model of the aircraft, which generates aircraft positions. These positions are then played back on SISTR.

SISTR is also equipped with testing apparatus to simulate different environmental conditions. There are some permanent fixtures. Stage lights placed near the top of the facility can be individually controlled to create varied lighting scenarios. Light-blocking curtains can be used to create night time conditions. Other environmental fixtures can be added as needed. A fog generator has been used to simulate obscurants. In the past a rain and dust machine [12] were created to simulate more extreme operating conditions.

### 3.2 SICK LMS200 Laser Range Finder

The sensor we tested was the SICK LMS200. The LMS200 is a 2D scanning laser range finder. A beam of laser light is projected onto a rotating mirror. This mirror deflects the beam, creating a fan of laser light. Any object that breaks this fan reflects laser light back to the sensor. The distance is calculated based on how long the laser takes to bounce back to the sensor. The sensor is capable of performing scans at a rate of up to  $75\text{Hz}$ .

The LMS200 utilizes a class I eye-safe laser. The wave length of the laser is  $905\text{nm}$ . According to the manufacturer, the LMS200 has a range of  $80\text{m}$  with an accuracy of  $\pm 4\text{cm}$  and a  $180^\circ$  field of view with  $.5^\circ$  resolution. This range is software selectable. The maximum detection distance can be shortened to increase the accuracy of measurements.

This sensor is very common among robotic ground vehicles, primarily because of its wide viewing angle and relatively long range. As the research done in [4] and [5] suggests, the detection range is well suited for rotorcraft operating in near-Earth environments. One major drawback to implementing this sensor on a rotorcraft is its size and weight. The LMS200 is  $156 \times 155 \times 210\text{mm}$  and  $4.5\text{kg}$ , the majority of the weight coming from the ruggedized steel encasing.

From research in ground vehicles documented in [13] and [14], the SICK laser sensors are susceptible to airborne particulate matter such as dust. Using SISTR, we sought to quantify these effects.

### 3.3 Obscurant Characterization

One unique quality of SISTR is its ability to simulate weather conditions and other disturbances. With test rigs constructed inside the facility, sunlight, rain, fog and

other effects can be generated in a controlled, repeatable manner. Testing conditions and standards were determined using the US military guidelines for all weather performance outlined in [15]. All military vehicles are held to these standards, including UAVs.

Military standards acknowledge that smoke, fog and similar environmental factors can affect electro-optical systems. However, the standards fall short of defining requirements for these factors as these conditions are difficult to quantify. In this paper the amount of smoke is qualitatively asserted based on the “visibility” through the smoke. This is given as the distance that objects can be seen through the smoke.

The smoke was simulated using Superior Signal Company #3C smoke candles. These smoke candles issue 40,000 cubic feet of smoke over 3 minutes. The substance is actually a zinc-chloride mist, which is not the same content as naturally occurring smoke. However, as suggested in [1], particle size plays a major role in dispersing light. It is therefore assumed that the principle effect was still effectively modeled by this substance. This infrastructure provides a solid basis for determining the sensor model in the presence of obscurants.

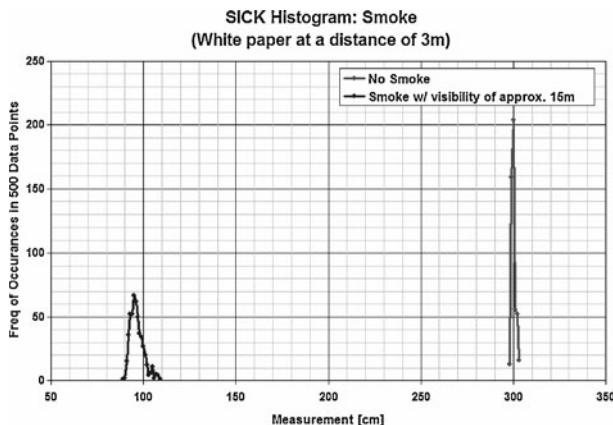
To characterize the sensor, a sheet of white paper was placed 3m from the sensor aligned at 90° (in the middle of the sensing range). 500 data points were recorded and the resulting histogram plotted. Characterization was performed both with and without smoke.

For the smoke test, part of one of the smoke candles was used. Since the testing volume measured only 6,840 cubic feet, roughly 20% of the smoke candle material was extracted and ignited. After some dissipation, this produced smoke with a visibility of approximately 15m. The testing environment with and without smoke is shown in Fig. 5.

**Fig. 5** To characterize the SICK LMS200, the sensor was placed inside the testing environment and pointed toward a white sheet of paper placed 3m away. 500 data points were recorded without smoke (*top*) and in smoke with a visibility of approx. 15m (*bottom*)



**Fig. 6** Histogram of characterization tests performed with and without smoke. Without smoke the measurements are normally distributed around 300.05cm with a standard deviation of 1.11cm. When smoke is introduced, the distribution shifts far to the left and the deviation increases, showing that smoke blinds the sensor



The resulting histogram is shown in Fig. 6. Without smoke the sensor measurements are normally distributed about the correct distance of 3m. The mean of these measurements was 300.05cm with a standard deviation of 1.11cm. When smoke was introduced, the distribution shifted far to the left. Measurements in smoke had a mean of 96.5cm with a standard deviation of 3.63cm.

These results show that the LMS200 is blinded when the volume of its scanning area is filled with smoke. These results are consistent with qualitative assessments expressed from field tests of the sensor. The tests performed in the lab removed effects such as distribution from wind and rotor down wash, undeniably confirming that the sensor can not see through smoke. This information could now be used to interpret results of field testing the sensor.

#### 4 Platform

Verification and validation of these test results was performed with a Rotomotion SR100 electric UAV helicopter, shown in Fig. 7. The SR100 is sold as a fully robotic helicopter capable of performing autonomous take off, landing, and GPS waypoint navigation when controlled from a laptop base station. Control from the base station to the helicopter is routed through an 802.11 wireless network adapter.

The SR100 has a rotor diameter of 2m allowing it to carry a payload of up to 8kg. For these experiments, we outfitted the helicopter with custom landing gear, a custom camera pan/tilt unit, the SICK LMS200, a serial to Ethernet converter, and two 12V batteries for payload power. In total we added approximately 7kg of payload. This greatly reduces the flight time, which is up to 45 min without a payload.

The biggest attraction of this platform, however, is the fact that it is already outfitted with all of the necessary sensors to calculate its pose. Gyros, an inertial measurement unit, and a magnetometer provide the craft's attitude and heading. This information is fused with a Novatel GPS system to provide position data. The position is reported as Cartesian coordinates relative to a global frame, who's origin is at the location where the helicopter was activated.

**Fig. 7** The SR100 robotic helicopter from Rotomotion, Inc. The SR100 is sold as a fully robotic package capable of automated take off, landing, and GPS waypoint following



With the pose information already calculated, this platform enables mapping and landing zone identification using the algorithms described earlier.

## 5 Experimental Results

To determine the feasibility of performing mapping and safe landing zone identification in the presence of obscurants, verification and validation of the SISTR tests was performed. These tests were conducted at the research facility of Piasecki Aircraft. A



**Fig. 8** The LMS200 was attached to the bottom of the SR100 helicopter and flown over smoke. Results showed that the LMS200 reflected off of the smoke, but was still able to locate flat terrain to land

location was found which contained a desirable landing area surrounded by cluttered terrain. The testing area used was a paved area surrounded by bushes and under-brush.

The helicopter was flown from a remote location, over the cluttered terrain, and into the desirable landing area. Simple software was written which evaluated the area directly beneath the helicopter to determine if it was flat. When the area was flat enough for the helicopter to land, the software displayed the scan as being green.

An initial test was conducted in the absence of smoke. During this test, the helicopter successfully identified the desirable landing area. Next, a smoke candle was placed in the desirable landing area and ignited. The test was repeated, this time with smoke obscuring the landing area. The results are depicted in Fig. 8.

As can be seen in the figure, the software successfully identified a flat region amongst the obscured area. However, the right side of the scan shows a detected obstacle depicted as a non flat region. Tests conducted in the lab confirmed that the sensor reflects off of smoke. It is therefore concluded that down wash exposed part of the obscured area, while the laser reflected off of the denser smoke.

## 6 Conclusions and Future Work

The results from the flight tests showed that the sensor detected a flat region amongst a smoke obscured area. Since previous tests inside a controlled environment proved that the sensor cannot see through smoke, it can be concluded that part of the obscured area was dispersed by down wash. This conclusion is further supported by video from the on board camera which suggests that there was smoke beneath the helicopter when a flat region was detected.

Now that the effect of smoke has been characterized and observed in the real world, we would like to incorporate these considerations into the algorithms outlined earlier. The mapping algorithm contains probabilistic models for both the pose sensors and the laser. The model of the sensor obtained from the characterization tests could be incorporated into this algorithm. If the helicopter is known to be flying over an obscured area, this model could be changed to match the conditions that the helicopter is operating in.

Furthermore, filters could be applied to the sensor data to remove noise from smoke or dust. This implementation could be augmented by providing the helicopter with existing terrain maps. In this scenario, laser scans gathered by the helicopter could be compared to the previously acquired terrain maps. The difference in information could be used to identify discrepancies in the two data sets. New obstacles that were introduced to the terrain such as cars and trucks would consistently appear in the helicopter scans. Noise from smoke or dust would appear inconsistent, and could then be filtered out.

However, the most plausible solution to this problem is to pair the LMS200 with a sensor that is capable of penetrating smoke. Previous work has shown us that sonar performs much better in the presence of airborne obscurants. If sonar is paired with the laser range finder, when the helicopter enters obscured conditions, more emphasis could be placed on the sonar data. We are currently in the process of investigating these avenues to create a sensor suite capable of navigating a helicopter in obscured conditions.

This paper showed how a cohesive design process of testing and evaluating followed by verification and validation can remove ambiguities in field testing robotic sensor suites. The lessons learned from these tests can be extended to many different sensor types and environmental effects. This framework provides a solid basis for developing UAV sensor suites and sensing algorithms, thereby decreasing developing time and reducing risks in field tests.

## References

1. Dimmeler, A., Clement, D., Bichtemann, W.: Effects of obscurants on the performance of laser range finders. Technical Report 6, FGAN-FfO, Tübingen, Germany (1998)
2. Brinkworth, B.: Calculation of attenuation and back-scattering in cloud and fog. *Atmos. Environ.* **5**(8), 605–611 (1971)
3. Arshinov, Y.F., Donchenko, V.A., Zuev, V.E., Kostin1, V.V., Samokhvalov, I.V.: Experimental investigation of attenuation and backscatter of laser radiation at  $\lambda=2.36\text{ }\mu\text{m}$  and  $\lambda=0.63\text{ }\mu\text{m}$  by artificial fogs and smokes. *Russ. Phys. J.* **16**(6), 789–793 (1973)
4. Thrun, S., Diel, M., Hahnel, D.: Scan alignment and 3-d surface modeling with a helicopter platform. In: The 4th Int. Conf. on Field and Service Robotics, pp. 14–16 (2003)
5. Scherer, S., Singh, S., Chamberlain, L., Saripalli, S.: Flying fast and low among obstacles. In: International Conference on Robotics and Automation (ICRA), pp. 2023–2029 (2007)
6. Saripalli, S., Montgomery, J., Sukhatme, G.: Vision-based autonomous landing of an unmanned aerial vehicle. In: Int. Conf. on Robotics and Automation (ICRA), pp. 2799–2804 (2002)
7. Saripalli, S., Sukhatme, G.: Landing on a moving target using an autonomous helicopter. In: Int. Conf. on Field and Service Robotics (2003)
8. Johnson, A., Montgomery, J., Matthies, L.: Vision guided landing of an autonomous helicopter in hazardous terrain. In: Int. Conf. on Robotics and Automation, pp. 3966–3971 (2005)
9. Meingast, M., Geyer, C., Sastry, S.: Vision based terrain recovery for landing unmanned aerial vehicles. In: IEEE Conf. on Decision and Control (CDC), pp. 1670–1675 (2004)
10. Johnson, A., Klumpp, A., Collier, J., Wolf, A.: LIDAR-based hazard avoidance for safe landing on Mars. *AIAA J. Guid. Control Dyn.* **25**(6), 1091–1099 (2002)
11. Narli, V., Oh, P.: A hardware-in-the-loop test rig for designing near-earth aerial robotics. In: International Conference on Robotics and Automation (ICRA), pp. 2509–2514 (2006)
12. Sevcik, K.W., Oh, P.Y.: Designing aerial robot sensor suites to account for obscurants. In: International Conference on Intelligent Robots and Systems (IROS), pp. 1582–1587 (2007)
13. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Stanley, the robot that won the darpa grand challenge. *Journal of Field Robotics* **23**(9), 661–692 (2006)
14. Urmson, C., Ragusa, C., Ray, D., Anhalt, J., Bartz, D., Galatali, T., Gutierrez, A., Johnston, J., Harbaugh, S., Kato, H., Messner, W., Miller, N., Peterson, K., Smith, B., Snider, J., Spiker, S., Ziglar, J., Whittaker, W., Clark, M., Koon, P., Mosher, A., Struble, J.: A robust approach to high-speed navigation for unrehearsed desert terrain. *Journal of Field Robotics* **23**(8), 467–508 (2006)
15. Research, development, test and evaluation of materiel for extreme climatic conditions. Army Regulation, pp. 70–38 (1979)

# Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller

Karl E. Wenzel · Paul Rosset · Andreas Zell

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 21 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** The growth of civil and military use has recently promoted the development of unmanned miniature aerial vehicles dedicated to surveillance tasks. These flying vehicles are often capable of carrying only a few dozen grammes of payload. To achieve autonomy for this kind of aircraft novel sensors are required, which need to cope with strictly limited onboard processing power. One of the key aspects in autonomous behaviour is target tracking. Our visual tracking approach differs from other methods by not using expensive cameras but a Wii remote camera, i.e. commodity consumer hardware. The system works without stationary sensors and all processing is done with an onboard microcontroller. The only assumptions are a good roll and pitch attitude estimation, provided by an inertial measurement unit and a stationary pattern of four infrared spots on the target or the landing spot. This paper details experiments for hovering above a landing place, but tracking a slowly moving target is also possible.

**Keywords** Visual tracking · Hovering flight control · Unmanned aerial vehicle (UAV) · Low-cost

## 1 Introduction

There has been great interest in unmanned aerial vehicles (UAVs) over the past decade. Valavanis constitutes in [18] that UAVs should become smaller and smarter. As tracking is one of the key features for autonomous flying robots, lightweight

---

K. E. Wenzel (✉) · P. Rosset · A. Zell  
Department of Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany  
e-mail: karl.e.wenzel@uni-tuebingen.de

P. Rosset  
e-mail: paul.rosset@student.uni-tuebingen.de

A. Zell  
e-mail: andreas.zell@uni-tuebingen.de



(a) The Wii remote.

(b) The internal camera sensor.

**Fig. 1** The sensor of the Wii remote

and low-cost tracking solutions are required. In this paper, we show how a cheap infrared (IR) consumer electronic camera can be used as main sensor for stable flight control. The control algorithm is running on an onboard microcontroller. This technique paves the way for low-cost target tracking, automatic starting and landing and position estimation of other robots for miniature flying robots (MFRs). Actually, this paper focuses on hovering a miniature quadrotor in a defined position over a landing place.

The camera, the primary sensor in our configuration, is part of the Wii remote controller (informally known as the Wiimote), distributed by Nintendo (Fig. 1). It is capable of tracking infrared blobs and the integrated circuit provides the pixel position of each tracked blob at a high frequency. The key idea of our approach is to track a pattern of infrared spots located at the landing place, or target, by looking downwards with a fixed camera in quasi stationary flight. The design of the pattern is a T-shape with known dimensions. So the system is able to calculate the distance, or  $z$  position and yaw angle ( $\psi$ ) directly from geometric information. Assuming that the inertial measurement unit (IMU) of the aircraft provides accurate roll and pitch angles, the  $x$  and  $y$  positions can also be estimated. These four degrees of freedom are the inputs for the control loop to hover above the target.

The aircraft we used is a X3D-BL Hummingbird quadrotor helicopter distributed by Ascending Technologies.<sup>1</sup> In our experiments, the aircraft hovers indoor at a target height between 50 cm and 1 m above the landing pad. High level control behaviours could take control of the aircraft from this position. The system is capable to hover at a height of 50 cm with a root mean square deviation of 1.7 cm.

As Bouabdallah et al. demonstrated in [2], a classical PID controller has the ability to control a quadrotor in the presence of minor perturbations. That is why the control loop could be designed as a simple algorithm running on an onboard microcontroller at a high frequency.

## 2 Related Work

Onboard vision-based tracking and navigation have mostly been done on vehicles of a significant size and weight using comparatively expensive, industrial cameras and

<sup>1</sup><http://www.asctec.de/>

high performance processors. The work by Shakernia et al. [15, 16] focuses on the control of a large, single rotor helicopter (overall length 3.6 m), where the position and velocity to a planar landing pad is estimated by a vision system. Nordberg et al. [11] present vision methods for motion estimation, navigation and tracking. Their helicopter is approximately  $2 \times 1$  meters and has a maximum payload of 30 kg. Saripalli et al. [13] introduce a vision-based autonomous landing algorithm, Hrabar shows path planning and stereo-based obstacle avoidance for a comparable aircraft in [8]. Frew et al. [3] present a vision-based road-following algorithm for a small, autonomous fixed-wing aircraft. This kind of aircrafts are capable of carrying personal computers with modern Gigahertz processors and accurate stereo cameras. Smaller aircrafts have to deal with very limited payload capacity. Hence, other solutions have to be found.

Roberts et al. [12] describe a low-cost flight control system for a small outdoor helicopter, where all processing is performed onboard. For a comparable solution on miniature UAVs, as the Hummingbird quadrocopter, the system has to be even smaller and lighter. Most MFR tracking systems are based on stationary cameras or processing is done on a ground station.

Kemp [9] developed a visual control method by using a sub-miniature onboard camera of only 9 g weight. Guenard et al. [4] present a visual servo control and Herisse et al. [7] describe a hovering flight and vertical landing control using optical flow. Calculations are transferred to the ground station in these projects, which allows to control a miniature quadrocopter, but leads to restrictions in autonomy.

Gurdan et al. [5] did flight experiments by using an external motion capture system. They were able to follow a trajectory with a maximum deviation of 10 cm from the desired position. Watanabe et al. [19] propose an assistant and training system for controlling unmanned helicopters in a flight field monitored by stationary cameras. Controlling UAVs without an onboard camera, but tracked by a motion capture system, leads to accurate positioning at a high control frequency, as cameras of a high quality and fast computers can be used. However, they can only operate in a few cubic meters, which is not practicable for most autonomous operations.

Mak et al. [10] describe a six degrees of freedom visual tracking system for a miniature helicopter using only three onboard LEDs and a single on-ground camera. Hay et al. [6] explain how optical tracking can be done using two Wiimotes and achieve excellent results with a minimum on hardware costs. These solutions deal with a minimum of hardware demands, but still depend on a permanent connection to the ground station when used for UAV control.

Our approach is different from those described above, as we are using the IR camera from a Wiimote onboard. The approach is only dependent on four IR diodes as external landmarks, which can easily be integrated in a landing pad and can be regarded as landing strip lights.

### 3 The Characteristics of the Wii Remote Infrared Camera

The Wiimote is a Bluetooth-compatible controller, designed for interfacing the Nintendo Wii game console. The device provides a three-axis accelerometer, 12 digital buttons and an IR camera tracker as sensors. An expansion port can be used

to connect additional input devices, like the Nunchuck controller. The price of 40 € is relatively inexpensive, considering the internal components.

The optical sensor is normally used in conjunction with a strip containing two infrared spots to determine the position and orientation of the controller to control a cursor on the screen. Recent publications benefit from the specialities of the Wiimote as it can be connected to an PC by using Bluetooth. Sreedharan et al. [17] analyses the motion by interpreting the acceleration sensor information. Shou et al. [14] integrates this controller into a game environment in a multi-wall virtual reality theatre.

As size and weight is important for miniature flying robots and the Bluetooth connection is not needed when operating onboard, we detached the camera from the controller. The sensor weight of 0.4 g at a dimension of  $8 \times 8 \times 5$  mm<sup>3</sup> makes it an ideal MFR onboard sensor. The multi-object tracking engine (MOT sensor), manufactured by PixArt Imaging is capable of blob tracking of up to four IR sources. By eight times subpixel analysis, the native resolution of 128×96 pixel is scaled up to 1024×768 pixels. The sensor supports different modes, which differ in sensitivity and information about the infrared blobs. The complete information includes the blob size, intensity and the bounding box. For our use, the basic mode, providing only the position, is sufficient. The horizontal field of view is approximately 45°, and the refresh rate of 100 Hz in Bluetooth mode is adequate for fast optical tracking. When operating in I<sup>2</sup>C bus mode, frequencies of 250 Hz and higher can be achieved.

Just a few electronic components are required to integrate the sensor in a microcontroller circuit. The camera runs at 3.3 V and needs an external synchronisation of 24 MHz. The I<sup>2</sup>C bus address and the communication protocol is known, so the data can be obtained easily by an I<sup>2</sup>C host via polling.

#### 4 The UAV System

The AscTec X3D-BL Hummingbird quadrocopter is 53 cm in diameter, weighing only approximately 0.5 kg. A 2.1 Ah lithium-polymer accumulator powers the electronic motors and the onboard electronics. Its flight time of up to 23 min depends on the additional payload and the flight maneuvers.

The Hummingbird platform comes with a three-axis gyroscope, an accelerometer, a compass module, a GPS sensor and pressure sensor. The sensors, the microcontroller and the flight control algorithm running at 1 kHz are fast enough to stabilise the quadrocopter sufficiently, making flying relatively easy compared to common model helicopters. The pressure sensor allows keeping a specific height, and compass and GPS are used to keep a given position in the air or to fly to waypoints, transmitted by the ground station connected via a ZigBee transmission module. So the Hummingbird provides basic outdoor autonomy without any additional hardware, but also permits high level applications to control the flight and to connect additive devices via two serial ports.

By varying the speed of the four motors, the aircraft can tilt, turn and change its height. A detailed analysis of the advantages of quadrotors and a description of their dynamic model can be found in [1].

As we did experiments with other aircrafts, we decided to program a separate board, populated with an Atmel ATmega 644P microcontroller. All processing is done on this board and the microcontroller acts as a gateway to send sensor

information to the base station. The 8-bit microcontroller is clocked at 14 MHz, which enables serial communication with the quadrocopter and the base station. While the first serial port is communicating with the robot, the second port is connected to the ZigBee module which sends messages to a ground station. The base station is used for monitoring the current pose estimation and control status and can be used for changing control loop parameters.

The framework for the microcontroller is able to communicate with different control protocols, so our tracking can be applied to various quadrocopters. The Hummingbird has proven to provide roll and pitch estimations of sufficient quality, leading to accurate position estimations.

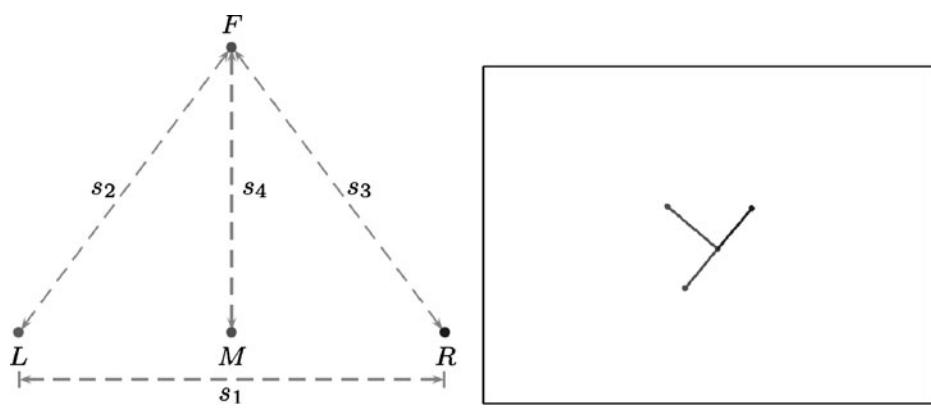
The camera is fixed in the center of the quadrocopter frame, so the position of the aircraft can easily be calculated without additional translations to compensate the position offset.

## 5 Retrieving the Pose From Camera and IMU

This section describes how the current position vector  $\mathbf{p} = (x, y, z, \psi)^T$  is estimated.  $x, y$  and  $z$  are the Cartesian coordinates of the landing place origin, relative to the camera and  $\psi$  is the orientation in yaw. The position vector  $\mathbf{p}$  can be estimated by using the sensor information, provided by the Wiimote camera combined with the roll and pitch angles provided by the IMU.

### 5.1 Pattern Analysis

By running the Wiimote camera in basic information mode, the position of four points are transmitted to the microcontroller. The data is forwarded to the base station, where an image of the current view can be visualised. Figure 2b shows a typical image, received at the base station while the quadrocopter was in 50 cm



(a) The pattern configuration in half original size.

(b) A typical image receive from the camera at a distance of 50 cm.

**Fig. 2** The pattern, attached to the landing place. **a** The pattern configuration in half original size. **b** A typical image received from the camera at a distance of 50 cm

height. The four dots are linked with colored lines, to get a faster visual feedback for the operator.

The pattern we constructed measures 90 mm from the left to the right IR spot ( $s_1$ ), and 60 mm from the middle to the front IR spot ( $s_4$ ), which fits easily in a standard circuit board. Each spot is represented by a single 940 nm wavelength infrared LED. This configuration has proven to be a good size for indoor landing pad tracking, where the pattern must be recognised at a relatively close distance. Larger patterns would allow for more precise tracking at a larger distance, but would no longer fit in the field of view when getting closer. Our pattern can be completely covered by the camera from a distance above 150 mm. However, when the aircraft is moving, single spots of the pattern usually get lost due to the inclination, even at larger distances.

To get unambiguous position information, the disordered points **F** (front), **L** (left), **M** (middle) and **R** (right) first have to be identified (Fig. 2a). Several techniques could be used to sort the points. Our approach focuses on fast processing on a microcontroller, which means that we need to limit floating point operations as well as trigonometric functions.

Given a set of four points, we calculate the distances of all points to lines, defined by two of the remaining three points. As **M,L,R** are lying in one line, a combination of **M,L,R** leads to the minimum point to line distance. The point not used in this combination can be indirectly identified as **F**.

The maximum distance between two of the three points left identifies **M** to lie between **L** and **R**. Identifying these points is sufficient for our tracking control.

For a complete identification, **L** and **R** should also be identified. Let **A** and **B** be the remaining last points to recognise. By using  $\mathbf{A} = (a_x, a_y)^T$ ,  $\mathbf{M} = (m_x, m_y)^T$  and  $\mathbf{F} = (f_x, f_y)^T$ , the following equation identifies **A** unambiguously as **L** ( $s$  is positive) or **R** ( $s$  is negative):

$$s = \text{sgn} [(a_x - m_x)(f_y - m_y) - (a_y - m_y)(f_x - m_x)] \quad (1)$$

If all points have been identified, the position vector **p** of the quadrocopter relative to the landing pad can be determined.

## 5.2 Estimating Yaw and Distance

The yaw angle  $\psi$  and the  $z$  distance can be calculated directly from geometric information, while the  $x$  and  $y$  position estimation depends on accurate roll and pitch angle information, provided by the IMU. The yaw angle  $\psi$  is measured relative to the straight line, defined by **M** and **F** and can be easily calculated:

$$\psi = \text{atan2}(m_x - f_x, m_y - f_y) \quad (2)$$

The viewing angle per pixel of the camera in radians is given as  $\rho = 7.61 \cdot 10^{-4}$ . Let  $s_i$  be the physical space between two LEDs and  $d_i$  be the pixel distance of the corresponding points. The physical distance  $z_i$  from the camera to the pattern can be calculated as:

$$z_i = \frac{s_i}{\tan(d_i \rho)} \quad (3)$$

The following points were used for distance measuring in our experiments:

$$d_1 = |\mathbf{L} - \mathbf{R}|, d_2 = |\mathbf{L} - \mathbf{F}|, d_3 = |\mathbf{R} - \mathbf{F}|, d_4 = |\mathbf{M} - \mathbf{F}|.$$

We got satisfactory results in distance measuring with a stationary camera by only using  $d_1$ . However, when flying, disturbances cause noise in subpixel analysis and notable errors in distance calculation. By using the mean value, the error caused by pixel noise can be decreased:

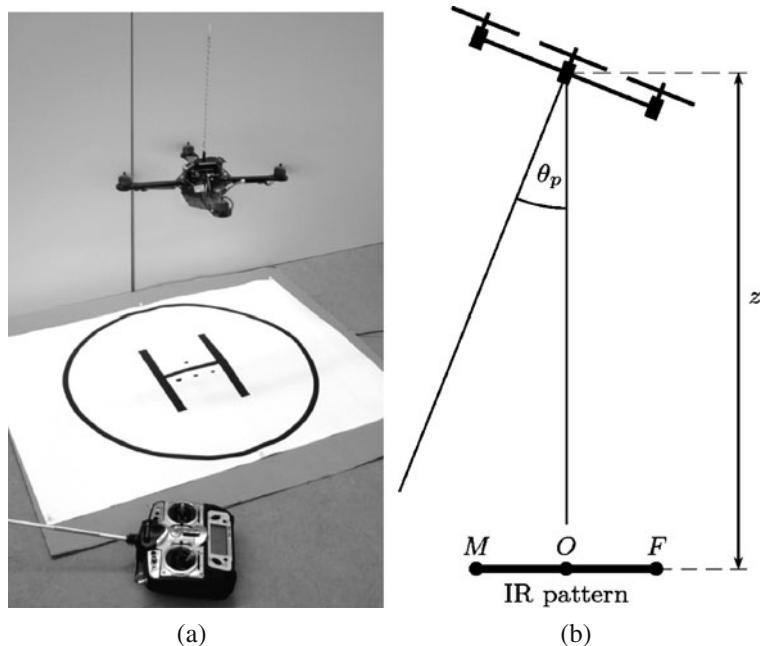
$$z = \frac{1}{4}(z_1 + z_2 + z_3 + z_4) \quad (4)$$

### 5.3 Estimating the $x$ and $y$ Position

This section describes how the  $x$  and  $y$  position of the aircraft can be estimated, and why an IMU is required with the presented pattern.

Since the aircraft's desired position is directly above the pattern, the view angle to the pattern is limited by the small roll and pitch angles required for position corrections. These angles are usually between  $\pm 3^\circ$  during controlled flight. In such an angular attitude, as shown in Fig. 3b, the pattern appears in a displaced position in the camera image. Let this angle be denoted by *geometric angle*.

If the camera orientation is known, the  $x$  and  $y$  position relative to the center of the pattern can be estimated. Any camera displacement also leads to distortion of the pattern, which permits to estimate the attitude relative to the pattern. Experiments



**Fig. 3** The robot autonomously hovering above the pattern. **a** The robot hovering. **b** Schematic lateral view of the configuration

have proven that an estimation with a resolution of only  $5^\circ$  at a distance of  $80\text{ cm}$  is possible. However, the notable distortion of the used pattern only counts a few pixel (depends on the distance) and is ambiguous for the pitch angle. A positive pitch angle leads to the same shortened  $d_4$  as a negative angle would do. These limitations make the use of the IMU necessary. A modified pattern, optimized for distortion analysis could allow for controlling without an IMU.

A displacement of the aircraft means that the physical angles provided by the IMU differ from the geometric angles. By combining the geometric and the physical roll and pitch angles, the  $x$  and  $y$  position relative to the pattern can be calculated in respect to the current  $z$  position.

Let  $\mathbf{O}$  be the origin position of the landing pad, defined by  $\mathbf{O} = \frac{1}{2}(\mathbf{M} + \mathbf{F})$ , and  $\mathbf{C}$  be the middle of the image, defined by the camera resolution of  $1024 \times 768$  pixels:  $\mathbf{C} = (c_x, c_y)^T = (512, 384)^T$ . The physical roll  $\varphi_p$  and pitch  $\theta_p$  angles are provided by the onboard IMU and requested before each position estimation. The geometric roll  $\varphi_g$  and pitch  $\theta_g$  angles, representing the angles of vision, are calculated by:

$$\theta_g = (o_x - c_x)\rho \quad (5)$$

$$\varphi_g = (o_y - c_y)\rho \quad (6)$$

Potential inaccuracies in camera positioning are compensated with the calibration angles  $\varphi_c$  and  $\theta_c$ . The calibration angles only have to be measured once after attaching the camera to the aircraft. By correcting the physical angles of the aircraft with the calibration angles and combining them with the visual angles, the real displacement of the landing pad can be estimated:

$$\varphi = \varphi_g + \varphi_p + \varphi_c \quad (7)$$

$$\theta = \theta_g + \theta_p + \theta_c \quad (8)$$

$$x = \tan(\varphi)z \quad (9)$$

$$y = \tan(\theta)z \quad (10)$$

## 6 Flight Control Algorithm

After successfully estimating the current position  $\mathbf{p} = (x, y, z, \psi)^T$ , the aircraft can be controlled to hover at a defined height above  $\mathbf{O}$ . The algorithm is inspired by Gurdan et al. [5], where four independent controllers were operating.

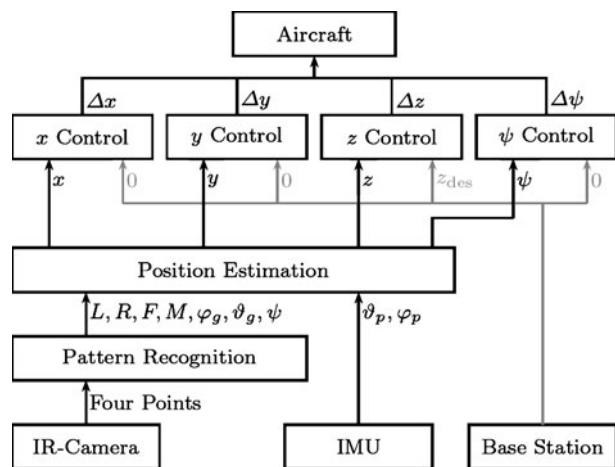
The only variable in our controller is the desired height ( $z_{\text{des}}$ ) above the target  $z$ . All other elements of  $\mathbf{p}$  should remain zero.

### 6.1 Tracking and Flight Control Method

An overview of our system including pattern recognition, position estimation and flight controller is shown in Fig. 4. The aircraft provides an interface to flight control, which uses the output of our four independent PID controllers. The four motors are controlled on a lower level by the internal controllers of the aircraft.

The control loop is currently performed with a frequency of 20 Hz. The controller requires an actual pose estimate by the IMU, whose request lasts approximately

**Fig. 4** Global picture of the control system. The desired height ( $z_{des}$ ) can be varied by the base station

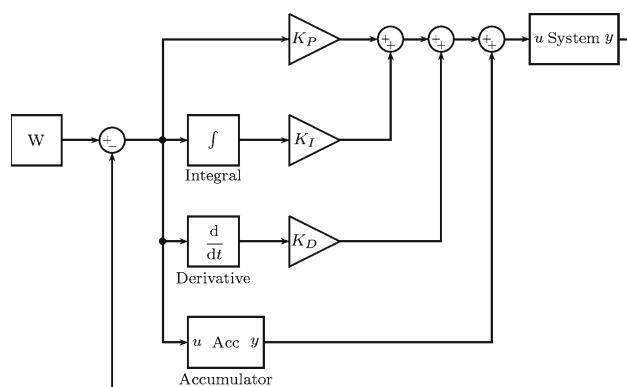


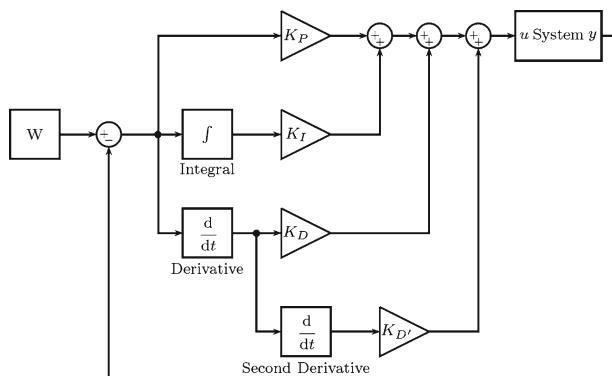
30 ms in total. Additional 10 ms are required to send sensor information to the base station, where the current status is monitored and displayed. The remaining 10 ms are available for retrieving sensor information, running the control algorithm and receiving data from the base station. A considerably higher control frequency would be possible with a faster IMU request.

## 6.2 Height Controller

The thrust control value, needed to hover at a desired height, changes, depending on the actual payload and battery charge. This is why the thrust controller is divided into two parts, an accumulator for the base value and a fast controller for position control. We obtained good results by counting up whenever the robot is below the desired height for some cycles, and down otherwise. The fast response controller is implemented as a PID loop with  $K_D$  being the biggest component. Figure 5 shows the structure of the height controller.

**Fig. 5** The height controller



**Fig. 6** The  $x$  and  $y$  controller

### 6.3 Roll/Pitch and Yaw Controller

The  $x$  and  $y$  controllers are identical and were harder to derive, since the behaviour response is not proportional to horizontal speed but to rotational velocities. A predictive control is required to achieve a stable position hold. By designing a cascaded control loop where not only the speed but also the acceleration is highly weighted, the behaviour obtains the desired prognostic ability. Figure 6 shows the structure of the  $x$  and  $y$  controller.

The yaw controller is implemented as a PID loop with a large  $K_P$  and a small  $K_I$ .

## 7 Experimental Results

Experiments with different quadrocopters have shown that an accurate attitude estimate is essential for reliable position estimation with our approach. The following example data represents typical experimental results and should demonstrate the performance of the system.

### 7.1 Attitude Estimation

To validate the quality of the attitude estimation provided by the IMU, a slightly different configuration of the presented system was used. A Wiimote camera, heading to the front, allows for easily designating the roll angle of the aircraft from image data. By using a bigger pattern, the angle estimation is very accurate and can be used as reference. Comparing the angles measured with camera and IMU proves good quality for the autopilot attitude estimation. We measured a standard deviation of approximately  $3^\circ$ , while the error depends on the angular velocity. These results are based on the micro-electro-mechanical systems (MEMS) gyrorotor sensor of the autopilot IMU. By contrast, IMUs using piezo gyrorotor sensors usually have to struggle with worse sensor data. The quadrotors with piezo gyrorotor sensors we tested were not capable of following the angle measured by the Wiimote camera for a longer time.

**Table 1** Controller characteristics of five minutes flight at 50 cm height

	$\Delta x/\text{mm}$	$\Delta y/\text{mm}$	$\Delta z/\text{mm}$	$\Delta \psi/^\circ$
Minimum	-60.00	-54.00	-80.00	-9.10
Maximum	54.00	64.00	87.00	7.90
Peak to peak	114.00	118.00	167.00	17.00
Mean	-0.02	-0.09	-1.16	-0.01
Standard deviation	17.54	18.55	27.78	3.03

## 7.2 Flight Control Accuracy

The Hummingbird quadrocopter comes with a GPS position controller and an air-pressure sensor for height control. These functions were been disabled during our indoor experiments. The yaw angle is controlled by the internal magnetic compass of the quadrocopter and can not be disabled. However, experiments with another quadrocopter have proven that yaw control is uncritical. The results shown here are with assistance of the internal yaw controller of the quadrocopter.

Following a large number of flights and retrieval of working parameters, the system has proven its applicability for short distance indoor landing place tracking and flight control. Tables 1 and 2 show the controller characteristics of flights in 50 cm and 1 m height above the landing place, 5 min each.

As mentioned in Section 5.3, the estimation of roll and pitch is essential for accurate  $x$  and  $y$  approximation. While the standard deviation of  $\Delta z$  in 50 cm and 1 m is rather comparable, one can notice a significant change in  $x$  and  $y$  position control. The yaw angle  $\psi$  is affected by balancing manoeuvres in flight and thus depends on the position changes in  $x$ -,  $y$ - and  $z$ -directions.

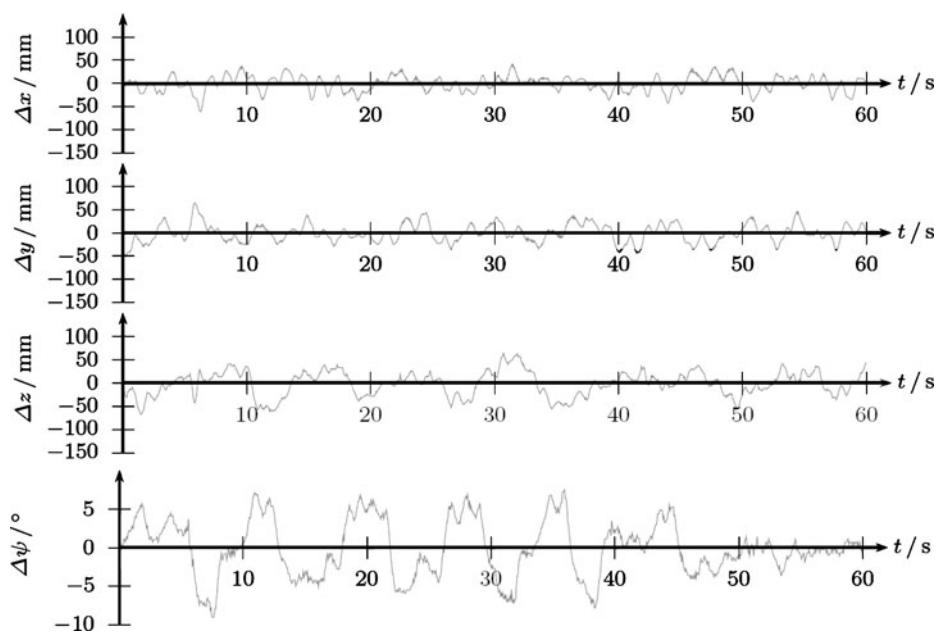
In approximately 60% of the flight time at 50 cm height, an accuracy of  $\pm 1.5$  cm in  $x$  and  $y$  position could be achieved. Details of the positioning probability are shown in Fig. 9. A higher control frequency would even improve the stabilisation, but is prevented by the slow response time of 30 ms at which the quadrocopter provides the pose estimation in the current configuration.

A detailed record of one minute of the flights is shown in Figs. 7 and 8. The plots show the position estimate sent to the base station while flying. The sensor data is smooth and nearly noise-free in a short distance. Outliers, caused by external influences as infrequent reflective lights, can easily be filtered. When operating in a height of 1 m, errors in distance approximation and especially impreciseness in roll and pitch lead to a rougher curve (Fig. 9).

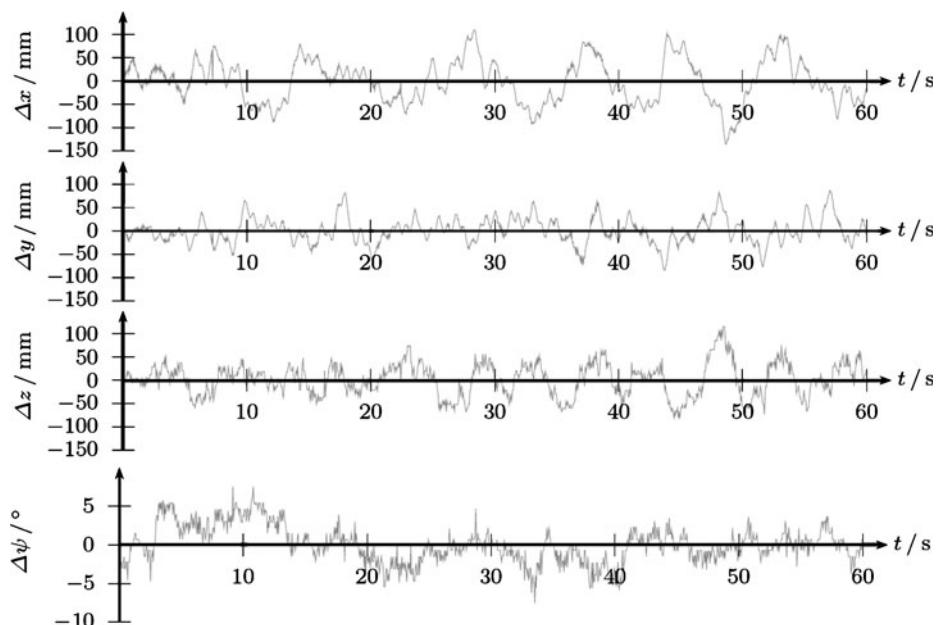
Some oscillations at different frequencies still remain in the control sequence. The ideal PID parameters depend on the working distance to the pattern. Hence, finding optimal parameters is a problem. A deviance in one axis leads to displacements in other directions. Accurate position hold can only be achieved by a combination of stable controllers for each axis. An automated parameter optimisation could help to

**Table 2** Controller characteristics of five minutes flight at 1 m height

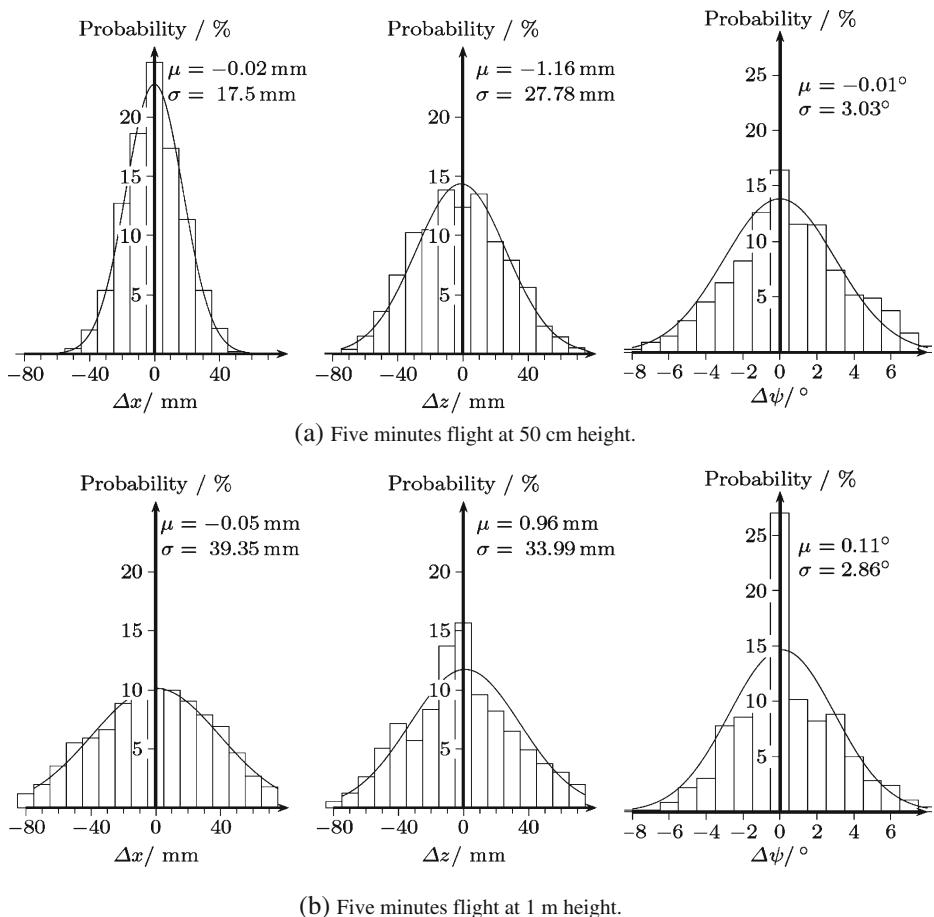
	$\Delta x/\text{mm}$	$\Delta y/\text{mm}$	$\Delta z/\text{mm}$	$\Delta \psi/^\circ$
Minimum	-156.00	-128.00	-109.00	-10.00
Maximum	130.00	117.00	155.00	13.20
Peak to peak	286.00	245.00	264.00	23.20
Mean	-0.05	-0.12	-0.96	0.10
Standard deviation	39.35	38.50	33.99	2.80



**Fig. 7** Errors of one minute flight at 50 cm height



**Fig. 8** Errors of one minute flight at 1 m height

**Fig. 9** Probability of positioning while tracking

find even better parameters. Anyway, increasing the control frequency shows greater promise for improved position hold capability.

One problem which we encountered are reflections of sunlight, leading to wrong pattern interpretation, as sunlight contains a notable fraction of infrared light.

## 8 Conclusion and Future Work

The Wiimote camera has proven to provide accurate tracking of infrared blobs and the infrared filter avoids most disturbances caused by external light sources, except sunlight. The pixel position information can easily be interpreted as an unambiguous pattern on a target. A microcontroller is capable of estimating the position to the target and controlling a miniature flying robot in hovering flight with our approach.

The field-of-view of the Wiimote camera is big enough for adjustment moves and tracking even in small distances.

The deviation from the desired position, which we obtain with our control algorithm, is small enough for quasi stationary flight for a longer time in an indoor environment. High level autonomy tasks could start from this position.

The distance to the target is limited, but by enlarging the dimension of the pattern and increasing the IR light emission by using multiple IR LEDs per point or stronger LEDs, the operating distance can be increased. However, an accurate roll and pitch estimate, provided by the IMU of the aircraft is essential. A small angular error leads to inadmissible position approximation. A second Wiimote camera, or fusion with other sensors would lead to better positioning at a larger distance.

By accelerating the IMU request, or integrating the tracking control loop in the internal program of the UAV, the control frequency could be increased considerably. A higher control frequency would advance the control accuracy and a customised control design would eliminate most of the remaining oscillations. However, the current frequency of 20 Hz is sufficient for robust hovering control.

A pattern where not all LEDs are lying in one layer, but **M** is raised in positive  $z$  direction, allows for better analysis. Assuming the aircraft operates in limited space above the landing place, such a 3D-pattern allows for estimating the  $x$  and  $y$  position in addition. This could lead to a flight control, independent from the IMU, if the achieved accuracy is sufficient.

## References

1. Bouabdallah, S., Murrieri, P., Siegwart, R.: Design and control of an indoor micro quadrotor. In: IEEE International Conference on Robotics and Automation (ICRA) (2004)
2. Bouabdallah, S., Noth, A., Siegwart, R.: PID vs LQ control techniques applied to an indoor micro quadrotor. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2451–2456 (2004)
3. Frew, E., Mcgee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., Sen-gupta, R.: Vision-based road-following using a small autonomous aircraft. In: IEEE Aerospace Conference, vol. 5, pp. 3006–3015 (2004)
4. Guenard, N., Hamel, T.: A practical visual servo control for an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1342–1348 (2007)
5. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.M., Hirzinger, G., Rus, D.: Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 361–366. Roma, Italy (2007)
6. Hay, S., Newman, J., Harle, R.: Optical tracking using commodity hardware. In: 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), pp. 159–160 (2008)
7. Herisse, B., Russotto, F.X., Hamel, T., Mahony, R.E.: Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 801–806 (2008)
8. Hrabar, S.: 3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 807–814 (2008)
9. Kemp, C.: Visual control of a miniature quad-rotor helicopter. Ph.D. thesis, Churchill College, University of Cambridge (2006)
10. Mak, L.C., Furukawa, T.: A 6 DoF visual tracking system for a miniature helicopter. In: 2nd International Conference on Sensing Technology (ICST), pp. 32–37 (2007)
11. Nordberg, K., Doherty, P., Farnebäck, G., Forssén, P.E., Granlund, G., Moe, A., Wiklund, J.: Vision for a UAV helicopter. In: International Conference on Intelligent Robots and Systems (IROS), workshop on aerial robotics. Lausanne, Switzerland (2002)

12. Roberts, J., Corke, P., Buskey, G.: Low-cost flight control system for a small autonomous helicopter. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 546–551. Taipai, Taiwan (2003)
13. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2799–2804. Washington, DC, USA (2002)
14. Schou, T., Gardner, H.J.: A Wii remote, a game engine, five sensor bars and a virtual reality theatre. In: OZCHI '07: Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design, activities, artifacts and environments, pp. 231–234. ACM, New York (2007)
15. Shakernia, O., Ma, Y., John, T., Sastry, K.S.: Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian J. Control* **1**, 128–145 (1999)
16. Sharp, C.S., Shakernia, O., Sastry, S.S.: A vision system for landing an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, pp. 1720–1727 (2001)
17. Sreedharan, S., Zurita, E.S., Plimmer, B.: 3d input for 3d worlds. In: OZCHI '07: Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design, activities, artifacts and environments, pp. 227–230. ACM, New York (2007)
18. Valavanis, K.P. (ed.): Advances in Unmanned Aerial Vehicles. State of the Art and the Road to Autonomy. Springer, New York (2007)
19. Watanabe, K., Iwatani, Y., Nonaka, K., Hashimoto, K.: A visual-servo-based assistant system for unmanned helicopter control. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 822–827 (2008)

## Landing and Perching on Vertical Surfaces with Microspines for Small Unmanned Air Vehicles

Alexis Lussier Desbiens · Mark R. Cutkosky

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 22 October 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** We present the first results of a system that allows small fixed-wing UAVs to land and cling on surfaces such as brick walls using arrays of microspines that engage asperities on the surface. The requirements of engaging and loading the spines lead to an approach in which an open-loop pitch-up motion is triggered by a range sensor as the plane nears the wall. The subsequent dynamics result in a period during which the plane stays within an envelope of acceptable orientation and velocity (pitch from 60–105 deg, vertical velocity from 0 to –2.7 m/s and up to 3 m/s of horizontal velocity) that permit successful perching. At touchdown, a non-linear suspension absorbs the remaining kinetic energy to minimize peak forces, prevents bouncing and facilitates spine engagement. The total maneuver duration is less than 1 s. We describe the spine suspension and its analysis and present results of typical perching maneuvers (10 landings under autonomous control and 20 under manual control). Under calm conditions, the success rate for autonomous perching on building walls is approximately 80%, the failures being attributed to erroneous wall detection. We conclude with a discussion of future work to increase the robustness of the approach (e.g. with wind) and allow subsequent take-offs to resume flight.

**Keywords** Perching · Landing · Vertical surfaces · Wall · Microspines · Adhesion · Suspension · Unmanned air vehicles · Endurance

---

A. Lussier Desbiens (✉) · M. R. Cutkosky  
Biomimetic and Dextrous Manipulation Laboratory,  
Center for Design Research, Stanford University, 424 Panama Mall,  
Bldg. 560, Stanford, CA, USA  
e-mail: alexisld@stanford.edu

M. R. Cutkosky  
e-mail: cutkosky@stanford.edu

## 1 Introduction

Miniature unmanned air vehicles are becoming increasingly popular for military and civilian applications. However, they suffer from a relatively short flight time, can be difficult to land safely on the ground, and are vulnerable when parked. An attractive alternative is to let them do as many small flying animals do: take frequent short flights with periods of perching in between. In particular, it is useful for small fixed-wing planes to perch on vertical surfaces such as cliffs or the walls of buildings. Clinging passively to such surfaces, they consume little power, allowing them to remain for hours or possibly days as a stable platform for unobtrusive surveillance, inspection or environmental monitoring. Vertical surfaces are especially attractive because they are often relatively uncluttered and free from debris. In addition, if the plane perches under an overhang, it can ride out a storm in relative safety.

## 2 Related Work

Although the ability to land and perch on vertical building surfaces is, to our knowledge, new, it draws upon two previous areas of work: (i) fixed-wing planes that execute dynamic maneuvers for landing and perching (ii) climbing robots that use micro-spines or directional adhesion for attachment to vertical surfaces.

### 2.1 Landing and Perching Maneuvers

In prior work, perching has been studied mostly from the aerodynamics and control point of view. For example, in one approach, researchers [6] have used motion capture cameras (119 Hz, sub-millimetre accuracy) to control an RC plane with an off-board controller for various indoor maneuvers such as flying in a room and using controlled hovering to land on a specially designed docking station. A similar system was used in [5] to create an accurate high-dimensional model of a glider during high angle-of-attack (AOA) maneuvers. This allows the plane to perform aggressive pitching maneuvers required to decelerate it to almost zero velocity before perching on a pole. Due to the challenge imposed by the very small target and the limited actuation control, the entire procedure was successful 20% of the time. In later work it was shown [14] that the glider becomes less controllable as its airspeed drops just before perching, even if controllability is improved with a fixed propeller or thrust vectoring.

In other work, autonomous hovering has been demonstrated with fixed-wing aircraft [7–9]. The controller is based on the PIC16F87 and uses a Microstrain 3DM-GX1 inertial measuring unit (30 g, 100 Hz update rate) to measure spatial orientation. The plane uses rudder and elevator to control pitch and yaw and has small propellers on the wing tips to control roll.

Still other work has focused on performing perching maneuvers using a morphing airplane [16–18]. Simulations show that pitching up the body while keeping wing and tail horizontal allows the plane to maintain control and create lift during the entire maneuver. This approach creates a shorter perching trajectory than one would require with a fixed-wing airplane although it adds some mechanical complexity.

Extensive biological research has been devoted both to flying and to ground locomotion. However, much less has focused on the physics of transitions that occur during perching. It has been suggested that flying evolved from the advantages of having only a small amount of lift to control and reduce landing forces [4]. An example of this phenomenon can be found in the flying squirrel: its low aspect ratio wing providing aerodynamic stability and lift at angles of attack up to 40 degrees. Furthermore, squirrels deliberately stall themselves prior to landing, allowing them to reduce by 60% their horizontal velocity before landing, while spreading the impact over all four limbs [3, 13].

In the present work we focus on a plane that, instead of perching on a wire, pole or docking fixture, lands on a vertical wall. As discussed in a later section, this approach provides different and possibly less restrictive constraints concerning the velocity of the plane at contact.

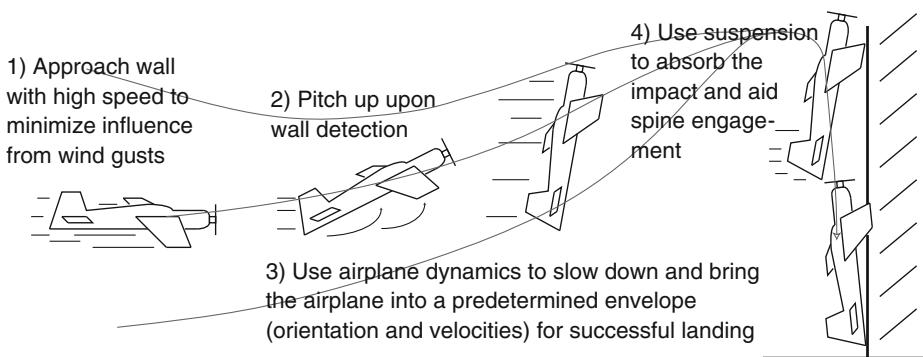
## 2.2 Vertical Climbing Robots

The mechanism by which the plane attaches itself to the wall is based on previous work on insect-inspired climbing robots that use arrays of directional spines in a compliant suspension [2, 15]. The spines are small and have tip radii ranging from 25  $\mu\text{m}$  for relatively rough materials such as stucco to 10  $\mu\text{m}$  for smoother materials such as cement or brick. Because each spine can only support a small load (1 N or less), many spines are used and it is the role of the suspension system to distribute the load among them. In comparison to other technologies such as suction, [10], magnets and pressure sensitive adhesives, spines have two main advantages: they require no power for clinging and they provide directional adhesion, which facilitates engagement and disengagement with minimal work [11]. However, to engage surfaces reliably, the spines require a particular approach trajectory. This is not difficult to achieve with a slowly climbing robot, but presents a challenge for landing and perching with an airplane.

## 3 Vertical Perching Strategy

The requirements for spine engagement translate to requirements for (i) the envelope of possible velocities and orientations of the plane as it approaches the wall and (ii) the mechanical properties of the suspension that connects the plane to the spined feet. In contrast to previous work on perching, we are not interested in contacting the surface at nearly zero velocity; the spines need to drag gently along the wall to engage asperities. A second difference with respect to previous work is that we do not assume high quality information regarding the plane velocity and orientation. We want a system with small and light sensors and a small CPU that can be placed onboard. We assume that once the wall is detected, the landing procedure will be mostly open-loop, with enough momentum to keep the plane from being highly sensitive to small disturbances.

The general sequence, illustrated in Fig. 1, is: (1) fly toward the wall at cruising speed to minimize gust disturbances, (2) pitch up when a few meters away from the wall to rapidly slow down, (3) take advantage of the airplane dynamics to position



**Fig. 1** Proposed landing sequence using the airplane dynamics to pitch up and relying on the suspension to provide the proper engaging motion on the microspines

it for landing while maintaining some forward velocity, and (4) to absorb the impact with a passive, nonlinear suspension that facilitates microspine engagement.

The focus of the work described in this paper is on the suspension in step (4) and the goal of the suspension design is to permit as generous an envelope of velocities and orientations as possible in step (3) while still ensuring spine engagement.

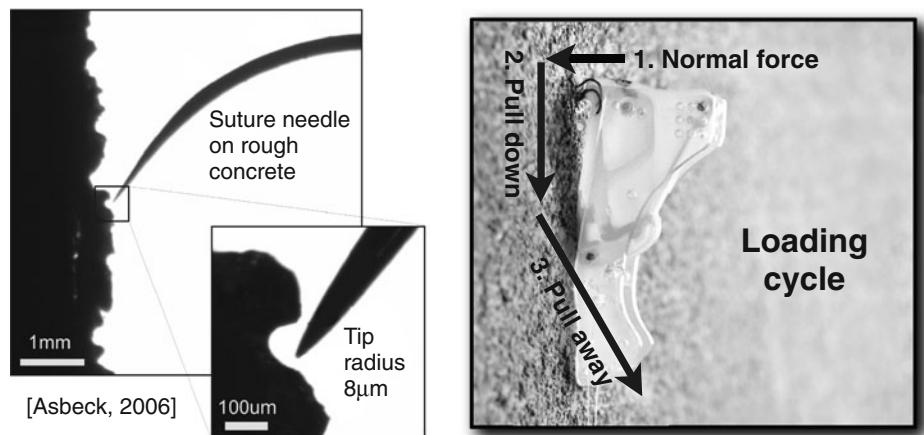
The airplane used for these experiments is a modified Flat-Out Flatana, designed for 3D maneuvers, low speed flight, and tight turns. For the experiments reported in this paper, the plane has been converted into a glider to simplify its design and reduce the number of components to be repaired after a crash; the motor has been replaced by an equivalent mass and only elevator control is used. A Paparazzi autopilot [12] has been added along with an XBee modem for telemetry, an LV-MaxSonar-EZ1 for wall detection, and a 3-axis accelerometer (ADXL300) and two 2-axis rate gyroscope (IDG300) for state sensing and estimation. The plane weighs a total of 375 g.

#### 4 Microspines and Suspension Design

Everything during the perching maneuver is done to bring the plane into a configuration that allows it to perch on the wall. It is thus important to understand the requirements of the adhesion system used, as different systems have different requirements and tradeoffs. Microspines have been chosen for this project as they can be used on a variety of surfaces [2, 15], are lightweight, require no power, and provide directional adhesion, allowing repeated use and low effort for engagement and disengagement. This section explains their requirements for adhering to a vertical surface and details the strategies used to design a suspension satisfying the requirements for a range of initial velocities and orientations.

##### 4.1 Microspine Requirements for Landing

The microspines are made of an array of small ( $\approx 15 \mu\text{m}$  tip radius) spines that hang on surface asperities as shown on Fig. 2. Each spine has its own suspension to distribute the load and conform to asperities. In this design, a single spine is

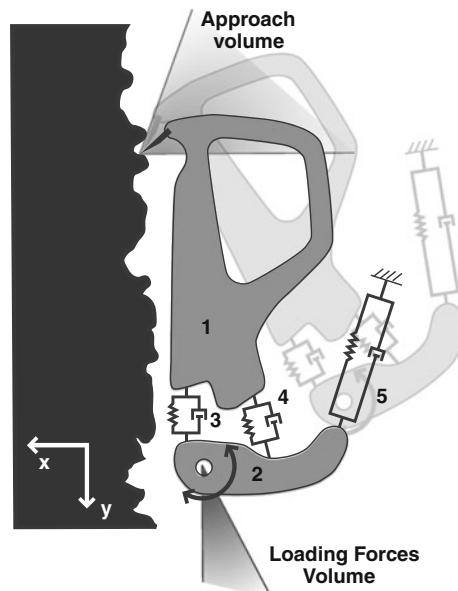


**Fig. 2** The figure on the left shows the spine tip approaching a concrete surface. The figure on the right shows the loading cycle required by the spines

enough to hold the weight of the airplane but a total of 10 spines, distributed over two feet, are used for some redundancy and to account for the higher dynamic load experienced during landing.

One challenge in using spines is that they require a specific loading cycle, shown in Fig. 2, to operate properly. One must first apply some force toward the wall, to favor engagement, while dragging the spines down. It is then possible, while maintaining a downward force, to pull away from the wall. The higher the downward force, the higher the adhesion force available. The spine suspension design follows the general procedure discussed in [2] and shown on Fig. 3. It consists of an elastic linkage that

**Fig. 3** Representation of the microspines. The spring elements 3 & 4 contribute to the tangential compliance, while element 5 provides the relatively soft compliance normal to the wall. The approach volume is mostly a function of the asperities' geometry while the loading volume depends on the coefficient of friction and the asperity geometry



is very compliant in compression (on the order of 5 N/m), in the direction normal to the wall on initial contact, to prevent bouncing. As the spine drags down the wall, it eventually (usually within a few millimeters of travel) encounters an asperity on which it can cling. At this point, a load can be applied primarily tangential to the wall, but with an outward tensile component as well. When pulled in this direction, the suspension is stiffer (on the order of 100 N/m) but compliant enough to promote load sharing between adjacent spines. For a given coefficient of friction and surface roughness, the ratio between the maximum normal and tangential force defines a fixed loading volume, shown on Fig. 3. As discussed later in Section 8, this means that for a plane to resist strong gusts of wind it will ultimately be necessary to use preloaded pairs of spines to increase the available normal force. The third criterion in designing the mechanism and setting the stiffnesses of the flexures is that the spines should not bend or rotate upwards as they are loaded, which could cause them to slip off any asperities that they find.

The ramifications of the spine design for the control of the airplane are that it should have a small velocity normal to the wall (to prevent bouncing) and a moderate downward velocity to load the spines once they make contact. In addition, the orientation of the plane should be maintained within some fixed range corresponding to desirable orientations of the spines as they contact the wall. In addition, it is desirable for the center of mass of the plane to be kept close to the wall after contact.

#### 4.2 Preventing Vertical Rebound

Although the small elastic “toe” mechanisms holding the spines help to prevent bouncing in the direction normal to the wall, they have a very limited suspension travel (a few millimeters), requiring an additional suspension in the “legs” of the landing gear interposed between the spines and the plane. More significantly, there is the danger of rebounding in the vertical direction, parallel to the wall, where the velocities and forces are higher.

Following the observation during initial tests that vertical rebound was the main cause of failure, three design goals were formulated for a good suspension:

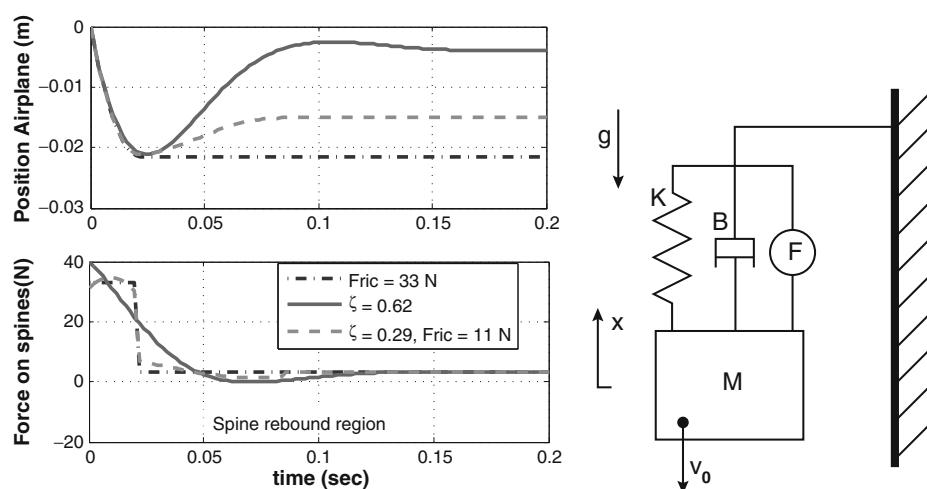
1. Minimize the maximum force ( $F_{max}$ ) in the vertical direction to allow for a lightweight structure.
2. Minimize the suspension travel ( $-x$ ).
3. Prevent spine rebound (negative vertical force) during the landing.

A solution satisfying these goals is to have a constant force for the full duration of the landing. Unfortunately, the force pattern that can be generated depends on the components available and, in the case of a small UAV, the task becomes challenging due to the size and weight constraints.

In the case of a simple spring-mass-damper system subject to gravity and having an initial downward velocity, as illustrated in Fig. 4, it is possible to write the goals previously mentioned in a single cost function minimizing:

$$J = |\max(F)| + \lambda|\max(-x)| + |(F < 0) \times \min(F)| \quad (1)$$

where  $\lambda$  is a weighting factor that can be adjusted to trade-off between the maximum force and suspension travel criteria. A Nelder-Mead simplex can then be used to minimize this cost function. The resulting trade-off curve obtained by varying  $\lambda$  is



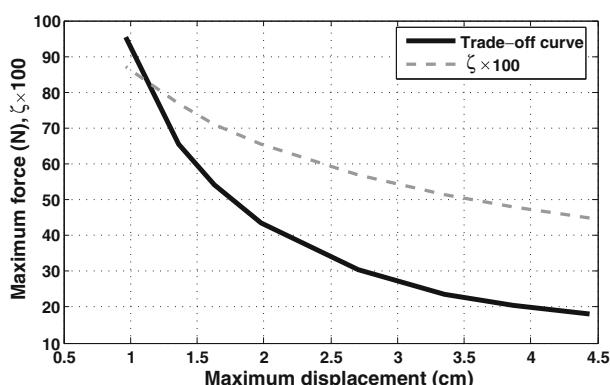
**Fig. 4** In this one-dimensional model in the vertical direction, different passive suspension parameters create a range of performances, but only the *blue curve* minimizes displacement and maximum force, while preventing spine rebound. The model includes a mass subject to gravity and having an initial velocity, being slowed down by a suspension consisting of a spring, a damper and a Coulomb friction term

shown in Fig. 5. This figure shows that as the maximum allowed displacement of the suspension is reduced, the damping ratio must be increased accordingly. This is due to the shorter landing time, corresponding to shorter suspension displacement, during which the damper can dissipate kinetic energy. This is unfavorable as a high damping ratio suspension creates a high initial force that decreases rapidly (see green curve in Fig. 4), requiring a sturdy structure to accommodate the initial force.

#### 4.2.1 Spring, Damper and Coulomb Friction

In order to prevent the high initial force caused by a strong viscous damper, a pure coulomb friction suspension could be used. However, a pure coulomb friction also has practical drawbacks: it does not return to a single equilibrium position after landing and it is difficult to adjust the level of friction precisely. In addition, it requires

**Fig. 5** Trade-off curve between maximum displacement and maximum force for a spring-damper suspension preventing rebounds. Also shown is the damping ratio required at any point in the trade-off curve. Results are shown for a 320 g airplane subject to an initial velocity of  $-2 \text{ m/s}$



a hard stop to limit the suspension travel. Fortunately, it is possible to combine coulomb friction with a spring and damper and still obtain a near optimal solution.

Knowing the desired maximum force ( $F_{max}$ ), the touchdown velocity ( $v_i$ ), the desired spring stiffness ( $k$ ) and assuming that we can design a near constant force suspension, the trajectory of the airplane during landing can be approximated by a mass subject to constant acceleration:

$$v(t) = \left( \frac{F_{max}}{m} - g \right) t + v_i \quad (2)$$

$$x_{max} = \frac{1}{2} \left( \frac{F_{max}}{m} - g \right) t^2 + v_i t \quad (3)$$

Combining, we get an expression for the maximum compression of the suspension when  $v(t) = 0$ :

$$x_{max} = -\frac{v_i^2}{2} \frac{m}{F_{max} - mg} \quad (4)$$

To obtain an approximately constant force profile on the spines during the landing, the initial force, a combination of damper and coulomb forces, must be equal to the force at maximum compression, which is a combination of the spring and coulomb forces. Thus:

$$F_{max} = -kx_{max} + F_{fric} = -b v_i + F_{fric} \quad (5)$$

From these equations, it is possible to solve for the required damping coefficient ( $b$ ), the friction force ( $F_{fric}$ ) and the damping ratio ( $\zeta$ ):

$$b = \frac{kx_{max}}{v_i} \quad (6)$$

$$F_{fric} = F_{max} + kx_{max} \quad (7)$$

$$\zeta = \frac{x_{max}}{2v_i} \sqrt{\frac{k}{m}} \quad (8)$$

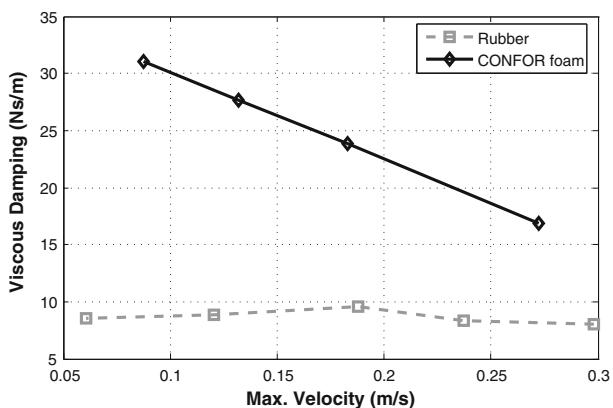
A suspension designed using these criteria provides an almost constant force during landing. It is more robust to variations in initial velocity and returns to its default state as long as  $F_{fric} > -kx_{max} - mg$  (where  $x_{max} < 0$ ). As an example, for a maximum force of 31 N, a spring stiffness of 860 N/m and a initial speed of -2 m/s as parameters, these equations lead to a damping ratio of 0.29, a friction force of 11.2 N and a maximum displacement of 2.3 cm.

#### 4.3 Non-Linear Damping

While the previous sections show what parameters are necessary to provide good spine engagement, the derivation assumes constant value parameters that can be varied independently. Furthermore, the size and weight constraints on a small UAV severely limit the kinds of components that can be used in a suspension and favor lightweight viscoelastic materials.

However, some viscoelastic materials have non-linear properties that can be advantageous for lightweight suspensions. It can be shown, as in [1], that in the case of a simple spring-damper suspension subject to an initial velocity, the force on the spines

**Fig. 6** The damping coefficient of materials varies in different ways. Rubber shows a near constant damping coefficient for the range of speed experienced by the suspension while pink CONFOR foam damping is significantly lower at high velocity



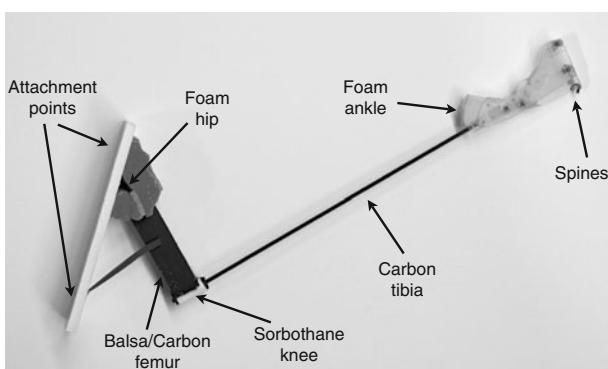
would be constant if the damping coefficient is equal to  $b = (F_{max} - kx(t))/v(t)$ . This means that a low damping coefficient is desirable initially (high velocity at contact), and that it should increase as the plane slows down.

An example of this kind of material can be found in CONFOR foam, a slow-recovery urethane foam. Although not having exactly the damping characteristics required to provide a perfectly constant force, Fig. 6 shows that this material has a damping coefficient that decreases with increasing speed compared to the near constant damping of rubber. These materials were tested by subjecting them to a sine wave of varying amplitude, to keep the amount of damping and spring force roughly proportional during the testing. An Adept One robotic arm was used to generate the motions and forces were measured with a JR3 wrist force sensor. The section tested measured  $25 \times 25$  mm for pink CONFOR foam and  $7 \times 2$  mm for the rubber, both 10 cm in length.

#### 4.4 Suspension Testing

Several foam suspensions were built and tested to obtain a rough estimate of the envelope of possible landing configurations, as the one shown in Fig. 7. No extensive characterization has been done yet, but the maximum values of orientation and velocity were recorded from 30 flights and are summarized in Table 1. The lower

**Fig. 7** Suspension made of slow-recovery urethane foam. The ankle joint provides a fast response to the surface profile, while the hip and knee joint absorb the impact without any rebound



**Table 1** Values of orientation and velocity observed at touchdown during 30 successful landings on concrete wall

Envelop parameters	Minimum	Maximum
Pitch	60 deg	105 deg
Pitch rate	0	200 deg/s
$v_x$	—	3 m/s
$v_y$	0 m/s	2.7 m/s

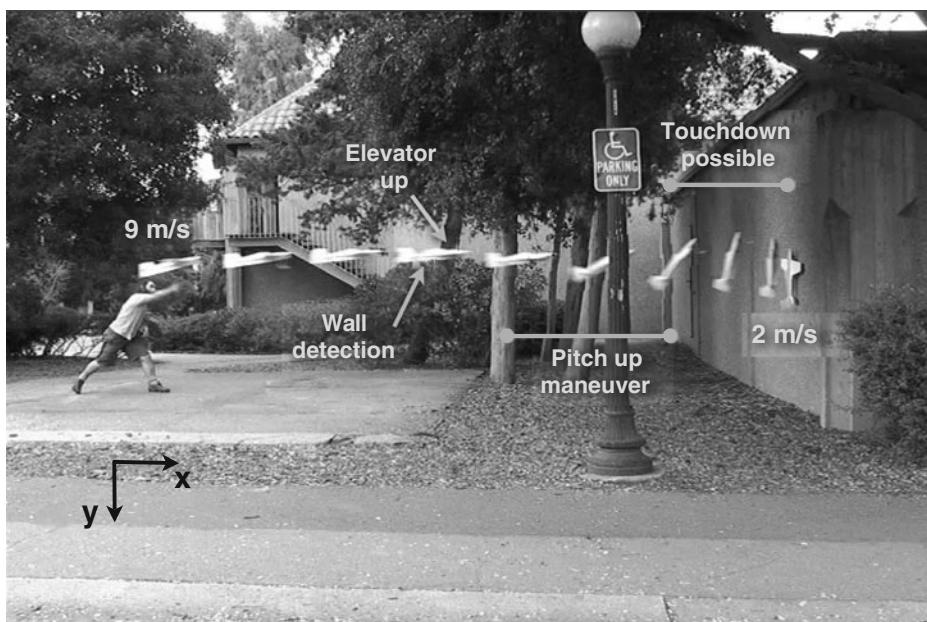
Minimum pitch and minimum  $v_y$  are currently limited by the linkage design

limits of pitch angle and  $v_y$  are currently set by the ankle design joint placement and could be improved in future designs.

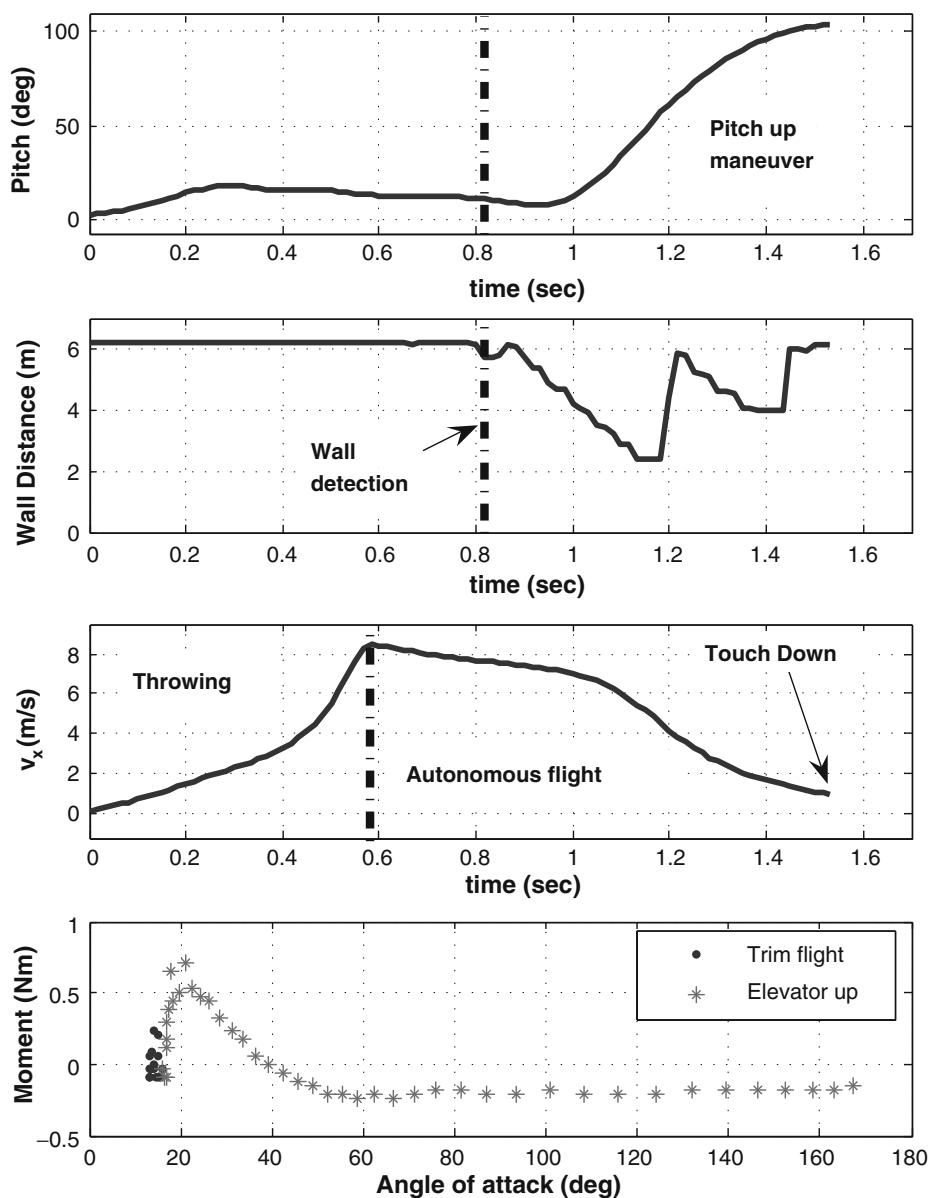
The suspension could probably allow landing with a horizontal velocity,  $v_x$ , higher than 3 m/s, but this hasn't been observed due to drag at high pitch angles drastically decelerating the plane; initial velocities have simply not been large enough to result in  $v_x$  above 3 m/s at contact. Pitch angles higher than 105 degrees also have not been observed, as the plane tends to return to 90 degrees pitch.

## 5 Airplane Trajectory

The relatively wide ranges of orientation and velocity at which the plane can perch impose relatively few constraints on the trajectory. This has allowed us to use the natural dynamics of the airplane platform. As shown in Fig. 8, the natural response



**Fig. 8** Camera frame grab (7.5 Hz) showing the perching maneuver. The glider is launched at 9 m/s, detects the wall 6 m away, pitches up and slows down to 2 m/s before touchdown. During touchdown, the suspension absorbs the impact and provides the necessary motion to engage the spines on the wall



**Fig. 9** Data collected during autonomous perching. The plane is released at 0.58 s, glides toward the wall, detects the wall at 0.82 s (6 m away from the wall) and starts pitching up. Pass 50 degrees, the plane rapidly slow down both its  $x$  velocity and its pitch rate. It then touch the wall at an pitch angle of 105 deg and a  $x$  velocity of 1 m/s. Some ripples are present in the wall distance sensor, thus the importance of having a robust maneuver and suspension. The pitching moment curve shows the trim flight condition (in blue), the high pitching moment created by the elevator up (red curve from 15–40 degrees) and the small negative moment that reduce the pitch rate to zero at about 90 degrees

of the plane flying at about 9 m/s to an elevator up (45 degrees) command is as follows:

- The plane rapidly pitches up to 60 degrees (minimum pitch angle for perching) in about 4 m, or 0.35 s, gaining little altitude.
- The drag created by the high pitch angle rapidly slows down the  $x$  velocity of the plane to about 3 m/s. Past 40 degrees of pitch, the pitching moment becomes negative and slowly reduces the pitch rate.
- The plane continues to pitch and maintains a small, positive  $x$  velocity, ready for touchdown.
- This continues for a total travelled distance of about 2 m, or 0.5 s, before the gravity has increased the downward velocity to a point outside the perching envelope.

It is interesting to note that the approach is very short, under 1 s, minimizing the period of time during which a disturbance could affect the plane. Furthermore, the plane initially flies toward the wall at 9 m/s and always maintains a slight forward velocity, minimizing the effect of any disturbances encountered.

This trajectory is possible for two reasons: the large elevator of the plane and the slight negative moment present at high angle of attack. As shown on Fig. 9, this glider flies at a trim pitch of 15 degrees but can generate a large pitching moment by commanding the elevator to a 45 degrees position. With the elevator up, the pitching moment remains positive up to about 40 degrees, and then becomes slightly negative, slowing down the pitching motion. The key to a successful maneuver is to create just enough initial angular momentum to reach a pitch angle of 90 degrees with zero pitch rate.

One could also think about a two step maneuver: commanding the elevator up to its maximum to create a higher pitching rate, followed by a second elevator command to slow down the plane to zero pitch rate when a pitch of 90 degrees is reached. This would have the advantage of keeping the maneuver as short as possible, but would require pitch sensing and hasn't been implemented yet.

## 6 Sensors and Control

The trajectory shown on Fig. 8 is particularly simple from a sensing point of view. Because the plane stops its pitching motion when it approaches 90 deg (a perfect pitch angle for landing), this maneuver can be executed without any pitch sensor. The only sensing requirement for a fully autonomous landing is to measure the distance from the wall to trigger the maneuver at the right time. Because of the large available touchdown region, the sensor doesn't need to be particularly accurate nor to monitor the wall position as the plane is pitching up. Furthermore, since the wall distance sensor is only used for triggering the maneuver, any time delay in the sensor does not affect the maneuver as long as the delay is known.

Considering the delay in the servo controlling the elevator (roughly 0.1 s, or 1 m), the distance to pitch up (4 m) and trying to land in the middle of the touchdown region (2 m wide), the maneuver should start from about 6 m away from the wall. The entire maneuver is open-loop, consisting of just moving the elevator up to 45 deg when the plane is approximately 6 m away from the wall. Fortunately, the

LV-MaxSonar-EZ1 ultrasound sensor has a range of 6.45 m, a 20 Hz update rate, is relatively accurate, and has a mass of only 7 g.

The plane also has a 3-axis accelerometer (ADXL330) and two 2-axis rate gyroscope (IDG300) onboard. These sensors are not used for control, but are useful to measure the motion of the airplane for analysis, as shown in Fig. 9. The accelerometer measurements are first combined with the rate gyro by using a second order complementary filter:

$$\theta_{\text{comp}} = \frac{(\tau s + 1)^2}{(\tau s + 1)^2} \theta = \frac{\tau s}{(\tau s + 1)^2} \dot{\theta}_{\text{rate gyro}} + \frac{2\tau s + 1}{(\tau s + 1)^2} \theta_{\text{gravity}} \quad (9)$$

This filter combines the pitch measurement from the low frequency signal from the accelerometer (gravity measurement) with the high frequency rate gyro measurement to create a signal that doesn't drift, is immune to other acceleration than gravity and responds to fast change. The frequency at which the transition occurs is chosen by the parameter  $\tau$  and has been experimentally chosen to be 0.1. Finally, using the pitch angle, the measurement from the accelerometer can be integrated to get the  $x$  and  $y$  velocities. Examples of these measurements can be seen in Fig. 9.

## 7 Results and Future Work

The early result reported in this paper is an integrated system. The non-linear suspension made from slow-recovery urethane is lightweight and robust while minimizing the impact forces and providing the proper engaging motion for the spines. Most importantly, the suspension-spines combination allows for a wide envelope of incoming velocities and orientations at touchdown which can be easily reached by using the natural dynamics of the airplane. The high initial positive pitching moment created by the upward elevator command is balanced by the small negative recovery moment that maximizes the plane's time in a favorable landing attitude. A typical perching maneuver, from wall detection to touchdown, lasts less than one second and reduces the speed of the airplane from 9 m/s to less than 2 m/s at touchdown. Figure 9 shows data collected during an autonomous landing and Fig. 8 shows a frame capture of the landing.

With such a system, it is possible to perch both autonomously and manually. A human can quickly learn the timing of the maneuver and perch the airplane on a wall. Approximately 30 successful landings have been performed so far: 20 manually operated and 10 autonomously controlled. The success rate is approximately 80% and it is possible to achieve many successive landings without having to tune or repair the system. The main cause of failure is false wall detection from the sensor, causing an early pitch-up maneuver which results in the plane not reaching the wall. The current distance sensor is operating near its maximum range, and the maneuver needs to be started as soon as something is detected. It is thus difficult to apply any kind of filter to prevent false detection and increase robustness.

The second cause of failure is from the plane pitching up too late, thus hitting the wall at a pitch angle of less than 60 degrees and breaking the suspension. This failure mode has been observed during manual landing, but has never been observed with autonomous perching.

The resulting perching system consisting of the foam suspension, spines and ultrasonic sensor weighs only 28 g, a small percentage of the 375 g weight of the total airplane. With optimization of the spines and structure, we believe that the weight can be reduced to less than 20 g.

## 8 Conclusions and Future Work

This paper presents the design of an integrated system allowing a small, fixed-wing plane to land and perch on vertical surfaces. The motivations are to greatly increase mission life and provide the plane with a stable, secure location that is relatively free of debris. The ability to grip vertical surfaces relies upon arrays of compliant microspines, adapted from climbing robots. The particular requirements of the spines for reliably engaging and gripping a surface lead to corresponding requirements for the incoming velocity and orientation of the plane and, most importantly, to requirements for a suspension that will absorb energy, maintain a steady engagement force, and prevent bouncing.

We present the design of a nonlinear damped suspension that meets these requirements, allowing a very simple fixed-wing glider to land with a relatively high success rate.<sup>1</sup>

Immediate work on the perching system will include increasing the wall sensor robustness and optimizing the trajectory to perform the maneuver in a shorter distance and maximize the amount of time over which touchdown is possible. We also plan to study the effect of adding a propeller and performing the perching maneuver when subjected to sidewinds.

Looking further ahead, a number of extensions are desirable to convert this technology into a practical solution for small UAVs. First, as soon as the plane comes to rest, it is desirable to engage a second set of spines that pulls upward, in opposition to the first set. By increasing the internal force between these opposed sets of spines, it becomes possible to sustain a much larger normal force due, for example, to gusts of wind. In preliminary tests, forces of several Newtons in the normal direction have been achieved.

The strategy of landing on buildings can also be extended to perching on other vertical surfaces such as tree trunks, which are actually easier to grip but much less regular, requiring a greater suspension travel. Another interesting possibility is to use directional dry adhesives, as used in gecko-inspired climbing robots [11] for climbing surfaces such as glass and smooth panels. The dry adhesives are conceptually similar to spines, but have narrower tolerances in terms of the required approach velocity and loading strategy; consequently they will require a more exact suspension design and more attention in controlling the approach velocity of the plane.

Finally, we need to address the ability to resume flight. One solution may be to use a small actuator to store elastic energy in the suspension “legs” and jump off the wall in a flight that is initially inverted, rolling to an upright position once away from the wall.

<sup>1</sup>A version of the same spined toes and leg suspension has also been taken to MIT and retrofitted to the planes reported in [5], with similar results.

**Acknowledgements** Alexis Lussier Desbiens is supported by the Natural Sciences and Engineering Research Council of Canada and the Organization of American States, with additional support from DARPA DSO. We would also like to thank Taylor Cone and the members of BDML (Alan Asbeck, Barrett Heyneman, Dan Aukes and others) at Stanford for all their help in conducting the experiments reported here.

## References

1. Akella, P.N.: Contact mechanics and the dynamics of manipulation. PhD in Mechanical Engineering, Stanford University (1992)
2. Asbeck, A.T., Kim, S., Cutkosky, M.R., Provancher, W.R., Lanzetta, M.: Scaling hard vertical surfaces with compliant microspine arrays. *Int. J. Rob. Res.* **25**(12), 14 (2006)
3. Byrnes, G., Lim, N.T.L., Spence, A.J.: Take-off and landing kinetics of a free-ranging gliding mammal, the Malayan colugo (*Galeopterus variegatus*). *Proc. R. Soc. Lond., B Biol. Sci.* **275**(1638), 1007–1013 (2008)
4. Caple, G., Balda, R.P., Willis, W.R.: The physics of leaping animals and the evolution of preflight. *Am. Nat.* **121**, 455–467 (1983)
5. Cory, R., Tedrake, R.: Experiments in fixed-wing uav perching. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference (2008)
6. Frank, A., McGrew, J.S., Valenti, M., Levine, D., How, J.P.: Hover, transition, and level flight control design for a single-propeller indoor airplane. AIAA Guidance, Navigation and Control Conference (2007)
7. Green, W., Oh, P.: A mav that flies like an airplane and hovers like a helicopter. In: Advanced Intelligent Mechatronics. Proceedings (2005)
8. Green, W., Oh, P.: A fixed-wing aircraft for hovering in caves, tunnels, and buildings. In: American Control Conference (2006)
9. Green, W., Oh, P.: Autonomous hovering of a fixed-wing micro air vehicle. In: IEEE International Conference of Robotics and Automation (2008)
10. Illingworth, L., Reinfeld, D.: Vortex attractor—US 6,565,321 B1. United States Patent, p. 40 (2003)
11. Kim, S., Spenko, M., Trujillo, S., Heyneman, B., Santos, D., Cutkosky, M.R.: Smooth vertical surface climbing with directional adhesion. *IEEE Transactions on Robotics* **24**(1), 65–74 (2008)
12. Paparazzi: Paparazzi, the free autopilot. <http://paparazzi.enac.fr> (2008)
13. Paskins, K.E., Bowyer, A., Megill, W.M., Scheibe, J.S.: Take-off and landing forces and the evolution of controlled gliding in northern flying squirrels *glaucomys sabrinus*. *J. Exp. Biol.* **210**(8), 1413–1423 (2007)
14. Roberts, J., Cory, R., Tedrake, R.: On the controllability of fixed-wing perching. In: American Controls Conference (2009)
15. Spenko, M., Haynes, G., Saunders, J., Cutkosky, M.R., Rizzi, A., Full, R.: Biologically inspired climbing with a hexapedal robot. *Journal of Field Robotics* **25**, 223–242 (2008)
16. Wickenheiser, A., Garcia, E.: Longitudinal dynamics of a perching aircraft. *J. Aircr.* **43**, 1386–1392 (2006)
17. Wickenheiser, A., Garcia, E.: Perching aerodynamics and trajectory optimization. In: Proceedings of SPIE (2007)
18. Wickenheiser, A.M., Garcia, E.: Optimization of perching maneuvers through vehicle morphing. *J. Guid.* **31**(4), 815–823. doi:10.2514/1.33819 (2008)

## Automating Human Thought Processes for a UAV Forced Landing

Pillar Eng · Luis Mejias · Xi Liu · Rodney Walker

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 12 November 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper describes the current status of a program to develop an automated forced landing system for a fixed-wing Unmanned Aerial Vehicle (UAV). This automated system seeks to emulate human pilot thought processes when planning for and conducting an engine-off emergency landing. Firstly, a path planning algorithm that extends Dubins curves to 3D space is presented. This planning element is then combined with a nonlinear guidance and control logic, and simulated test results demonstrate the robustness of this approach to strong winds during a glided descent. The average path deviation errors incurred are comparable to or even better than that of manned, powered aircraft. Secondly, a study into suitable multi-criteria decision making approaches and the problems that confront the decision-maker is presented. From this study, it is believed that decision processes that utilize human expert knowledge and fuzzy logic reasoning are most suited to the problem at hand, and further investigations will be conducted to identify the particular technique/s to be implemented in simulations and field tests. The automated UAV forced landing approach presented in this paper is promising, and will allow the progression of this technology from the development and simulation stages through to a prototype system.

**Keywords** UAV · Forced landing · Path planning · Control ·  
Multi-criteria decision making

---

P. Eng (✉) · L. Mejias · X. Liu · R. Walker

Australian Research Centre for Aerospace Automation, Queensland University of Technology,  
Brisbane, Queensland, 4001, Australia  
e-mail: p.eng@qut.edu.au

L. Mejias  
e-mail: luis.mejias@qut.edu.au

X. Liu  
e-mail: xi.liu@qut.edu.au

R. Walker  
e-mail: ra.walker@qut.edu.au

## 1 Introduction

The field robotics community has existed for over 25 years and has made good progress in the areas of ground, underwater and aerial robotics. During this time, maturity in Information and Communication Technologies (ICT) and sensor technologies has brought, to a certain extent, the dream of commercial field robots to reality. There are numerous prototype field robot systems deployed for military experiments, with some robots even seeing routine use (particularly aerial surveillance robots). Many of these systems have been rapidly pressed into service due to operational demands in times of conflict, rather than through careful development of the design requirements. Military experiences with aerial robots, or Unmanned Aerial Vehicles (UAVs) have also indicated the difficulty of integrating a field robot into an environment where failure of the robot can harm the general public. The key being that, particularly in the case of a UAV, the pertinent safety regulations describe human-centred capabilities. We have learnt in recent times that it is impossible to directly apply these regulations to an autonomous system, and the problem remains in how to integrate fundamentally new technology into a highly regulated human-centred environment where failure can lead to loss of human lives. In our case, this means integrating UAVs into an environment predominated by human pilots and human air traffic controllers, where a mid-air collision between a UAV and a passenger aircraft could have catastrophic results both in the air and on the ground.

The universal position of the safety regulators is to require developers of aerial robots to prove that their systems have equivalence to human performance and to their human-centred safety regulations [1]. In this regard, algorithms for UAV Sense-and-Avoid and Force Landings are recognized as two major enabling technologies that will allow the integration of UAVs into civilian airspace [2]. In the former case, the aircraft must be able to reliably detect and avoid collision with both stationary and moving objects in its path of interest, which may/may not announce their position. However, the assumption is that the robot is still capable of powered, controlled flight. In the latter case, the aircraft is forced to perform an unplanned landing due to the occurrence of some onboard emergency, such as engine, systems, sensors or control surface failure. A forced landing due to engine failure is commonly practiced by pilots during flight training and for ongoing safety certification, and the ability to conduct a safe landing in such situations is primarily used to benchmark performance of the manned aviation industry. This procedure involves firstly visual estimation of wind conditions and recognition of appropriate landing sites, then the formulation of a descent trajectory which accounts for wind changes as well as the glide range and manoeuvrability of the aircraft. On final approach to land, the pilot must also avoid trees, power lines, buildings and other obstacles which may have been invisible from the air [1]. Note also that there is limited possibility for replanning the path, since the aircraft is devoid of thrust control and is continually descending. As many of the same problems confronting manned aircraft also affect robotic aircraft, we believe that UAVs must be capable of safe flight termination following an engine failure, as a UAV plummeting uncontrollably into the middle of a busy freeway or a school yard is a risk that the public will be unwilling to accept.

To date, no commercial system exists that allows a UAV to autonomously select the safest emergency landing area in an unknown environment. The most commonly employed method to allay the severity of a UAV forced landing is the use of

parachutes or parafoils to retard the rate of descent, while still providing some degree of controllability for the aircraft [3]. Whilst this concept is attractive in that it still enables limited vehicle controllability even when both the engine and control surfaces have failed, it is highly susceptible to wind gusts and other atmospheric effects which may adversely affect the final impact point. Having a parachute or parafoil onboard also adds to the weight and complexity of the aircraft. Other safety systems currently available allow the UAV to fly towards a pre-defined safe ditching area selected from a database of such, known locations. However, these systems must be preprogrammed with up-to-date information, thus requiring a continuous communications link between a human operator and the air vehicle to ensure that the latter will not attempt landing at an unsuitable location. To date, the only reported successful UAV forced landing involves the U.S. Air Force Global Hawk, which performed a gliding descent under remotely-piloted control to an emergency airstrip in 2006 [4].

An alternative would be to have an automated system onboard the UAV which can process information in a way similar to human pilots, during emergency situations that require the aircraft to land. Here, we have restricted our discussion to the case of engine failure only, and assumed that the onboard avionics and flight control surfaces are still intact for glided flight. Further, to simplify the planning process, we have also omitted obstacles in the flight path and assumed that wind velocities can be estimated by onboard instruments (albeit with certain errors). Such a system is currently under development in this study, and is divided into the three research areas of:

1. Automated visual identification and classification of UAV forced landing sites;
2. Automated multi-criteria decision making for high-level reasoning during the descent; and
3. Automated path planning, guidance and control for descent and landing;

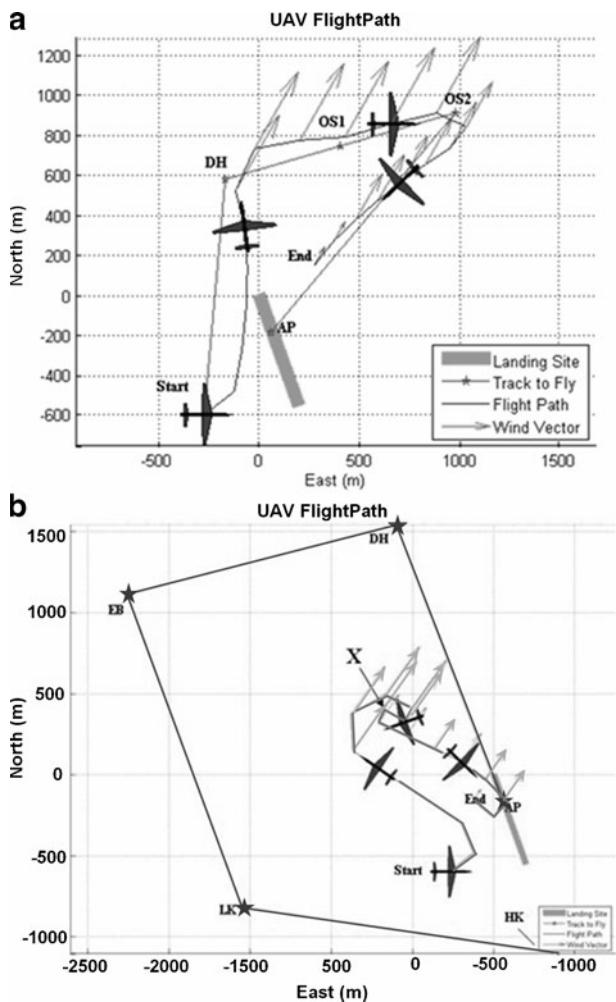
The site identification and classification component uses computer vision and onboard sensors to quickly identify suitable landing areas, and is described in detail in [5, 6]. This paper will present the current research progresses in the path planning, guidance and control component, as well as in the multi-criteria decision making component, respectively. Recommendations will also be given on how to further improve and enhance this research.

## 2 Path Planning, Guidance and Control

The current work progress in this area has involved mainly the simulation of path planning and control strategies using MATLAB. The advantages of simulation are that it simplifies the debugging of code, allows repeatable comparison of different planning and control scenarios, and allows analysis of the UAV response under ideal conditions that set the benchmark performance to be pursued in later experiments.

Previously, two algorithms derived from piloted forced landing procedures as outlined in [1] were developed and tested in simulation using the MATLAB computing program. Algorithm 1 attempted to guide the aircraft (a model of an Aerosonde UAV provided by MATLAB), along a predetermined circuit to the touchdown point on the desired landing site while correcting for the wind on course (Fig. 1a).

**Fig. 1** Path planning simulation results showing flight path in top view for **a** the path described using Algorithm 1, and **b** the path described using Algorithm 2



A number of predefined paths were available and the UAV could choose which path to follow depending on the wind conditions and its proximity to the landing site. The second algorithm did not restrict the aircraft to a predetermined circuit; instead, the UAV was allowed to construct its own path depending on the wind conditions and its ability to reach a certain waypoint (Fig. 1b). This ability was determined from the aircraft glide slope, which was a function of the current wind condition. In both cases, flight stability for the aircraft was maintained using a cascade of Proportional-Integral-Derivative (PID) controllers and obstacles in the flight path were not considered. In addition, both algorithms utilized the great-circle navigation method [1], together with wind triangle calculations [1], to navigate between waypoints. Although a flat-earth approximation was sufficient, the great-circle method was chosen as the basic MATLAB source code had already been written for a different project at ARCAA, and it was decided to extend this work to reduce the development time. The wind velocities supplied to the model reflected

average wind measurements recorded for Brisbane from 1950 to 2000 (available on the Australian Government Bureau of Meteorology website [7]), and an internal AeroSim function then used these velocities to calculate wind shear and turbulence effects on the aircraft using a von Karman approximation. To test the effectiveness of the two algorithms, a Monte Carlo simulation with 100 forced landing scenarios was set up in which the aircraft position and bearing, as well as the wind velocities were randomised. The results showed that using Algorithm 2, the UAV was able to land within the designated area 52% of the time, compared to 26% using Algorithm 1. In addition, Algorithm 2 produced a lower miss distance of less than or equal to 200 m from the touchdown point, compared to the miss distance of Algorithm 1, which was less than or equal to 400 m. A major factor affecting the miss distance was attributed to the strong winds modelled (up to 8 m/s), which were often greater than the forward speed of the aircraft. The fact that the UAV could not adjust its airspeed to counter changing wind conditions, but only its heading, could also have contributed to the large miss distances. Full details of these early developments can be found in [2, 8].

Thus, we have endeavoured to overcome these limitations in our current design approach. Firstly, we have replaced the previous Aerosonde model with a 6 degree-of-freedom (6-DOF) model of a Boomerang radio-controlled aircraft, which represents the UAV platform to be used in future flight tests. The model was constructed using the classic coefficient build-up method [9] as part of an undergraduate project at the Queensland University of Technology (QUT). Specifically, the challenge we face is that of how to guide an unpowered, fixed-wing aircraft to arrive at a specific point in space (approach point) where it is aligned with the crash site for landing/ditching, and at a certain airspeed and heading while accounting for any kinodynamic constraints, regardless of the ambient wind conditions. Note that in this case, the approach point is likened to the touchdown point in our previous experiments, but we have classified the planning and guidance involved from the approach point to the touchdown point as outside the scope of the current research. In designing the planning, guidance and control algorithms we have also assumed that a feasible landing area exists and that the desired final approach point, airspeed and heading are supplied by the multi-criteria decision making algorithm.

## 2.1 Path Planning

Numerous robotics path planning techniques are presented in the literature, and a comprehensive summary of existing methods can be found in [10]. Since a gliding, fixed-wing aircraft can achieve only forward motion and is also limited by constraints in its turn and descent rates, we have found that trajectories derived from Dubins curves [11] present one of the simplest solution that satisfies these constraints. We have also assumed that the aircraft can achieve a nominal lift-to-drag ratio of 9:1 in planning the path, meaning that for every 1000 ft loss in altitude, the aircraft glides 9000 ft.

Initially, a 2D Dubins path is constructed having the form:

$$\{L_\alpha R_\beta L_\gamma, R_\alpha L_\beta R_\gamma, L_\alpha S_d L_\gamma, L_\alpha S_d R_\gamma, R_\alpha S_d L_\gamma, R_\alpha S_d R_\gamma\} \quad (1)$$

in which L and R correspond to left and right turns at a bank angle that does not exceed the maximum bank angle of the aircraft, S corresponds to flying in a straight

line, and  $\alpha, \gamma \in [0, 2\pi)$ ,  $\beta \in (\pi, 2\pi)$ , and  $d \geq 0$ . The radii of the arcs were calculated using the equation:

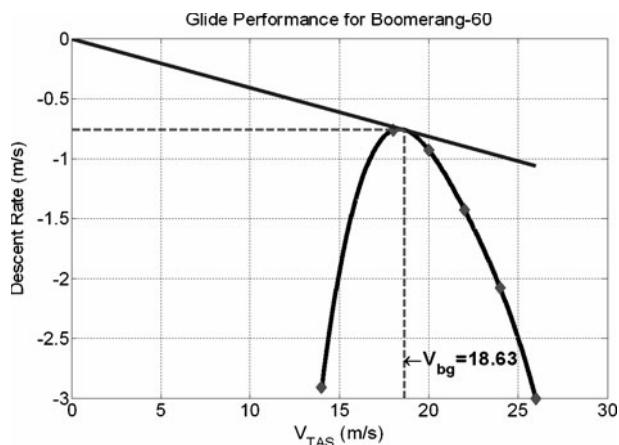
$$R_{0,f} = \frac{V_{TAS}^2}{g \times \tan(\phi_{0,f})} \quad (2)$$

where  $R_{0,f}$  are the initial and final radii of the arcs of circumference,  $V_{TAS}$  is the True Airspeed of the aircraft,  $g$  is the gravitational acceleration constant ( $9.80665 \text{ m/s}^2$ ), and  $\phi_{0,f}$  are the initial and final bank angles respectively, which can be different.

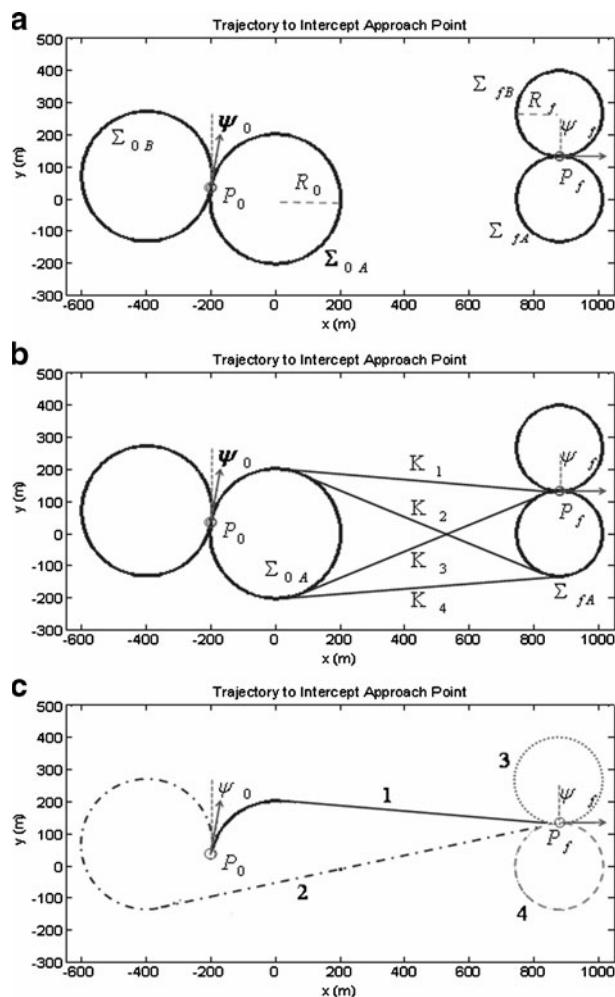
To simplify the path planning process, we have taken  $V_{TAS}$  to be the best glide speed,  $V_{bg}$  of the aircraft, which gives the greatest straight line flight distance in still air from the potential energy of height. As shown in Fig. 2, we can estimate  $V_{bg}$  by first fitting a curve (black) through the descent rates at various airspeeds (red diamonds). The best glide speed is then obtained by drawing the blue line from the origin tangent to the curve, giving  $V_{bg}$  as 18.63 m/s. The descent rates were calculated in simulation by setting the wind speed to zero and commanding the aircraft to fly continuously in a box circuit, with a different airspeed selected for each leg of the circuit.

Once the radii are determined, the optimal 2D path is obtained with a geometrical construction adapted from [12]. Initially, two circles with radii  $R_0$  are drawn containing the starting point  $P_0$  and a vector pointing along the aircraft's initial heading  $\psi_0$  (Fig. 3a). The circumferences of the circles are denoted by  $\sum_{0A}$  and  $\sum_{0B}$ . Next, the same process is repeated at the goal point  $P_f$  with the final aircraft heading  $\psi_f$ , and circumferences  $\sum_{fA}$  and  $\sum_{fA}$ . Following this, tangent lines are constructed that join the circumferences of these circles, such as depicted in Fig. 3b for  $\sum_{0A}$  and  $\sum_{fA}$ . Considering Fig. 3b, we readily observe that there are four paths connecting  $P_0$  to  $P_f$ , where a path is formed by the union of an arc on the circumference  $\sum_{0A}$ , a segment  $K$  on one of the four tangent lines, and finally an arc on the circumference  $\sum_{fA}$ . However, only one of these paths,  $\Gamma_{AA}$  is compatible with the initial and final

**Fig. 2** Speed polar diagram for a Boomerang 60 size UAV, showing how the best glide speed ( $V_{bg}$ ) is obtained



**Fig. 3** **a** Step 1 of generating the 2D path. **b** Step 2 of generating the 2D path. **c** Step 3 of generating the 2D path. Four plausible paths are obtained; the optimal path is path no. 1



headings of the UAV (Fig. 3b and c). In a similar way, three other paths  $\Gamma_{AB}$ ,  $\Gamma_{BA}$  and  $\Gamma_{BB}$  can be obtained—the optimal path is the shortest between  $\Gamma_{AA}$ ,  $\Gamma_{AB}$ ,  $\Gamma_{BA}$  and  $\Gamma_{BB}$  and is depicted as a thick, solid line in Fig. 3c.

Following the construction of  $\Gamma_{xy}$ , and given the distance  $d_{tgt}$  (distance from the initial point of failure to the approach point), we can then obtain the path angle:

$$\gamma_{xy} = \tan^{-1} \left( \frac{z_{arc_f} - z_{arc_0}}{d_{tgt}} \right) \quad (3)$$

which allows the UAV to descend from an altitude of  $z_{arc_0}$  to  $z_{arc_f}$ . To ensure stability,  $\gamma_{min} \leq \gamma \leq \gamma_{max}$ . However, if the difference in altitude between the start and end positions should result in the maximum allowable path angle being exceeded, one of the other suboptimal paths can be selected to lose the approximate amount of altitude required. Other options include enlarging  $R_0$  and/or  $R_f$ , as well as

commanding the aircraft along a helical trajectory (similar to a spring) to lose excess altitude, before joining the path at the start of the first arc.

To form the 3D path, we note that a gliding aircraft that is rolled into a steady, coordinated turn at a constant bank angle  $\phi$  and flies at a constant descent angle will trace a helical path  $\gamma$  on an imaginary cylinder with radius  $R$ . Thus, the 3D path can be formed by a straight line at a constant pitch angle that joins two arc sections. In order to simplify the design, we have not included the  $\{L_\alpha R_\beta L_\gamma\}$  or  $\{R_\alpha L_\beta R_\gamma\}$  types paths, these will be addressed in future work. The relationship between  $\phi$  and  $\gamma$  is given by:

$$\cot_{\gamma_{0,f}} = \frac{V_{TAS}}{V_s} \cos_{\phi_{0,f}} \quad (4)$$

and  $V_s$  is the descent/sink rate of the powerless aircraft as shown in Fig. 2. Now, the altitude lost while transversing the two arc sections can be calculated as:

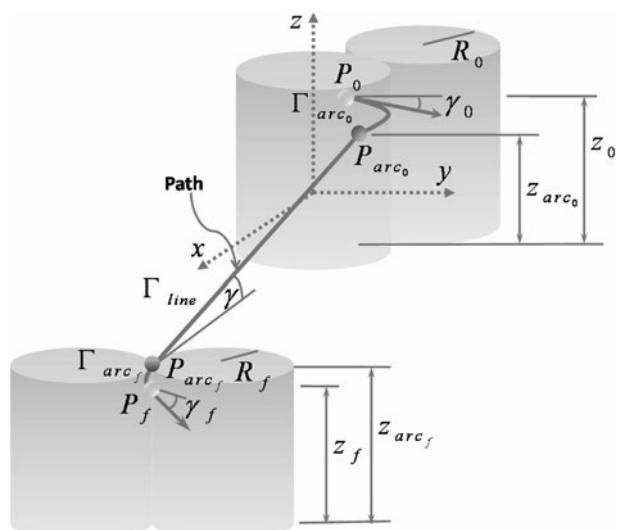
$$S_{0,f} = \frac{1}{2\pi} \frac{\|\sum_{0,f}\|}{R_{0,f}} S_{\phi_{0,f}} \quad (5)$$

Giving the altitude where the arc sections join the 3D line as:

$$\begin{aligned} z_{arc_0} &= z_0 - S_0 \\ z_{arc_f} &= z_f + S_f \end{aligned} \quad (6)$$

where,  $z_0$  is the altitude at the start of a forced landing, and  $z_f$  is the desired altitude to achieve at the final approach point. Given the terminal points on the arcs  $P_{arc_0} = [x_{arc_0}, y_{arc_0}, z_{arc_0}]$  and  $P_{arc_f} = [x_{arc_f}, y_{arc_f}, z_{arc_f}]$ , it is then a straightforward process to obtain  $\Gamma_{line}$ . The relationship between the different elements of the 3D path ( $\Gamma = \Gamma_{arc_0} \cup \Gamma_{line} \cup \Gamma_{arc_f}$ ) is illustrated in Fig. 4.

**Fig. 4** Relationship between elements of the generated 3D flight path. The generated path is  $\Gamma_{arc_0} \cup \Gamma_{line} \cup \Gamma_{arc_f}$



## 2.2 Guidance and Control

Our guidance algorithm is composed of both a lateral and longitudinal component. The lateral component is based on the work presented in [13, 14]; however, we have also built upon this algorithm to encapsulate wind information in the guidance logic, rather than merely treating wind as an adaptive element for the control system. This addition to the control law has demonstrated robust performances for linear path following in strong winds. We have also made a simple assumption in formulating the guidance equation for circular path following, such that the guidance logic is simplified while still providing acceptable performance. In addition, we have implemented a longitudinal guidance and control element that caters for the dynamics of powerless flight. Following well-established aerospace control design procedures [9], we have chosen to separate our design into two modes: an inner control loop that provides aircraft dynamic stability, and an outer guidance loop that generates the required acceleration and position commands to follow a path.

In the lateral guidance mode, a reference point  $P_{ref}$  is first selected on the desired trajectory, and this reference point is then used to generate a lateral acceleration command. As shown in Fig. 5,  $P_{ref}$  is located a distance  $L_1$  ahead of the vehicle and, at each point in time, a circular path (dotted line) can be defined by the position of  $L_1$ , the vehicle position, and tangential to  $V$ , the aircraft velocity vector.

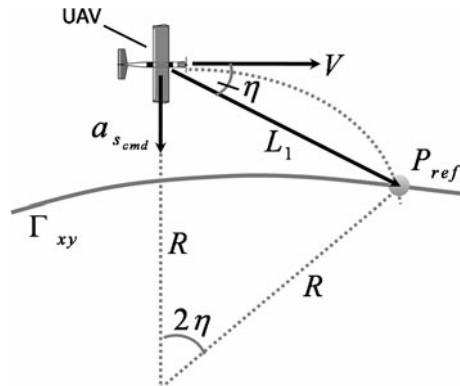
The acceleration required to follow the instantaneous circular segment, for any radius  $R$ , is then given by:

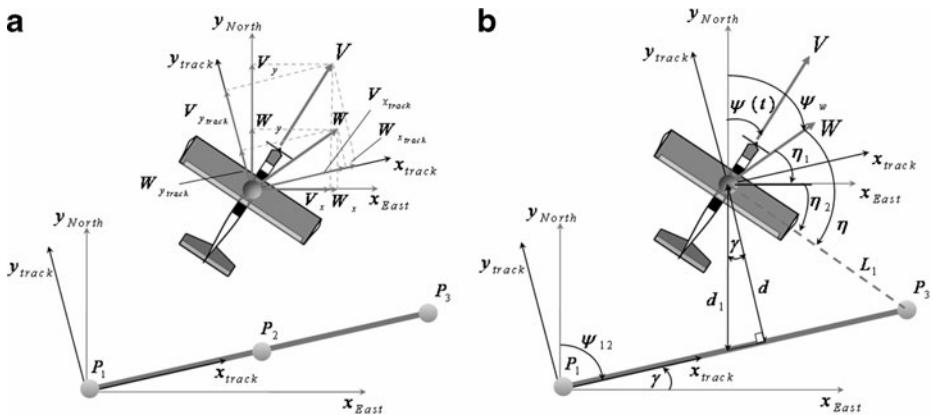
$$a_{s_{cmd}} = \frac{V^2}{R} = 2 \frac{V^2}{L_1} \sin \eta \quad (7)$$

Thus, the guidance logic will tend to rotate the aircraft such that its velocity direction will always approach the desired path at an angle that is proportional to the relative distance between vehicle and path. For following a straight line, we can model the vehicle kinematics as shown in Fig. 6.

Now, consider the UAV in a straight glide at an arbitrary position relative to the path between waypoints  $P_1$  and  $P_3$ , and at a heading  $\psi$  (Fig. 6a). Given the aircraft velocity and position in the North, East reference frame, and the angular

**Fig. 5** Diagram showing the lateral guidance logic





**Fig. 6** Vehicle kinematics for straight line following, showing **a** the relationship between the aircraft velocity  $V$  and wind velocity  $W$ , and **b** the relationship between the aircraft bearing  $\psi(t)$ , the path bearing  $\psi_{12}$ , the wind bearing  $\psi_w$ , and the angle  $\eta$ . In addition, the relationship between the cross-track error  $d$ , the hypotenuse  $d_1$  and the look-ahead distance  $L_1$  is also shown

measurements defined in Fig. 6b, we can obtain the position and velocity components in the  $\{x_{track}, y_{track}\}$  reference frame by:

$$\begin{aligned}\vec{V}_{track} &= T_\phi \vec{V} \\ \vec{W}_{track} &= T_\phi \vec{W}\end{aligned}\quad (8)$$

where the rotation matrix

$$T_\psi = \begin{bmatrix} \cos(\psi_{12} - \pi/2) & -\sin(\psi_{12} - \pi/2) \\ \sin(\psi_{12} - \pi/2) & \cos(\psi_{12} - \pi/2) \end{bmatrix} \quad (9)$$

The cross-track velocity can then be written as:

$$\begin{aligned}\dot{y}_{track} &= V_{y_{track}} + W_{y_{track}} \\ &= -V\sin(\psi(t) - \psi_{12}) - W\sin(\psi_w - \psi_{12})\end{aligned}\quad (10)$$

And assuming  $\eta$  is small, we get:

$$\sin\eta = \eta_1 + \eta_2 \quad (11)$$

and

$$\begin{aligned}\eta_1 &\approx \frac{d}{L_1} \\ \eta_2 &\approx \frac{d}{V}\end{aligned}\quad (12)$$

where the cross-track velocity  $\dot{y}_{track}$  has been relabeled as  $\dot{d}$ , and  $d$  is the cross-track error. If N and E are the North and East coordinates, we can obtain  $d$  by letting

$$d_1 = N_{aircraft} - \tan(\gamma) E_{aircraft} \quad (13)$$

and

$$d = d_1 \cos \gamma \quad (14)$$

Combining Eq. 7 to 12, we obtain:

$$a_{s_{cmd}} = 2 \frac{V}{L_1} \left( \dot{d} + \frac{V}{L_1} d \right) \quad (15)$$

For following an arc of circumference, we can model the vehicle kinematics as shown in Fig. 7.

Here, the angles  $\eta_1$  and  $\eta_2$  are assumed to be small, but  $\eta_3$  is not necessarily small,

$$\eta_1 \approx 0, \eta_2 \approx 0, |\eta_3| >> 0. \quad (16)$$

As shown in [13], we can estimate

$$\sin\eta_3 = \frac{L_1}{2R} \quad (17)$$

and define

$$c \equiv \cos\eta_3 \approx \sqrt{1 - \left(\frac{L_1}{2R}\right)^2} \quad (18)$$

Then, using small angle assumptions for  $\eta_1$  and  $\eta_2$ , we can show

$$a_{cmd} = \frac{2V^2}{L_1} \{ \eta_1 \cos \eta_3 + \eta_2 \cos \eta_3 + \sin \eta_3 \} \quad (19)$$

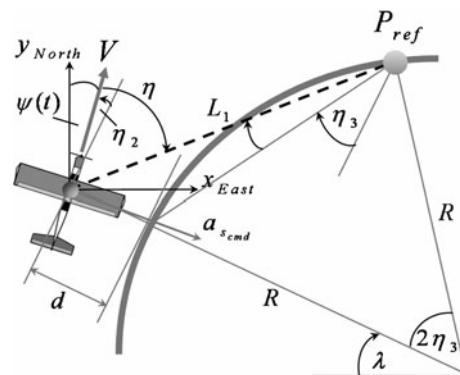
with

$$\eta_1 \approx \frac{d}{L_1} \cos \eta_3 , \quad \dot{d} = V \sin \eta_2 \approx V \eta_2 \quad (20)$$

and applying Eqs. 17 and 18, Eq. 19 becomes

$$\frac{2V^2}{L_1} = \frac{2V^2 c^2}{L_1^2} d + \frac{2Vc}{L_1} \dot{d} + \frac{V^2}{R} \quad (21)$$

**Fig. 7** Vehicle kinematics for circular path following



Now, if we assume that a good yaw damper can be designed to damp the aircraft Dutch roll motion and reduce the sideslip to zero, then we can neglect the second term on the R.H.S of Eq. 21 to obtain;

$$a_{s_{cmd}} = \frac{2V^2}{L_1} \sin\eta = \frac{2V^2 c^2}{L_1^2} d + \frac{V^2}{R} \quad (22)$$

To convert the acceleration to a desired roll command and simplify calculations, we assume that the aircraft maintains sufficient lift to balance weight, even though banked at an angle  $\phi$ . This gives

$$L \cos\phi = W = mg, \quad L \sin\phi = ma \quad (23)$$

and

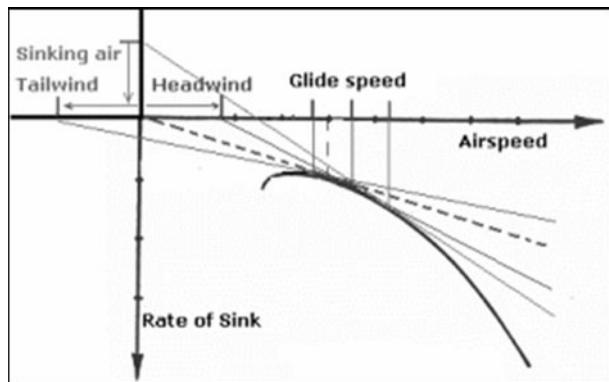
$$\phi_d = \tan^{-1} \left( \frac{a_s}{g} \right) \quad (24)$$

In actual fact, a gliding aircraft will never overcome gravity, but will descend with a vertical velocity  $h$ ; this is an inherent limitation in the forced landing problem. Notice also that here we have not included additional terms in the equation for wind effects, as we have done with the case of following a straight line. The reason is that in this case the vehicle ground speed (as a surrogate for inertial velocity) is used for  $V$  in Eq. 7 at each instant in generating the acceleration command. Since the ground speed is a function of the airspeed and wind speed, the guidance logic accounts for the inertial velocity changes due to wind and adapts to the situation accordingly.

Following the design of the outer loop guidance logic, we then proceeded to use a PID control approach as it is widely used in UAV applications [9], and would also provide a seamless integration with an off-the-shelf UAV flight computer for field testing. Here, the difference between the desired and actual roll angle is used to produce an error signal that activates the aileron control servo. The controller also includes roll rate feedback for improved damping, and saturation limits on the outputs to avoid instability. A classical yaw damper was also included in the design which has enabled the aircraft to follow straight and circular paths with greatly reduced oscillations. In the longitudinal guidance mode, the difference between the desired and actual aircraft flight path angles is used to generate the desired pitch angle, which in turn controls the elevators for longitudinal path following. The desired path angle is composed of  $\gamma_{\phi_{0,f}}$ , the path angles corresponding to the sections of helices in the 3D path, and  $\gamma_{line}$ , the path angle of the line segment.

To cater for the effects of wind while gliding, we use the well-known MacCready theory (discussed in [15]). From the speed polar diagram (Fig. 8), we see that to counter a headwind, a glide speed above  $V_{bg}$  must be selected; this increases the ground speed and allows the aircraft to penetrate further through the air. Similarly, the glide speed must be reduced below  $V_{bg}$  when flying downwind to avoid overshooting the target waypoint. In a tailwind, the starting point on the horizontal axis (airspeed) is shifted left by a distance equal to the magnitude of the wind speed, and a line is drawn from this point tangential to the curve to obtain the desired glide speed. For a headwind the starting point is shifted to the right, for sinking air it is shifted upwards on the vertical axis, and for rising air it is shifted downwards. Using the speed polar, one can also determine the speed-to-fly in different combinations of vertical and horizontal winds when both exist. Finally, to convert the speed-to-fly into

**Fig. 8** Using the speed polar diagram while gliding in winds  
(Image obtained from [12])



the desired path angle, we simply take the inverse tangent of the slope corresponding to the tangent line.

In designing the inner loop pitch controller, we have used the pitch error to control the desired elevator deflection and the pitch rate to provide additional damping. As before, PID controllers are used to regulate the system, and limiters are placed on the outputs to prevent saturating the elevator servomotor, which we have assumed to possess similar characteristics to that controlling the ailerons. We have also implemented scheduling of the PID and  $L_1$  gains such that the required path following performance could be met. For testing purposes, this performance is specified as having a horizontal (lateral) and vertical (longitudinal) cross-track error at the approach point of no greater than 2 m (approx. 6.56 ft), and a maximum vertical and horizontal deviation of no greater than 30 m (approx. 100 ft) on average. These upper and lower bounds are commonly accepted as the performance standard for general aviation aircraft [1].

### 2.3 Results and Discussion

A total of 128 simulations have been performed to gain an initial understanding of the efficacy of our planning, guidance and control algorithms. In each case, the aircraft initial altitude and wind conditions were allowed to vary, while the initial and final aircraft headings and positions were kept constant. A Monte Carlo simulation using a larger input set will be performed at a later date. For these experiments, we have also assumed perfect knowledge of the wind conditions and no errors in the sensor readings. A sample of the test data is included in Table 1.

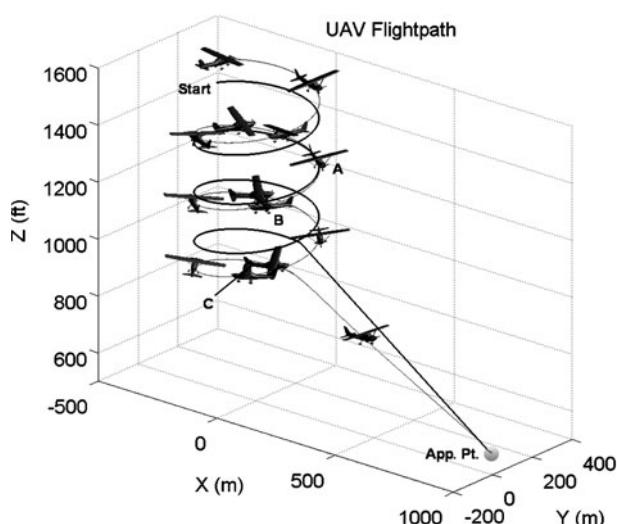
Table 1a shows the initial and desired final aircraft positions used by the path planner in constructing a trajectory from an altitude of 1640 ft to 500 ft in calm conditions, while Table 1b shows the positions for a descent from 850 ft to 500 ft in winds, with a maximum wind speed of 7 m/s. Note that as we are concerned with gliding flight, the required airspeed at the approach point is absorbed by the flight path angle requirement at this point.

As depicted in Fig. 9, in planning a path for a forced landing from a high initial altitude (Table 1a), the algorithm is able to generate the required number of helix spirals to “bleed off” the excess altitude, before joining the spirals with an arc-line-arc Dubins path (solid black line). This ensures that constraints on the flight path

**Table 1** Sample test data for the path planning and path following algorithms

	Initial cond.	Final cond.
a) No wind		
x (m)	-199	885
y (m)	37	133
z (ft)	1640	500
$\psi$ (deg)	10	90
$\gamma$ (deg)	0	-6
b) With wind		
x (m)	-199	885
y (m)	37	133
z (ft)	850	500
$\psi$ (deg)	10	90
$\gamma$ (deg)	0	-6

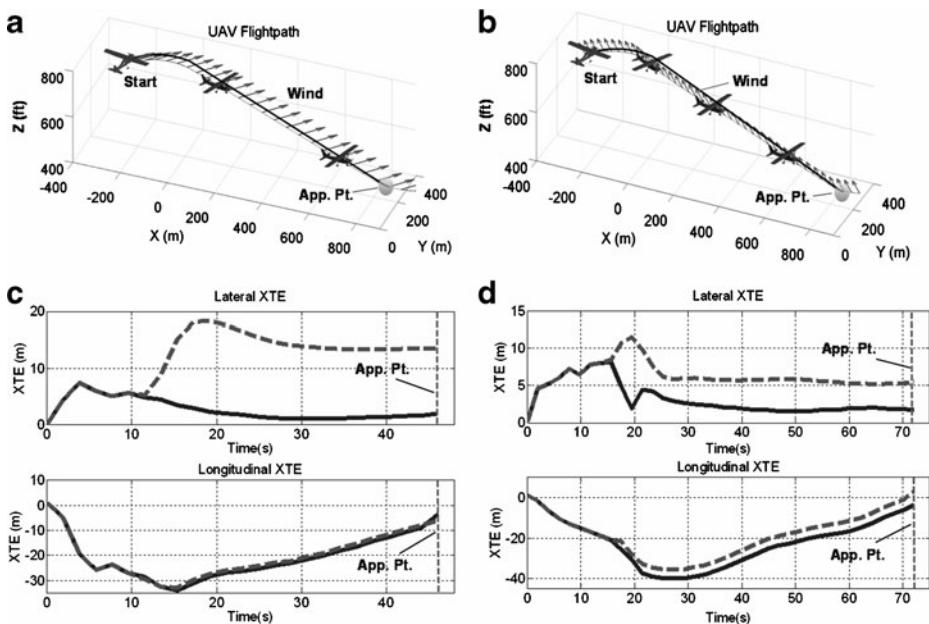
angle are satisfied and prevents excessive stress to the UAV structure. The horizontal and vertical track errors at the approach point are 0.3 m and 1.3 m, which are well within our stated tolerances and comparable to the results obtained for other flight path angles. The apparent difference in altitudes between the aircraft and the path at the start is due to the planning algorithm rounding the required number of helix spirals to the nearest complete ( $360^\circ$ ) spiral turn, in order to preserve the desired initial heading. As can be seen from the diagram, the aircraft is still able to converge onto the path at Point A despite this offset in altitudes. However, at Point B the aircraft descends below the path and reaches a maximum vertical deviation of 600 ft at Point C (the horizontal error at this point was approximately 9 m), but recovers to intercept the approach point with the errors stated above. The poor longitudinal path following performance in the first halve of the descent may be attributed to the assumption that the airspeed remains constant at the best glide speed, as it is not possible to predict in advance what the actual airspeed due to the control actions may be. In an actual descent, the airspeed is allowed to vary and this gives rise to a

**Fig. 9** Path planning and following for a forced landing from a high initial altitude in nil wind conditions

non-uniform loss in altitude. In addition, an aircraft rolled into a continuous banking motion will also experience some amount of yawing motion called sideslip, no matter how good the yaw damper may be, and this in turn increases the altitude lost. Thus the amount of loss in altitude factored into the path planning equations is ideal at best, and does not fully take into account the associated loss in altitude due to varying airspeeds and other atmospheric effects. Hence, the current solution relies on the path following algorithm being robust to these uncertainties in guiding the aircraft to the desired approach point. A possible alternative is to increase the path angle of the initial helices to more closely match that of the straight segment, and/or increase the radii of the helices such that the number of spirals is reduced. These will help reduce the amount of altitude loss due to sideslips and a prolonged banking action.

Next, we show the performances of our path following algorithm in winds (Table 1b) and compare the results with those obtained using the original path following algorithm in [13], hereby referred to as the unmodified non-linear guidance (UNG) algorithm. Two different wind scenarios are chosen for illustration. Figure 10a shows the aircraft able to follow the desired path (solid black line) in a 6 m/s South-South-Westerly wind (green arrows), while Fig. 10b shows a similar case albeit with the wind coming from the South-South-East. As shown in the top halves of Fig. 10c and d, using our path following algorithm, the lateral error (blue line) at the approach point was 1.8 m and 1.2 m respectively; compare this with the performance of UNG, which has lateral errors (red dashed line) of 14.1 m and 6 m for the two different wind conditions. Although UNG did not include a longitudinal path following component, we have nonetheless also plotted the vertical track error to show what might have transpired had that lateral guidance algorithm, coupled with our longitudinal guidance algorithm, been used to follow the path. As shown in the lower halve of Fig. 10c and d, the vertical track errors for our guidance algorithm is approximately 1.2 m and 1.5 m respectively for the two wind conditions, while that for the case of UNG coupled with our longitudinal guidance algorithm is 1.5 m and 2.4 m. Thus it can be clearly seen that our path following algorithm outperforms the UNG algorithm. In addition, the average lateral and longitudinal path errors for both cases are well within 100 ft, which as mentioned earlier is commonly accepted as the maximum allowable path deviation for general aviation aircraft.

From the simulations, we have also observed that the path following algorithm is able to contain the errors at the approach point, and within the stated tolerances, for wind speeds not exceeding 7 m/s. In stronger winds, these errors can degrade to  $>20$  m horizontally, and up to 5 m vertically, or the aircraft may lose control and crash. A possible explanation for these factors is that since we are using an unpowered, scaled model of a real aircraft as our UAV platform, the small size and weight (5.55 kg) of the aircraft, as well as the limited thrust available means that it cannot achieve the necessary control authority to overcome strong winds and gusts. We have also noticed from simulations that the vertical track error at the approach point is  $>7$  m in sinking air of 1 m/s. Once again, this relatively poor performance may be due to the structural and aerodynamic factors stated above. However, when compared to our previous work as discussed in Section 2, we find that the performances of our current algorithms are far superior. We believe that this improved performance may be largely due to the fact that we have allowed the airspeed and flight path angle to vary to counter different wind scenarios. In addition, the simulation results presented here indicate that the path planning, guidance and



**Fig. 10** Path planning and following in ambient winds, showing **a** Aircraft response in 6 m/s wind from SSW; **b** Aircraft response in 6 m/s wind from SSE; **c** Horizontal and vertical track errors for case a; and **d** Horizontal and vertical track errors for case b

control techniques developed are suitable for further assessment in Monte Carlo simulations and even flight trials.

#### 2.4 Future Work

In the future, we plan to enhance the control elements to be robust to uncertainties in sensor errors and wind measurements. Secondly, we will experiment with different techniques to reduce the vertical track error when the aircraft is following a helix spiral, and several options have already been proposed in the preceding section. Thirdly, we desire to extend the path planning component to include a re-planning capability for cases where the efficacy of the original plan is reduced or nullified. This could be due to winds that are simply too strong for the aircraft to overcome, or when a better landing site is identified as the aircraft nears the ground. These modular enhancements will also be progressively flight tested.

### 3 Multi-criteria Decision Making

One of the most important aspects in the initial stages of a forced landing is to make the right decision regarding which site to land on and how to approach the chosen landing site. These two aspects are closely related to the multi-criteria decision analysis and the path planning, guidance and control component of the overall approach, respectively. This section will shed light on the main concepts behind the

challenging decision-making process, which in reality is continuously validated and updated throughout much of the descent if new information should yield a more appropriate landing site.

### 3.1 Multiple Criteria

According to the Australian Civil Aviation Safety Authority's latest Visual Flight Rules flight guide [1] there are seven "S" criteria to selecting the best site for a forced landing, in addition to the critical factor of wind strength and direction. These are:

- Size
- Shape
- Surface
- Slope
- Surrounding (risk to nearby infrastructure/population)
- S(c)ivilisation (proximity to aid)
- Sun (reduced visual capability)

These are the primary elements which a human pilot use when making decisions on the selection of a preferred landing site. When applied in the context of UAVs, many of these factors still hold their significance, and a number of other variables also come into consideration which are not explicitly stated for piloted aircraft. These include the aircraft dynamics, the uncertainty of sensor data and wind estimation, etc.

Also to be considered is the geometrical relationship between the various candidate sites. As the aircraft descends, the number of landing site options will rapidly decrease. Thus, it is generally better to plan the approach towards several possible sites in close proximity than to one that is isolated, as this keeps multiple landing site options open for as long as possible. This is important so as to have several options if obstacles are detected on the candidate landing sites at lower altitudes.

The number of structures and the population density that lies in the descent path to each site must also be accounted for if possible, as it would be safer to fly over empty terrain than a populated area, in case further mishaps occur. These points, along with other factors which remain to be identified, will be evaluated to reach an optimal, verifiable decision on which candidate landing site the aircraft will aim for.

Further investigations will be conducted in order to identify any other influences that affect this decision process, possibly including surveys and simulations involving experienced pilots and/or UAV controllers.

### 3.2 Multiple Objectives

The complexity of the forced landing decision process due to multiple criteria is further increased by multiple objectives that must be met. In many cases, these objectives may be conflicting, and thus compromises must be made to accommodate the achievement of the most critical objective/s.

According to the Civil UAV Capability Assessment [16], in the event of an emergency landing the UAV needs to be able to respond according to the following objectives in the following order:

1. Minimize expectation of human casualty;
2. Minimize external property damage;

3. Maximize the chance of aircraft survival; and
4. Maximize the chance of payload survival.

In many scenarios, the best landing site for meeting objectives 3 and 4 may compromise the more important objectives 1 and/or 2, or vice versa. This complex trade-off process between the risks and uncertainties involved with each possible choice is an example of multiple objectives that the system must evaluate between and is what makes this problem difficult.

### 3.3 Decision Considerations

The descent planning and decision making modules will initially have preplanned contingency plans from map data to give fast, reflex responses to emergencies that guide the aircraft towards known landing sites, or large flat areas based on the slope map data.

The guidance and control module (discussed in the previous section) will constantly make estimates of the wind speed and direction, which will be taken as input for decision making. The aircraft dynamics will also be accounted for and necessary constraints applied when judging the feasibility of a decision.

The decision making algorithm must run in real time; in the highly dynamic and critical situation of an aircraft forced landing, it is important that the decision making component is able to respond to changing circumstances in a timely manner. The decision making algorithm will also need to be deterministic. Even in uncertain conditions, where the site identification phase is not able to provide candidate landing sites with an adequate level of confidence, or in situations where there are no desirable alternatives, a single decision outcome must always be reached.

As the aircraft descends, the landing site identification and classification module will continually analyse the terrain the aircraft is flying over. Possible landing sites, buildings, and roads will be identified, including the associated uncertainties of objects in each map. Armed with this information, the decision making module will then be able to continually validate and update its decision in real-time. Another consideration arising from this is the length of time committed to a decision, which essentially means that the longer one particular alternative has been committed to, the higher the cost to switch to another alternative. Thus, to a certain extent, past decision outcomes must be taken into consideration.

It is expected that uncertainties will reduce as the aircraft descends, however the options available will also reduce. It may be possible that an initially selected landing site will eventually be deemed unsuitable by the site selection module, and an alternative must be sought after. It is the responsibility of the decision making module to be prepared for such situations by maximizing the number of alternative choices available.

### 3.4 Decision Making Methods

From the literature review, it was concluded that there are essentially two broad classes of multi-criteria decision analysis methods; one follows the outranking philosophy and builds a set of outranking relations between each pair of alternatives, then aggregates that according to some suitable technique. The other essentially involves determining utility/value functions for each criterion, and finding out the

‘utility’ of each alternative based on each criterion, then aggregating those with a suitable technique to find the overall utility of the alternative.

Many of the existing techniques are not designed for ‘decision making’; rather they are intended as ‘decision aid’ methods, and hence some only generate additional information to aid the human decision maker, who makes the final decision. Decision making is in many ways also a subjective matter. As discussed earlier, in most cases there is no ‘best’ decision, and it is subject to the preference of the human decision makers. Due to the nature of the forced landing problem, where decisions made could potentially lead to damage to property or even harm life, it is critical then that the decision making system to be developed must be based on justifiable and generally accepted preference data. This means that the technique chosen should require preference data that is clear and understandable by people who don’t understand the mathematics of method, and also that the technique should be as transparent as possible for purposes of accountability. Additional requirements used to evaluate the various techniques include the ability to handle uncertainty in terms of input data, and the assumptions made regarding the decision problem. A number of the techniques are currently under trial, such as PROMETHEE [17] and MAUT [18]. PROMETHEE is an outranking method that requires relatively simple preference data in terms of criterion weights and preference functions. MAUT, which is based on Expected Utility Theorem [19] makes the assumption of independence, meaning that only the probability distribution of risks of individual criterion are considered, and they don’t affect each other. This may be unrealistic for the forced landing scenario, yet it can be addressed by using Fuzzy Choquet Integrals [20], which addresses synergy and redundancy between criteria.

The technique of most interest does not readily fit in to either of the main families of multicriteria decision analysis methods, namely, the Decision Rules Approach [21], and the one specifically described here is the Dominance-based Rough Set Approach (DRSA) [22]. This method takes samples of decisions made by human experts, and analyses them to determine the minimum set of decision rules expressed in the form of “if, then” statements. These statements are then used to evaluate the alternatives in the multi-criteria decision problem, and aggregated with an appropriate aggregation technique such as the Fuzzy Net Flow Score [23]. There is the capacity to deal with inconsistent preference information from the human decision makers by using the rough sets, and fuzzy sets can be implemented to address uncertainty in the input data. This method is the most transparent and understandable of all those investigated so far, and further comparisons will be made between these decision making techniques under consideration.

#### 4 Conclusions

In this article we have presented in detail the design, implementation and simulation of the path planning, guidance and control strategies for an automated forced landing system for UAVs. We have also introduced the multi-criteria decision making approach that aims to emulate human pilot thought processes in the event of a forced landing. Simulated results of the planning, guidance and control module demonstrate the ability of the gliding aircraft to follow the prescribed path in winds, with average path deviation errors that are comparable to or even better than that of manned,

powered aircraft. Although the path planning, guidance and control strategies were derived from existing work in the literature, we have enhanced these algorithms and added further functionality to suit the case of an unpowered fixed-wing UAV forced landing. Some examples include the extension of Dubins curves to 3D while accounting for aircraft dynamics, and encapsulating wind information in the guidance logic, rather than treating wind as an adaptive element to be overcome by the control system. However, many complex decision making problems still remain to be investigated. These problems involve multiple conflicting objectives, and it is often true that no dominant alternative will exist that is better than all other alternatives. Generally it is impossible to maximize several objectives simultaneously, and hence the problem becomes one of value tradeoffs. The tradeoff issue often requires subjective judgement of the decision maker, and there may be no right or wrong answers to these value questions. Currently, we are in the process of testing several multi-criteria decision making techniques, and simultaneously integrating strategies between the path planning, guidance and control and the decision making modules before progressing to flight trials. It is believed that the automated UAV forced landing approach presented here will allow the progression of this technology from the development and simulation stages through to a prototype system that can demonstrate its effectiveness to the UAV research community.

**Acknowledgements** The work presented in this paper is sponsored by the Australian Postgraduate Award, the Queensland University of Technology (QUT) Vice-Chancellors Top-Up Award and the CSIRO ICT Scholarship Top-Up Award.

## References

1. CASA, Visual Flight Rules Guide, 2nd ed. Civil Aviation Safety Authority Australia, Canberra (2007)
2. Fitzgerald, D., Mejias, L., Eng, P., Liu, X.: Towards flight trials for an autonomous uav emergency landing using machine vision. In: Australasian Conference on Robotics and Automation, Brisbane, Australia, December 2007
3. Redelinghuys, C.: A flight simulation algorithm for a parafoil suspending an air vehicle. *J. Guid. Control Dyn.* **30**, 791–803 (2007)
4. Froeschner, C.P.: UAS—Flying safety. <http://findarticles.com/> (2008). Accessed 27 Nov 2008
5. Fitzgerald, D.: Candidate landing site selection for uav forced landings using machine vision. Ph.D. dissertation, Queensland University of Technology. School of Engineering Systems (2007)
6. Mejias, L., Fitzgerald, D., Eng, P., Liu, X.: Aerial vehicles. I-Tech Education and Publishing, 2009, ch. Forced Landing Technologies for Unmanned Aerial Vehicles: Towards Safer Operations, pp. 415–440
7. Australian Bureau of Meteorology: The wind across Australia. <http://www.bom.gov.au/> (2007). Accessed 24 June 2007
8. Eng, P., Mejias, L., Fitzgerald, D., Walker, R.: Simulation of a fixed-wing uav forced landing with dynamics path planning. In: Australasian Conference on Robotics and Automation, Brisbane, Australia, December 2007
9. Nelson, R.C.: Flight Stability and Automatic Control, 2nd ed. McGraw-Hill Higher Education, New York (1998)
10. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York (2006)
11. Dubins, L.E.: On curves of minimal length with a constraint on average curvature with prescribed initial and terminal positions and tangents. *Am. J. Math.* **79**, 471–477 (1957)
12. Ambrosino, G., Ariola, M., Ciniglio, U., Corrado, F., Pironti, A., Virgilio, M.: Algorithms for 3-d uav path generation and tracking. In: 45th IEEE Conference on Decision & Control, pp. 5275–5280. San Diego, CA (2006)

13. Park, S.: Avionics and control system development for mid-air rendezvous of two unmanned aerial vehicles. Ph.D. dissertation, Department of Aeronautics and Astronautics, M.I.T (2004)
14. Park, S., Deyst, J., How, J.P.: Performance and lyapunov stability of a nonlinear path-following guidance method. *J. Guid. Control Dyn.* **30**, 1718–1728 (2007)
15. Brandon, J.: Forced landing procedures. <http://www.auf.asn.au/emergencies/forcedlanding.html>. Accessed 25 March 2007
16. Cox, T.H., Nagy, C.J., Skoog, M.A., Somers, I.A.: Civil uav capability assessment. NASA, Tech. Rep., draft Version (2004)
17. Brans, J.-P., Mareschal, B.: Promethee methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, New York (2005)
18. Dyer, J.S.: Maut—multiattribute utility theory. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, New York (2005)
19. von Neumann, J., Morgenstern, O.: *Theory of Games and Economic Behavior*. Princeton University Press, Princeton (1944)
20. Grabisch, M., Roubens, M.: Application of the choquet integral in multicriteria decision making. In: Grabisch, T.M.M., Sugeno, M. (eds.) *Fuzzy Measures and Integrals - Theory and Applications*, pp. 348–374. Physica Verlag, Würzburg (2000)
21. Greco, S., Matarazzo, B., Slowinski, R.: Decision rule approach. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, New York (2005)
22. Graco, S., Matarazzo, B., Slowinski, R.: Rough approximation by dominance relations. *Int. J. Intell. Syst.* **17**, 153–171 (2002)
23. Bouyssou, D.: Ranking methods based on valued preference relations: a characterization of the net flow method. *Eur. J. Oper. Res.* **60**(1), 61–67 (1992). Available <http://www.sciencedirect.com/science/article/B6VCT-48NBG9R-1TT/2/3590a17dce5c9186172b2099f2c50f47>

# Autonomous Autorotation of Unmanned Rotorcraft using Nonlinear Model Predictive Control

Konstantinos Dalamagkidis · Kimon P. Valavanis ·  
Les A. Piegl

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 16 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** Safe operations of unmanned rotorcraft hinge on successfully accommodating failures during flight, either via control reconfiguration or by terminating flight early in a controlled manner. This paper focuses on autorotation, a common maneuver used to bring helicopters safely to the ground even in the case of loss of power to the main rotor. A novel nonlinear model predictive controller augmented with a recurrent neural network is presented that is capable of performing an autonomous autorotation. Main advantages of the proposed approach are on-line, real-time trajectory optimization and reduced hardware requirements.

**Keywords** Autonomous autorotation · Unmanned rotorcraft · Nonlinear model predictive control

## Nomenclature

$A$	Rotor disc area
$C$	Constraint function
$C_d$	Blade drag coefficient
$C_{d,0}$	Average blade drag coefficient
$C_l$	Blade lift coefficient
$C_{l,a}$	Lift curve slope
$C_T$	Thrust coefficient
$E$	Neural network epoch size
$f_e$	Equivalent unit drag coefficient area

---

K. Dalamagkidis (✉) · L. A. Piegl  
Computer Science and Engineering Department, University of South Florida,  
4202 East Fowler Avenue, ENB 118, Tampa, FL 33620, USA  
e-mail: kdalamag@mail.usf.edu

K. P. Valavanis  
Department of Electrical and Computer Engineering, University of Denver,  
Clarence M. Knudson Hall, 300, 2390 S. York Street, Denver, CO 80208, USA

---

$f_i$	Inflow velocity factor
$f_r$	Ratio of maximum to nominal rotor rpm
$g$	Acceleration of gravity
$I_R$	Rotor moment of inertia
$J$	Jacobian matrix
$L$	Cost function
$M$	Helicopter mass
$N_c$	Control horizon
$N_s$	Prediction horizon
$R$	Rotor radius
$T$	Thrust
$t_f$	Final time
$t_s$	Sampling period
$u$	Action
$v$	Velocity
$v_H$	Helicopter sink rate
$w$	Weight factor
$\chi$	Auxiliary vector
$x$	State
$z$	Helicopter altitude
$z_0$	Helicopter initial altitude

### Greek Letters

$\gamma$	Learning rate
$\theta$	Blade pitch at 3/4 of its length
$\kappa$	Induced power correction factor
$\lambda$	Inflow ratio
$\rho_\alpha$	Air density
$\sigma$	Rotor solidity factor
$\Omega$	Rotor speed of rotation
$\Omega_0$	Nominal rotor speed of rotation

### Subscripts

$h$	At hover
$i$	Induced
$M$	Maximum
$m$	Minimum
$s$	Steady-state

## 1 Introduction

Small unmanned rotorcraft are very attractive both in the military and the civilian application domains mainly due to two key characteristics; maneuverability and portability. Specifically helicopters can maneuver in tight spaces, hover over areas of interest and can take-off from and land almost anywhere. On the other hand they have been described as “ungainly, aerodynamic mavericks” [4], since they exhibit a complex aerodynamic performance that is extremely difficult to accurately predict.

Penetration of such systems into the market is contingent on resolution of safety issues among others. This paper will not go into details on these issues, the interested reader is referred to the literature [2, 5–7, 9, 10, 16]. Nevertheless, solution of such issues will undoubtedly involve provisions to handle on-board failures in a manner that minimizes the risks to the public and third party property to acceptable levels. The focus of this work is on failures that would jeopardize or completely preclude continued flight under manual control, but can be accommodated by an on-board emergency flight controller. Such failures include loss of power or control to the tail rotor and possibly main rotor.

In manned helicopters, a pilot faced with such a failure can employ the autorotation maneuver to safely land the aircraft. Due to the aerodynamics of the main rotor, even when no power is supplied to it, it is possible to maintain a steady rate of descent. This is accomplished by using the air flowing through the rotor disk to rotate the main rotor—the reverse process from normal flight. In this case the main rotor acts as a parachute, breaking the helicopter. Just before touchdown the rotor rpm is exchanged for a reduction in the descent rate thus allowing the helicopter to land safely.

The proposed emergency controller is an independent system on-board unmanned helicopters that is capable of performing the autorotation maneuver autonomously. Its primary purpose is to minimize the probability of fatalities or injuries to people on the ground. Secondary goals are to minimize the risk of collision with other aircraft or stationary objects, as well as minimize the damage to the aircraft itself. As a result, vertical autorotation is preferred in this case, to minimize the airspace volume through which the helicopter will need to fly. Additionally the sink rate is reduced during the last 3 m of descent to minimize the effects of impact even if there are people in the area.

## 2 History and State of the Art of Autonomous Autorotation

In 1977, Johnson derived an autorotation model that includes vertical as well as longitudinal movement [12]. The optimal control was derived using a cost function that depended on horizontal and vertical speed at touchdown. The derivation of the control law was based on iterative numerical integration forwards and then backwards between the two boundary points and updating using the steepest descent method. The results were then compared with the performance of a modified Bell OH-58A carrying a High Energy Rotor System (HERS).

A few years later in Stanford, Allan Yeow-Nam Lee improved on Johnson's work by introducing state inequality constraints that were converted to equality using slack variables [13]. The controller was derived by numerical parameter optimization using the Sequential Gradient Restoration technique (an iterative method).

Although Johnson and Lee derived optimal autorotation trajectories, the issue of autonomous autorotation was not addressed until almost a decade later. In Japan, Hazawa et al. [11] derived two autorotation models (one linear and one non-linear) and used a PI controller to land a small unmanned helicopter.

In a continuation of the work of Johnson and Lee, Aponso et al. presented their own method to optimize the trajectory and control inputs for a full-size helicopter during an autorotation landing [3]. The goal of their work was to ensure the survival

of sensitive sensors and data stored on board the helicopter in the case of non-catastrophic failures. A significant drawback of their method is that it precalculates the control inputs and the trajectory before entering the autorotation maneuver and as a result is not robust with respect to modeling errors and outside interference. Because of this mismatch between model and simulation a flare law was necessary, that forces the flare to occur at 30 ft. Their work was evaluated against a high fidelity Bell 206 simulator.

During 2008, two groups presented results for autonomous autorotation using machine learning techniques. In the first approach [1], the controller was trained using pre-recorded pilot reference autorotations that provided a model of the aircraft and the “ideal” trajectory. The landing itself was achieved by forcing the helicopter to hover at 0.5 m. The performance of the controller was demonstrated using a small unmanned helicopter (XCell Tempest). The second approach was a straightforward application of reinforcement learning to train a controller using the Johnson model, cost function and experimental data. The final state-action space has 10 dimensions and was covered using RBFs, whose parameters were updated using backpropagation. After 9000 epochs the number of RBFs was about 19,000 and the success rate around 80%.

### 3 Proposed Approach

Current applications for unmanned helicopters typically require hovering at relatively low altitudes of a few hundred meters. Since the sink rate during autorotation can be significant, the whole maneuver may take only a few seconds to complete making manual intervention difficult if not impossible. Furthermore, on-board processing capacity of small unmanned helicopters is limited, which in turn puts bounds on the computational complexity of the controller if real-time operation is to be achieved. To meet these performance requirements the use of a nonlinear, model-predictive controller augmented by a recurrent neural network is proposed.

The idea behind model predictive control is to start with a fixed prediction horizon ( $N_s$ ), using the current state of the plant as the initial state. An optimal control sequence of length  $N_c$  ( $N_s \geq N_c$ ) is then obtained that minimizes an objective function while at the same time satisfying posed constraints. After applying the first element of that sequence as an input to the plant, the new state is observed and used as an initial state repeating the process.

Model predictive control is used extensively for the control of production processes and the properties of linear model predictive controllers are well understood. In the case of nonlinear problems, linearization techniques are usually preferred because NMPC suffers from the same issues as any nonlinear optimization approach; convergence, stability and computational complexity. Nevertheless many NMPC-based solutions to various nonlinear problems have been proposed in the literature, with encouraging results.

The NMPC problem can be expressed as an optimization problem, specifically:

$$\min L(\mathbf{u}, \mathbf{x}) \quad \text{s.t. } \mathcal{C}(\mathbf{u}, \mathbf{x}) \leq 0 \quad (1)$$

where a control sequence  $\mathbf{u}$  is determined that minimizes the objective function  $L$  and satisfies the inequality constraints  $\mathcal{C}$ . This optimization problem needs to be solved each time a new state is observed. To solve such problems, Xia et al.

have proposed a series of recurrent neural networks [17–20]. The idea behind their approach is to build a neural network that models an ODE whose equilibrium point is the optimal solution to the problem (1). This approach has two major advantages; guaranteed exponential convergence in the case of convex problems and fast execution speed when using hardware that can perform parallel computations.

The update rule of the recurrent neural network used in this paper was adapted from [20] and is given by:

$$d \begin{pmatrix} \mathbf{u} \\ \chi \end{pmatrix} = \gamma \begin{pmatrix} -\mathbf{u} + \left( \mathbf{u} - \frac{dL}{d\mathbf{u}} - \frac{d\mathcal{C}}{d\mathbf{u}} \chi \right)^+ \\ -\chi + (\chi - \mathcal{C}(\mathbf{u}))^+ \end{pmatrix} \quad (2)$$

where  $\gamma$  is a learning rate parameter,  $(\cdot)^+$  is an activation function and  $\chi$  is an auxiliary vector with size equal to the number of constraints.

### 3.1 Vertical Autorotation Model

The vertical autorotation model used for the controller is given by:

$$\begin{aligned} \dot{v}_H &= g - \frac{\rho_\alpha A R^2 \Omega^2}{2M} \sigma C_{l,a} \left[ \frac{\theta}{3} - \frac{v_i - v_H}{2\Omega R} \right] - \frac{\rho_\alpha f_e}{2M} v_H^2 \\ \dot{z} &= -v_H \end{aligned} \quad (4)$$

$$\dot{\Omega} = -\frac{\rho_\alpha A R^3 \Omega^2}{I_R} \frac{v_i - v_H}{R\Omega} \sigma C_{l,a} \left[ \frac{\theta}{3} - \frac{v_i - v_H}{2\Omega R} \right] - \frac{\rho_\alpha A R^3 \Omega^2}{8I_R} \sigma C_{d,0} \quad (5)$$

where Eq. 3 is derived from the balance of the three forces acting on the helicopter; the thrust provided by the main rotor, the aerodynamic drag from moving through the air and the weight of the aircraft. Similarly Eq. 5 is obtained from a torque balance between the torque added or subtracted by the inflow and torque subtracted due to blade drag. For the detailed derivation of these equations the reader is referred to [13].

Since the inflow dynamics are generally not measurable during flight, the model used internally by the controller will be simplified to include only the measurable states namely  $v_H$ ,  $z$  and  $\Omega$ . As a result the controller assumes a steady-state inflow velocity, that is:

$$v_i = v_{i,s} = v_{i,h} f_i \quad (6)$$

$$v_{i,h} = \sqrt{\frac{Mg}{2\rho_\alpha A}} \text{ and } f_i = \begin{cases} \kappa + 0.75 \frac{v_H}{v_{i,h}} & , 0 \leq \frac{v_H}{v_{i,h}} \leq \frac{8\kappa}{4\kappa + 1} \\ 7\kappa - 3\kappa \frac{v_H}{v_{i,h}} & , \frac{8\kappa}{4\kappa + 1} \leq \frac{v_H}{v_{i,h}} \leq 2 \\ \kappa \frac{v_H}{2v_{i,h}} - \kappa \sqrt{\left( \frac{v_H}{2v_{i,h}} \right)^2 - 1} & , 2 \leq \frac{v_H}{v_{i,h}} \end{cases}$$

$$v_{i,h} = \sqrt{\frac{Mg}{2\rho_\alpha A}} \text{ and } f_i = \begin{cases} \kappa + 0.75 \frac{v_H}{v_{i,h}} & , 0 \leq \frac{v_H}{v_{i,h}} \leq \frac{8\kappa}{4\kappa + 1} \\ 7\kappa - 3\kappa \frac{v_H}{v_{i,h}} & , \frac{8\kappa}{4\kappa + 1} \leq \frac{v_H}{v_{i,h}} \leq 2 \\ \kappa \frac{v_H}{2v_{i,h}} - \kappa \sqrt{\left( \frac{v_H}{2v_{i,h}} \right)^2 - 1} & , 2 \leq \frac{v_H}{v_{i,h}} \end{cases}$$

The control input is scaled to the [0, 1] range, while the model equations are non-dimensionalized using the nominal rotor angular velocity  $\Omega_0$  and the rotor radius  $R$ :

$$\tau = \frac{\Omega_0 t}{100} \Rightarrow \frac{d}{dt} = \frac{\Omega_0}{100} \frac{d}{d\tau} \quad (7)$$

$$x_1 = \frac{100v_H}{\Omega_0 R} \Rightarrow v_H = \frac{\Omega_0 R}{100} x_1 \quad (8)$$

$$x_2 = \frac{z}{10R} \Rightarrow z = 10Rx_2 \quad (9)$$

$$x_3 = \frac{\Omega}{\Omega_0} \Rightarrow \Omega = \Omega_0 x_3 \quad (10)$$

$$x_4 = \frac{100v_{i,s}}{\Omega_0 R} \Rightarrow v_{i,s} = \frac{\Omega_0 R}{100} x_4 \quad (11)$$

$$u = \frac{\theta + \theta_m}{\theta_M - \theta_m} \Rightarrow \theta = u(\theta_M - \theta_m) + \theta_m \quad (12)$$

The final controller model equations are:

$$\begin{aligned} \dot{x}_1 &= \frac{10^4}{\Omega_0^2 R} g - \frac{\rho_\alpha f_e R}{2M} x_1^2 + \frac{25\rho_\alpha A R \sigma C_{l,a}}{M} x_3(x_4 - x_1) - \frac{29.1\rho_\alpha A R \sigma C_{l,a} \theta_m}{M} x_3^2 \\ &\quad - \frac{29.1\rho_\alpha A R \sigma C_{l,a} (\theta_M - \theta_m)}{M} x_3^2 u \end{aligned} \quad (13)$$

$$\dot{x}_2 = -\frac{1}{10} x_1 \quad (14)$$

$$\begin{aligned} \dot{x}_3 &= -\frac{100\rho_\alpha A R^3 \sigma C_{d,0}}{8I_R} x_3^2 + \frac{\rho_\alpha A R^3 \sigma C_{l,a}}{400I_R} (x_4 - x_1)^2 - \frac{2.91\rho_\alpha A R^3 \sigma C_{l,a} \theta_m}{10^3 I_R} (x_4 - x_1) x_3 \\ &\quad - \frac{2.91\rho_\alpha A R^3 \sigma C_{l,a} (\theta_M - \theta_m)}{10^3 I_R} (x_4 - x_1) x_3 u \end{aligned} \quad (15)$$

There are three types of constraints imposed on the controller. The first concerns the physical limits of the actuator:

$$\theta_m \leq \theta \leq \theta_M \Rightarrow 0 \leq u \leq 1 \quad (16)$$

This type of constraint is easily handled by appropriate design of the neural network activation function. In this case the activation function is a saturation function that limits the input to the range [0, 1], so that the control sequence is always within the actuator limits:

$$(\cdot)^+ = \min(1, \max(\cdot, 0)) \quad (17)$$

The second constraint is incorporated to avoid blade stall:

$$C_T \leq \frac{\sigma}{8} \Rightarrow \frac{1}{2}\sigma C_l \left( \frac{1}{3}u(\theta_M - \theta_m) + \frac{1}{3}\theta_m - \frac{1}{200} \frac{x_4(i) - x_1(i)}{x_3(i)} \right) \leq \frac{\sigma}{8} \quad (18)$$

The last constraint is designed to protect the main rotor from mechanical stress. This is because for very low blade pitch the rotor angular velocity may increase above nominal, possibly damaging the rotor assembly.

$$\Omega \leq f_r \Omega_0 \Rightarrow x_3 \leq f_r \quad (19)$$

where from the literature  $f_r$  typically takes values between 1.05 and 1.25 [3, 14].

### 3.2 Simulation Model

As mentioned earlier the inflow dynamics and the ground effect phenomenon are ignored by the controller. Nevertheless both are taken into account by the simulator developed to test the controller. The response of the induced velocity to thrust changes is not instantaneous but exhibits a dynamic behavior. This behavior can be modeled using an inertia model with an apparent mass equal to 63.7% that of a sphere of air with the same radius as that of the rotor ( $m_a = 0.637 \rho_\alpha \frac{4}{3} \pi R^3$ ) [15]. As a result the thrust will be given by:

$$\begin{aligned} T &= m_a \dot{v}_i + 2\rho_\alpha A v_i (v_i - v_H) \\ &= 0.849 \rho_\alpha A R \dot{v}_i + 2\rho_\alpha A v_i (v_i - v_H) \end{aligned} \quad (20)$$

In steady-state conditions the thrust is given by

$$T = 2\rho_\alpha A (v_i - v_H) v_i \quad (21)$$

which when combined with Eq. 20 produces:

$$\begin{aligned} 2\rho_\alpha A (v_H + v_{i,s}) v_{i,s} &= 0.849 \rho_\alpha A R \dot{v}_i + 2\rho_\alpha A v_i (v_i - v_H) \\ (v_H + v_{i,s}) v_{i,s} &= 0.4245 R \dot{v}_i + v_i (v_i - v_H) \\ \dot{v}_i &= -\frac{2.356}{R} [v_i (v_i - v_H) - v_{i,s} (v_{i,s} - v_H)] \\ \dot{v}_i &= -\frac{2.356}{R} (v_i - v_{i,s}) (v_i + v_{i,s} - v_H) \end{aligned} \quad (22)$$

where  $v_{i,s}$  is the steady state induced velocity typically calculated from empirical models as a function of  $v_H$ . It is obvious that the derived equation has two possible steady state solutions;  $v_i = v_{i,s}$  and  $v_i = v_H - v_{i,s}$  although only the former is of interest. To overcome this problem, it is assumed that  $v_i v_H \simeq v_{i,s} v_H$  and as a result the final induced velocity model is given by:

$$\dot{v}_i = -\frac{2.356}{R} (v_i^2 - v_{i,s}^2) \quad (23)$$

An additional correction factor is required for hovering near the ground, due to a phenomenon called ground effect. Because the rotor wake meets the ground the pressure below the rotor rises resulting in higher thrust generation for the same power. There are several empirical models of ground effects, the one used in this work is given by:

$$\left[ \frac{T}{T_\infty} \right]_{P=\text{const}} = \frac{1}{1 - \frac{\sigma C_i \lambda_i}{4C_T} \left( \frac{R}{4z} \right)^2} \quad (24)$$

## 4 Controller Derivation

The vertical autorotation model can be expressed as a general SIMO nonlinear affine in the control problem given by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

$$\min_{\mathbf{u} \in U} \int_0^{t_f} L(\mathbf{x}, \mathbf{u})$$

which in discrete time becomes:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + t_s \mathbf{f}(\mathbf{x}(t)) + t_s \mathbf{g}(\mathbf{x}(t))\mathbf{u}(t) \quad (25)$$

$$\min_{\mathbf{u}(i) \in U} \sum_{i=0}^{t_f} L(\mathbf{x}(i), \mathbf{u}(i)) \quad (26)$$

To find the optimal sequence using the recurrent neural network of Eq. 2, the  $\frac{dL}{du}$ ,  $\mathcal{C}(\mathbf{u})$  and  $\frac{d\mathcal{C}}{du}$  quantities need first be calculated.

Taking into account that actions can only affect future states and don't depend on past or future actions:

$$\frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} = 0, \forall x(k), u(i) : i \geq k \quad \frac{d\mathbf{u}(k)}{d\mathbf{u}(i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

Differentiating Eq. 25 with respect to a control action  $u(i)$ :

$$\begin{aligned} \frac{d\mathbf{x}(k+1)}{d\mathbf{u}(i)} &= \frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} + t_s J_f(\mathbf{x}(k)) \frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} + t_s J_g(\mathbf{x}(k)) \frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} u(k) + t_s g(\mathbf{x}(k)) \frac{d\mathbf{u}(k)}{d\mathbf{u}(i)} \\ &= [I + t_s J_f(\mathbf{x}(k)) + t_s J_g(\mathbf{x}(k))u(k)] \frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} + t_s g(\mathbf{x}(k)) \frac{d\mathbf{u}(k)}{d\mathbf{u}(i)} \end{aligned} \quad (28)$$

Using Eqs. 28 and 27 the following update rule for calculating the  $\frac{d\mathbf{x}(t)}{d\mathbf{u}(i)}$  from the previous prediction step is obtained:

$$\frac{d\mathbf{x}(t)}{d\mathbf{u}(i)} = \begin{cases} 0 & \text{if } t < i \\ t_s g(\mathbf{x}(k)) & \text{if } t = i \\ [I + t_s J_f(\mathbf{x}(k)) + t_s J_g(\mathbf{x}(k))u(k)] \frac{d\mathbf{x}(k)}{d\mathbf{u}(i)} & \text{otherwise} \end{cases} \quad (29)$$

If an objective function of the following form is assumed:

$$L = \sum_{j=1}^{N_s} L^*(\mathbf{x}(j)) + w \mathbf{u}^T \mathbf{u} \quad (30)$$

where  $L^*$  is a function of the state and  $w$  is a positive weight factor, then:

$$\frac{dL}{d\mathbf{u}} = \sum_{i=1}^{N_s} \left( \frac{\partial L^*}{\partial \mathbf{x}(i)} \frac{d\mathbf{x}(i)}{d\mathbf{u}} \right) + 2w \mathbf{u} \quad (31)$$

The constraints in the vector form required by the recurrent neural network are given by:

$$\mathcal{C}(\mathbf{u}) = \begin{bmatrix} [C_T(i) - \frac{\sigma}{8}]_{1 \leq i \leq N_c} \\ [\Omega(j) - f_r \Omega_0]_{1 \leq j \leq N_c} \end{bmatrix} \quad (32)$$

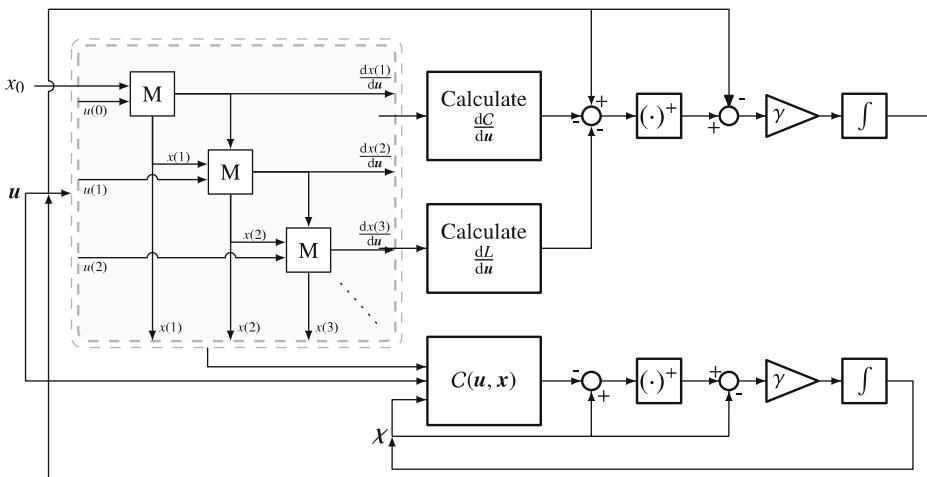
or

$$\mathcal{C}(\mathbf{u}) = \begin{bmatrix} \left[ \frac{\sigma C_{l,a}(\theta_M - \theta_m)}{6} \left( \mathbf{u} + \frac{\theta_m}{\theta_M - \theta_m} - \frac{3}{4} \frac{1}{\theta_M - \theta_m} \frac{1}{C_{l,a}} \right. \right. \\ \left. \left. - \frac{3}{400} \frac{1}{\theta_M - \theta_m} \frac{x_4(i) - x_1(i)}{x_3(i)} \right) \right]_{1 \leq i \leq N_c} \\ [x_3(j) - f_r]_{1 \leq j \leq N_c} \end{bmatrix} \quad (33)$$

Finally differentiating Eq. 33 with respect to the control sequence gives  $\frac{d\mathcal{C}}{du}$ :

$$\frac{d\mathcal{C}}{du} = \begin{bmatrix} \frac{\sigma C_{l,a}(\theta_M - \theta_m)}{6} [I]_{N_c \times N_c} \\ -\frac{\sigma C_{l,a}}{800} \left[ \frac{\frac{dx_4(i)}{dx_1(i)} - 1}{x_3(i)} \frac{dx_1(i)}{du(j)} - \frac{x_4(i) - x_1(i)}{x_3^2(i)} \frac{dx_3(i)}{du(j)} \right]_{1 \leq i \leq N_c} \\ \left[ \frac{dx_3(i)}{du(j)} \right]_{1 \leq i \leq N_c} \end{bmatrix}_{1 \leq j \leq N_c} \quad (34)$$

Although Eq. 29 requires that each  $\frac{dx(i)}{du}$ ,  $i \in [1, 2, \dots, N_s]$  is computed in sequence, Eqs. 31–34 are independent and can be calculated in parallel. Furthermore



**Fig. 1** Block diagram of the controller. With the exception of the shaded block, the other operations can be run in parallel. Inside the shaded block a cascaded connection is used to calculate  $\frac{dx(i)}{du}$  from  $\frac{dx(i-1)}{du}$

the rest of the neural network also allows parallel computations as can be seen in Fig. 1.

## 5 Results

The controller was tested using a model of the OH-58A helicopter with high energy rotor system (HERS). Although this helicopter does not represent a small rotorcraft, this model was chosen because it has been used extensively in the literature thus facilitating comparison with other developed controllers. The model parameters used are summarized in Table 1. For this paper a simple objective function was used:

$$L^* = 0.05 (\max(v_H, 5.5) - 0.5)^2 e^{1-1.25 \min(z, 1)} \quad (35)$$

This corresponds to an objective of lowering the sink rate to  $0.5 \text{ m s}^{-1}$  for the last 3 m of the descent.

### 5.1 Baseline Scenario

The baseline simulation was carried out modeling a descent from an initial altitude of 120 m using a prediction horizon of  $N_s = 10$  and a control horizon of  $N_c = 5$ . The simulation is performed at an update rate of 1kHz while the controller is run at 10Hz. The neural network parameters are  $E = 150$  and  $\gamma = 0.08$ .

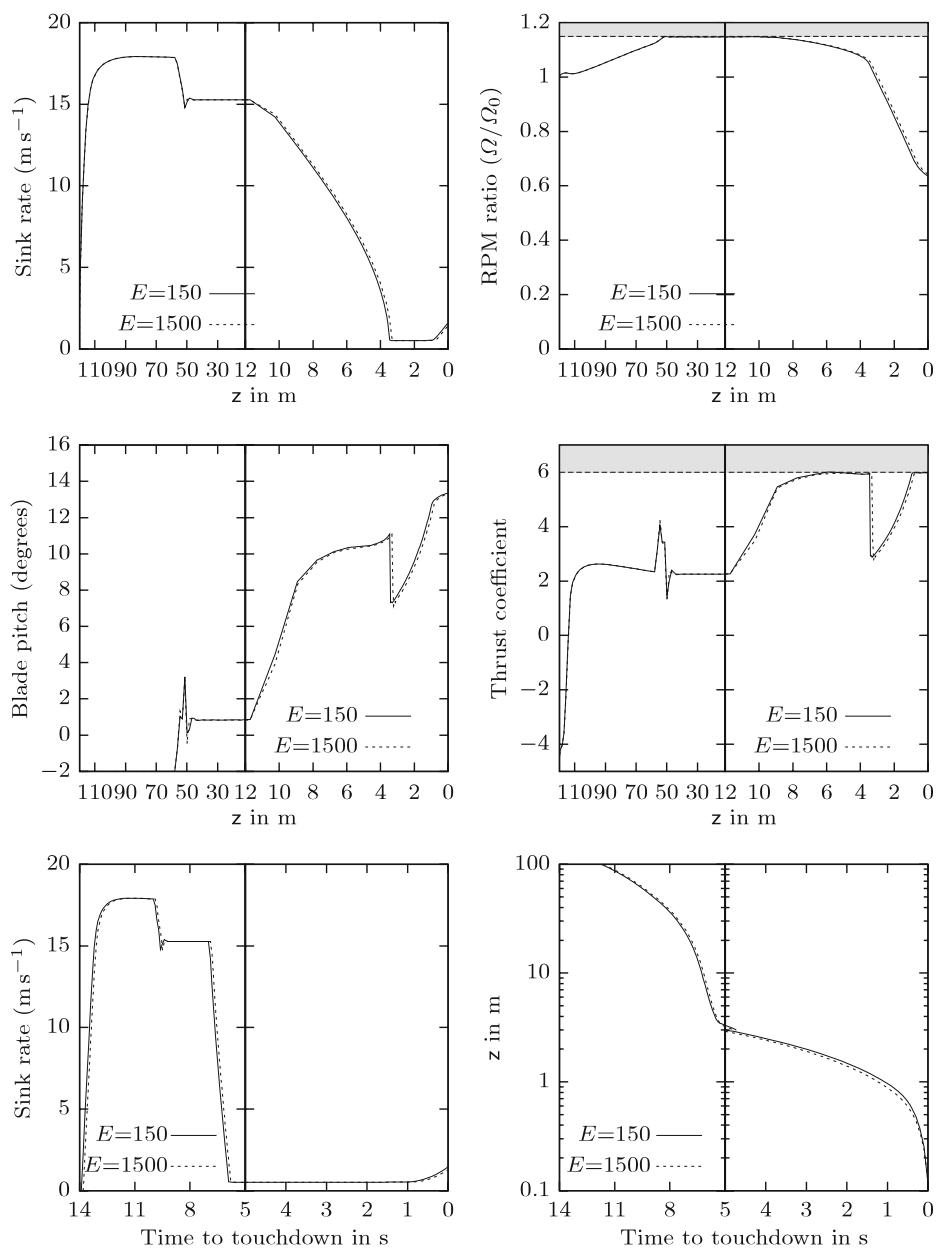
The results, presented in Fig. 2, show that the helicopter accomplished the stated objective, without violating the thrust coefficient and rpm constraints. The whole maneuver lasts for about 14 s, although about half of it corresponds to the last 4 m of the descent. The sudden jump in blade pitch at an altitude of about 60 m is due to the big drop required in the sink rate so that the  $\Omega$ -constraint is not violated. A smoother transition is also possible, but at the expense of either violating the constraint or requiring a longer prediction horizon.

It should also be noted that although a sink rate of  $0.5 \text{ m s}^{-1}$  is achieved, towards the end of the maneuver an increase in the sink rate is observed. This is due to the rapid loss of inertial energy in the rotor and the onset of stall in the blades that

**Table 1** Vertical autorotation model parameters for a modified OH-58A helicopter with high energy rotor system

Helicopter mass ( $M$ )	1,360 kg
Rotor moment of inertia ( $I_R$ )	911 kg m <sup>2</sup>
Solidity factor ( $\sigma$ )	0.048
Rotor disc radius ( $R$ )	5.37 m
Mean drag coefficient ( $C_d$ )	0.008,7
Lift coefficient ( $C_l$ )	5.73 rad <sup>-1</sup>
Equivalent unit drag coefficient area ( $f_e$ )	2.32 m <sup>2</sup>
Induced power correction factor ( $\kappa$ )	1.13
Nominal rotor speed ( $\Omega_0$ )	354 rpm or 37 rad s <sup>-1</sup>
Air density ( $\rho_\alpha$ )	1.225 kg m <sup>-3</sup>
Minimum main rotor pitch	-2°
Maximum main rotor pitch	16°
Maximum main rotor rpm	1.15Ω <sub>0</sub>

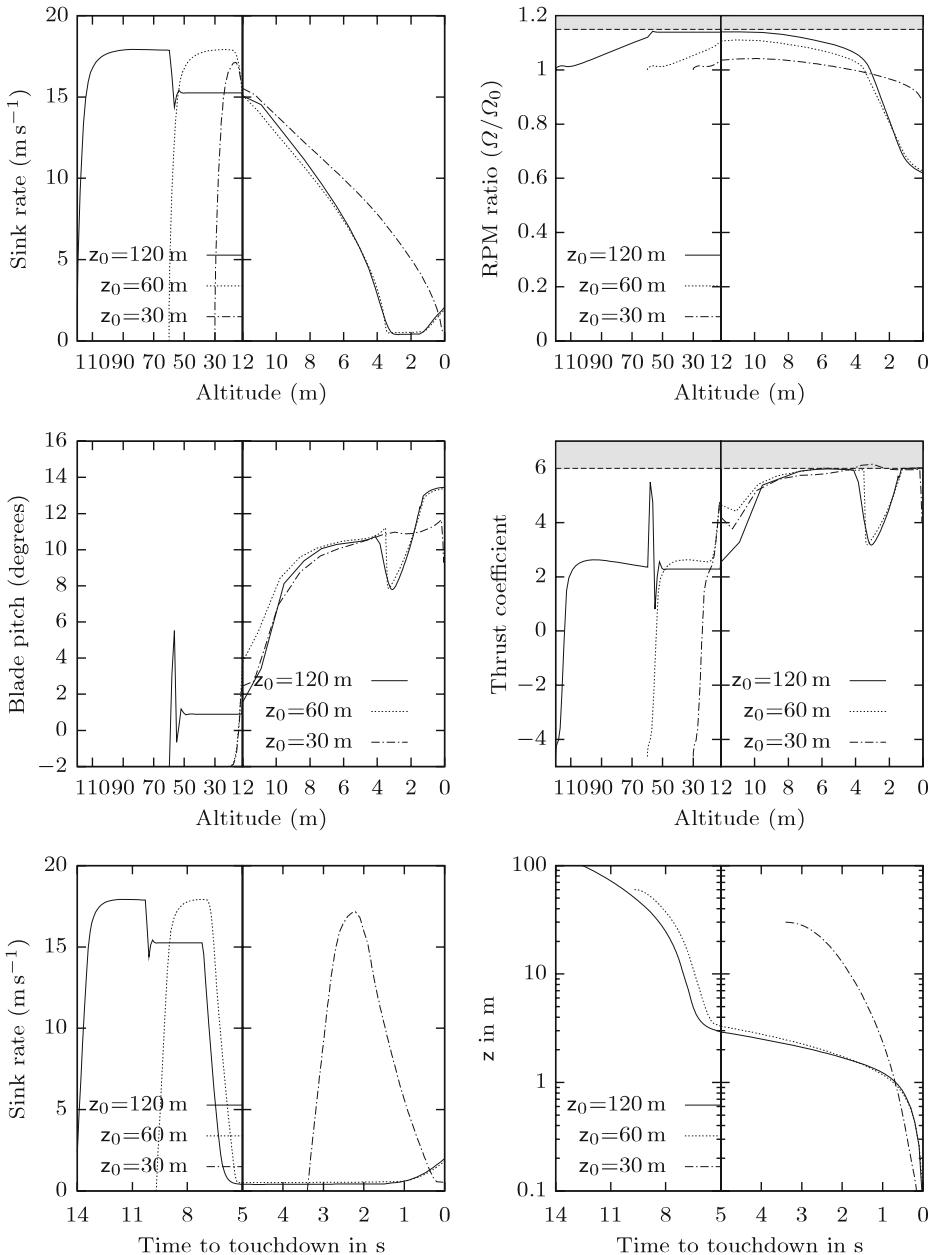
Source: [8, 13]



**Fig. 2** The sink rate, rotor rpm and control input of the OH-58A for a descent from an initial altitude of 120 m ( $N_s = 10$ ,  $N_c = 5$ ). The shaded regions represent the posed constraints. The scale on the right side of the graphs has been altered to show the last stage of the descent in higher detail

requires checking the rate of increase of the blade pitch. Nevertheless, and despite the increase, the velocity at touchdown remains within mechanical tolerances of the landing gear, since the latter is typically designed for sink rates up to  $3 m s^{-1}$ .

Figure 2 also presents the resulting trajectories for  $E = 1500$ . It is obvious that despite the tenfold increase in the time available to the neural network, the output is not significantly different and the results are comparable.



**Fig. 3** The effect of different initial altitudes on the performance of the controller

## 5.2 Initial Altitude

The initial altitude can influence the state of the helicopter at the final stage of the descent and specifically the rotational energy available in the rotor to reduce the sink rate. For initial altitudes exceeding 90 m the helicopter will have time to reach the limit of the allowable rotor rpm. As a result in each case it will reach the flare altitude with  $\Omega \simeq 1.15\Omega_0$  and  $v_H \simeq 15 \text{ m s}^{-1}$ .

Conversely in the occasions were the initial altitude is lower than 90 m, the kinetic energy stored in the rotor will be lower. The sink rate can be lower than  $15 \text{ m s}^{-1}$  if the helicopter failed at a very low altitude ( $< 10 \text{ m}$ ) or higher for intermediate altitudes. Figure 3 presents the trajectories for three simulations that feature an initial altitude of 120 m, 60 m and 30 m. In the last instance and although the output of the controller does not deviate significantly from that of the other simulations, the target sink rate is achieved only centimeters above the ground. This is because of the thrust coefficient constraint and the lower energy stored in the rotor. To improve the performance in this case, either the prediction horizon would need to be increased or the thrust coefficient constraint be relaxed.

## 5.3 Noise

To investigate the impact of sensor noise on controller performance, two simulations were carried out using different noise levels. The noise is assumed to be zero-mean, gaussian with varying standard deviation. The noise levels are summarized in Table 2.

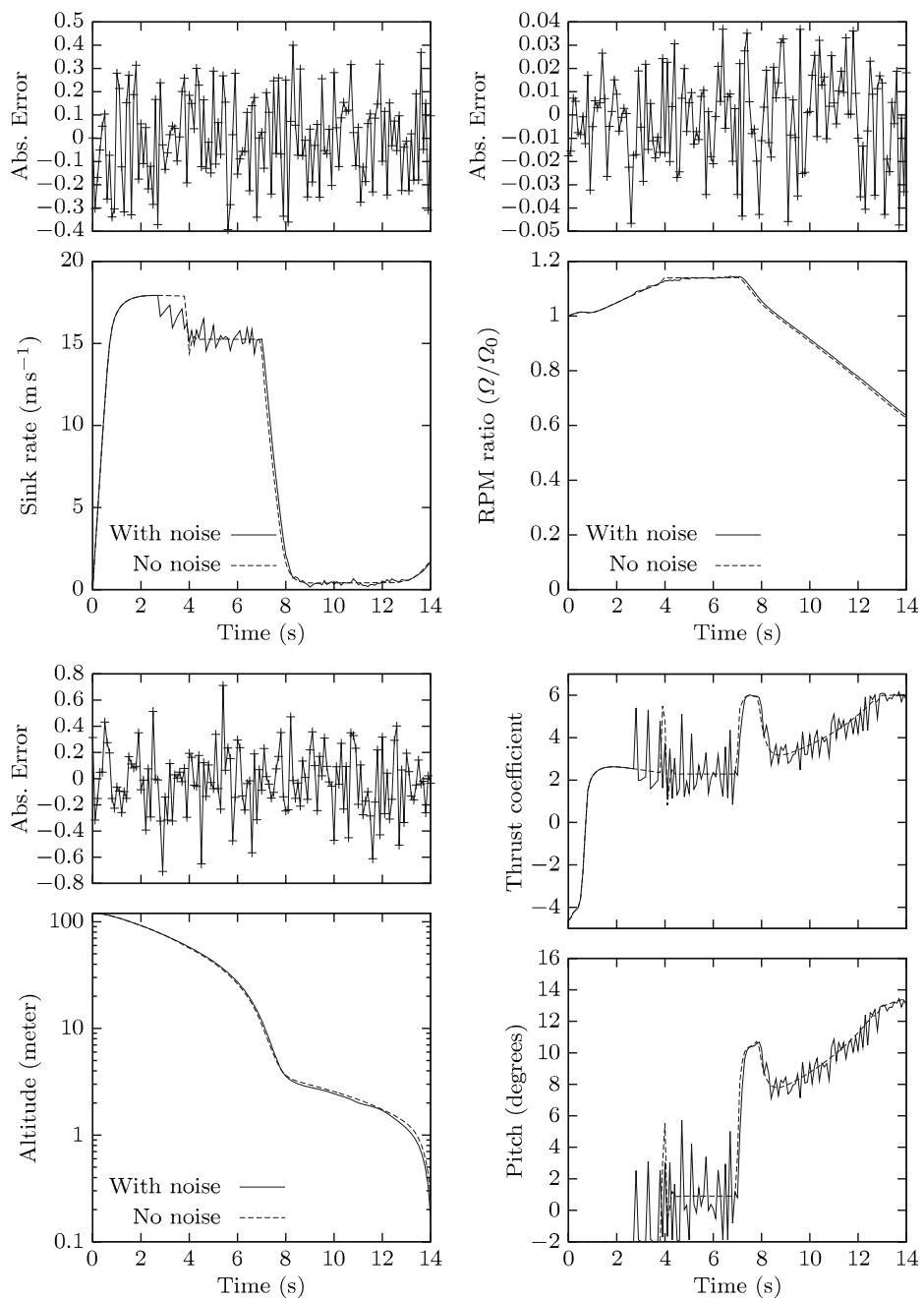
Figure 4 shows the simulation results for the first noise level scenario. The trajectory is not significantly changed and the only thing affected is the controller output during the  $\Omega$ -controlled descent. The latter occurs because of the design of the controller that tries to maintain constant rpm and becomes more pronounced as the noise in the rpm measurements increases as shown in Fig. 5. As the noise level increases the  $v_H$ -controlled region starts to get affected as well. This is because the controller is required to maintain a constant velocity in the face of noisy measurements. It should be noted that in the second case the simulation is terminated early. This is because according to the information available to the controller, the helicopter has reached the ground and the simulation terminated.

## 5.4 Execution Speed

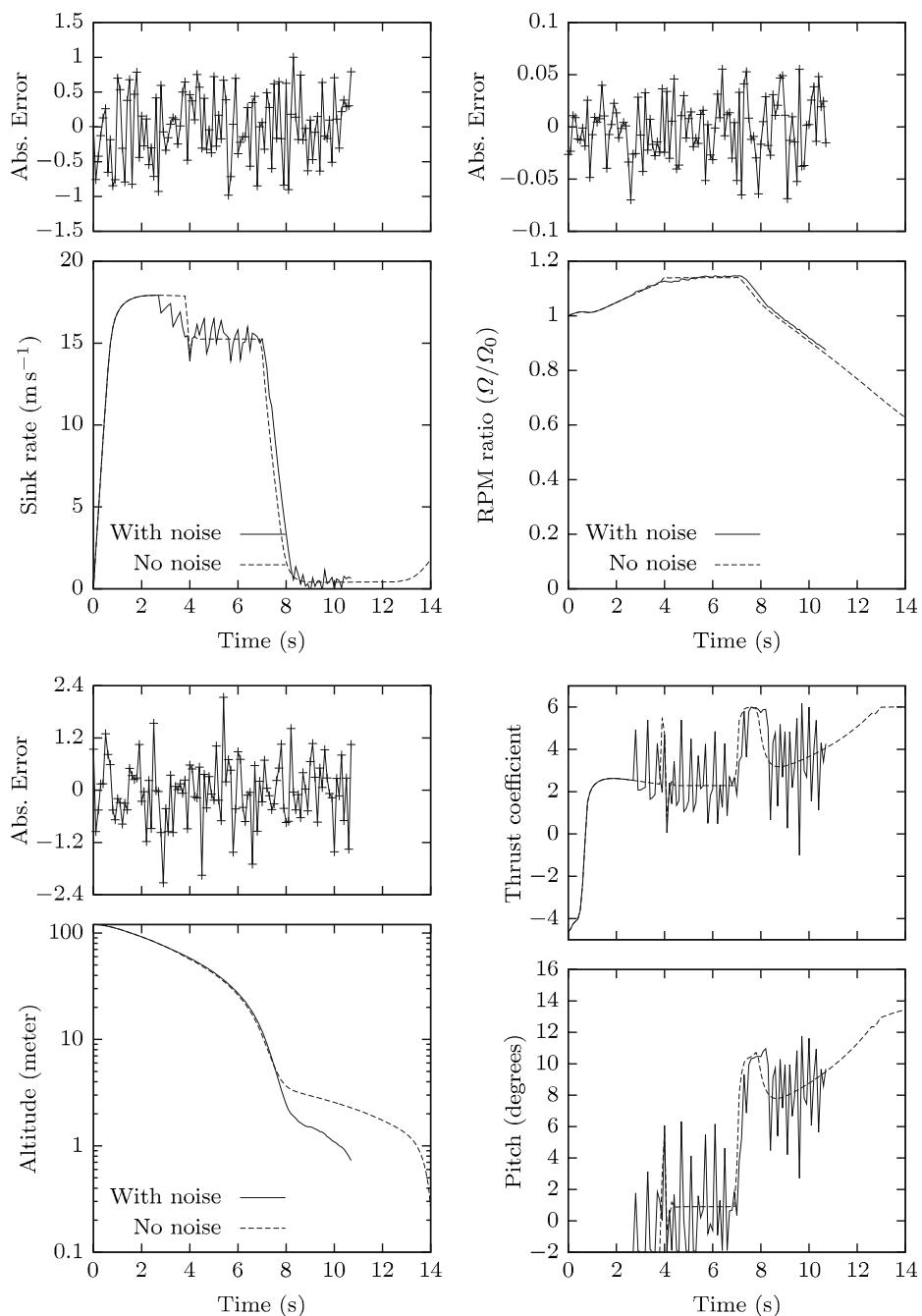
An additional parameter that is important in this problem is the execution speed of the controller. To determine this, the execution time of a single iteration was calculated for different prediction and control horizons. Since a single iteration is very fast, its execution time was estimated by measuring the total time required for 2,000 iterations. The tests were carried out using a single-core Athlon XP 3200+

**Table 2** Sensor noise levels

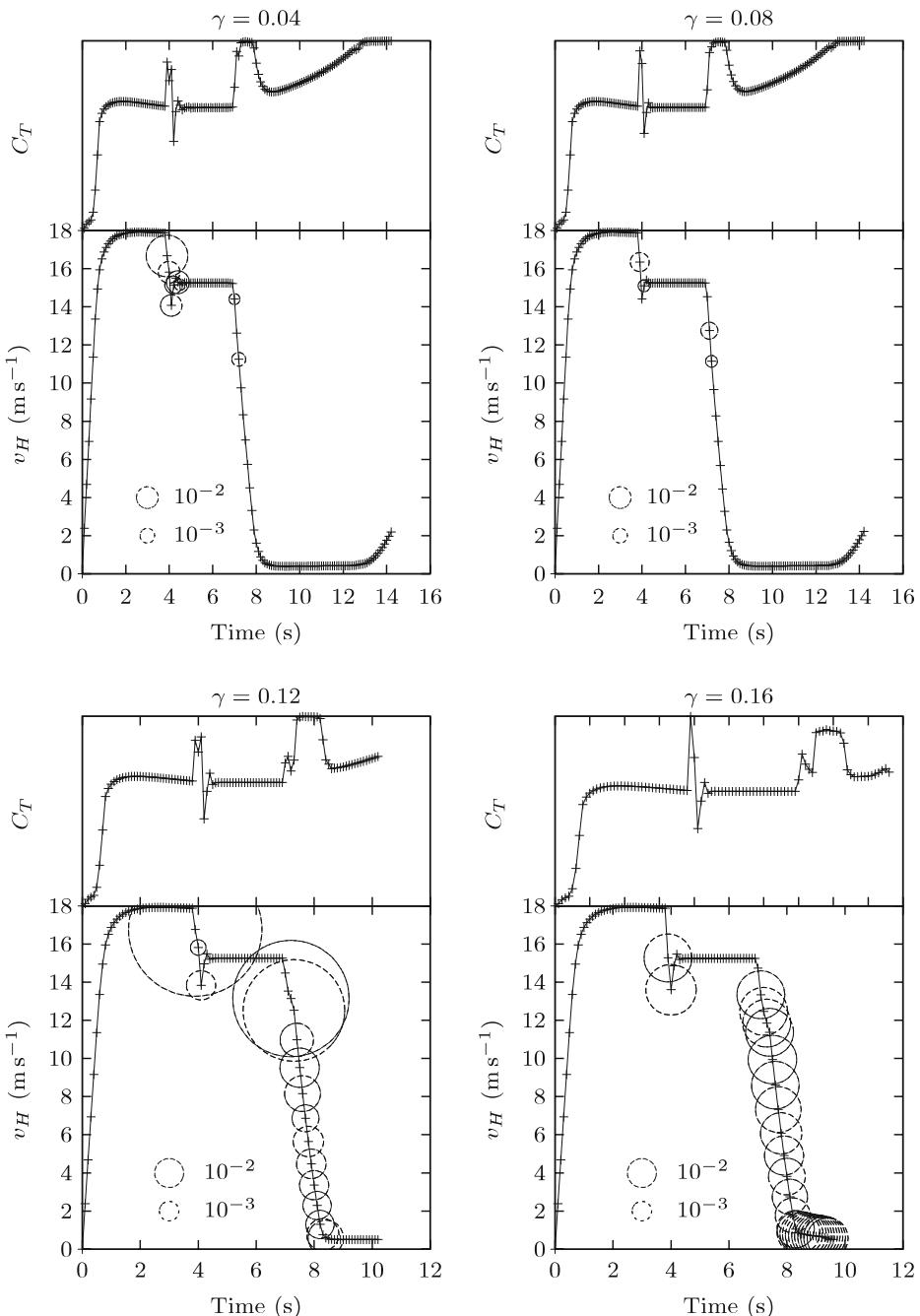
	Standard deviation	
	Level 1	Level 2
Sink rate ( $\text{m s}^{-1}$ )	0.2	0.5
Altitude (m)	0.25	0.75
Rotor speed (RPM)	7	10.5



**Fig. 4** The trajectory of the helicopter with and without noise under the second noise level. The top graphs present the absolute error of the sensor measurement as a function of time



**Fig. 5** The trajectory of the helicopter with and without noise under the third noise level. The top graphs present the absolute error of the sensor measurement as a function of time



**Fig. 6** The variance of the neural network output with respect to time and helicopter sink rate for four values of the learning rate parameter. Larger circles mean larger variance while variances smaller than  $10^{-4}$  are not shown. For comparison purposes the thrust coefficient at the corresponding time is also provided

(model of 2005) CPU, throttled to a frequency of 1 GHz running a 32-bit version of Debian Linux. No parallelization or off-line optimization was used to improve the execution time. The cost of the EKF was also calculated under the same conditions and was found to be approximately 128  $\mu$ s.

For  $N_s = 10$  and  $N_c = 5$  a controller iteration required 0.88 ms whereas for  $N_s = 12$  and  $N_c = 6$  this time increased to 1.23 ms. If an update rate of 10 Hz is chosen, under the aforementioned conditions the maximum epoch length is 113 and 81 respectively. For higher prediction and control horizon the number of possible iterations drops considerably, down to 30 for  $N_s = 20$  and  $N_c = 10$ .

## 5.5 Learning Rate and Convergence

To investigate the effect of the learning rate parameter as well as the convergence characteristics of the neural network, four simulations were carried out for different values of  $\gamma$ . Higher values of the latter parameter are typically used to improve the convergence speed. On the other hand, such values can lead to over-corrections and oscillations. This is exacerbated by constraint enforcement and results in producing the opposite of the desired effect. The influence of the effect of the controller design parameter  $\gamma$  was evaluated using the variance in the neural network output during the last 20% of the repetitions of each cycle. If the neural network has converged, low output variance is expected. The results summarized in Fig. 6 show that high variance is exhibited mainly in regions where the helicopter transitions from one mode to another. This is expected because the cost and constraint derivatives take their higher values there. Additionally for learning rates 0.12 and above, variance is exhibited first in the region of deceleration and then in the entire velocity-controlled region.

## 6 Conclusions and Future Work

Current proposed automated autorotation methods have one or more significant drawbacks that don't allow them to be incorporated as they are in current and future aircraft:

- The trajectory is not calculated on-line because the calculations cannot be carried out in real-time. This is significant because any discrepancies between the model and the actual aircraft as well as any external disturbances can lead to accumulating error. This error can be significant by the time the aircraft approaches touchdown and may lead to a catastrophic accident.
- The objective of the autorotation maneuver is typically chosen to be zero vertical and horizontal velocity at zero height. Nevertheless, especially in the case of unmanned helicopters the foremost objective should be to minimize human injuries and fatalities.
- Autonomous autorotation is based on training using pre-recorded attempts by an expert. This can be problematic since the limitations of a human pilot are incorporated into the design and the performance will be as good as a human pilot. Furthermore it does not allow for different objectives and large deviations from the conditions under which the experiments were recorded.

- The controller is designed as a black box, trained through repeated, simulated trial and error. In this approach the accuracy of the simulation model used is very important. Furthermore the controller needs to be repeatedly trained under all possible conditions.

On the other hand the controller presented in this paper is capable of real-time, on-line trajectory optimization using different objective functions without a requirement for training beforehand. Due to the characteristics of the NMPC approach, it is robust to sensor errors and the introduction of a recurrent neural network simplifies the non-linear optimization and significantly improves the speed with which the optimal control sequence is obtained. The convergence speed may be further improved utilizing specialized hardware, that allows parallel computations.

Future work in this area will entail further testing of the controller with different helicopter models, different scenarios (e.g. wind) as well as testing against commercial flight simulators. Gain scheduling or alternate objective functions need to be investigated to determine if it possible to overcome the convergence problems encountered in the regions where the helicopter rapidly changes state. Furthermore better tuning of the objective function and NMPC and neural network parameters is planned, along with an investigation of their impact on the controller performance in terms of both accuracy and speed. Long-term goals include an investigation of the convergence characteristics of the neural network; whether convergence can be guaranteed and under what conditions.

In the future, it is envisioned that this controller will be a part of larger emergency system that will be capable of handling a multitude of failures from detection to resolution. Such a system would undoubtedly improve the safety performance of unmanned helicopters in general and pave the way for further integration of such systems into the national airspace system.

**Acknowledgement** This research has been partially supported by NSF Grant, IIP-0856311 (DU Grant number 36563).

## References

1. Abbeel, P., Coates, A., Hunter, T., Ng, A.Y.: Autonomous autorotation of an RC helicopter. In: Proc. 11th International Symposium on Experimental Robotics (2008)
2. Anand, S.: Domestic use of unmanned aircraft systems: evaluation of policy constraints and the role of industry consensus standards. *Journal of Engineering and Public Policy* **11** (2007)
3. Aponso, B., Bachelder, E., Lee, D.: Automated autorotation for unmanned rotorcraft recovery. Presented at AHS international specialists' meeting on unmanned rotorcraft (2005)
4. Carlson, R.: Helicopter performance—transportation's latest chromosome: the 31st annual Alexander A. Nikolsky lecture. *J. Am. Helicopter Soc.* **47**(1) (2002)
5. Clothier, R., Walker, R.: Determination and evaluation of UAV safety objectives. In: Proc. 21st International Unmanned Air Vehicle Systems Conference, pp. 18.1–18.16 (2006)
6. Clothier, R., Walker, R., Fulton, N., Campbell, D.: A casualty risk analysis for unmanned aerial system (UAS) operations over inhabited areas. In: Proc. 12th Australian International Aerospace Congress and 2nd Australasian Unmanned Air Vehicles Conference (2007)
7. Dalamagkidis, K., Valavanis, K., Piegl, L.: On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations, Intelligent Systems, Control and Automation: Science and Engineering, vol. 36. Springer, New York (2009)
8. Dooley, L.W., Yearly, R.D.: Flight test evaluation of the high inertia rotor system. Final report for period 21 September 1976–February 1979 USARTL-TR-79-9, Bell Helicopter Textron (1979)

9. Haddon, D.R., Whittaker, C.J.: UK-CAA policy for light UAV systems. UK Civil Aviation Authority (2004)
10. Hayhurst, K.J., Maddalon, J.M., Miner, P.S., Dewalt, M.P., McCormick, G.F.: Unmanned aircraft hazards and their implications for regulation. In: Proc. 25th IEEE/AIAA Digital Avionics Systems Conference, pp. 1–12 (2006)
11. Hazawa, K., Shin, J., Fujiwara, D., Igarashi, K., Fernando, D., Nonami, K.: Autonomous autorotation landing of small unmanned helicopter. Transactions of the Japan Society of Mechanical Engineers C **70**(698), 2862–2869 (2004)
12. Johnson, W.: Helicopter optimal descent and landing after power loss. NASA TM 73244, Ames Research Center, National Aeronautics and Space Administration (1977)
13. Lee, A.Y.N.: Optimal landing of a helicopter in autorotation. Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University (1985)
14. Lee, A.Y.N.: Optimal autorotational descent of a helicopter with control and state inequality constraints. J. Guid. Control Dyn. **13**(5), 922–924 (1990)
15. Leishman, J.G.: Principles of Helicopter Aerodynamics, 2nd edn. Cambridge Aerospace Series. Cambridge University Press, Cambridge (2006)
16. Weibel, R.E.: Safety considerations for operation of different classes of unmanned aerial vehicles in the national airspace system. Master's thesis, Massachusetts Institute of Technology (2005)
17. Xia, Y., Wang, J.: A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints. IEEE Trans. Circuits Syst. I **51**(7), 1385–1394 (2004). doi:10.1109/TCSI.2004.830694
18. Xia, Y., Wang, J.: A recurrent neural network for solving nonlinear convex programs subject to linear constraints. IEEE Trans. Neural Netw. **16**(2), 379–386 (2005). doi:10.1109/TNN.2004.841779
19. Xia, Y., Leung, H., Wang, J.: A projection neural network and its application to constrained optimization problems. IEEE Trans. Circuits Syst. I **49**(4), 447–458 (2002). doi:10.1109/81.995659
20. Xia, Y., Feng, G., Kamel, M.: Development and analysis of a neural dynamical approach to nonlinear programming problems. IEEE Trans. Automat. Contr. **52**(11), 2154–2159 (2007). doi:10.1109/TAC.2007.908342

# Multimodal Interface Technologies for UAV Ground Control Stations

## A Comparative Analysis

I. Maza · F. Caballero · R. Molina · N. Peña · A. Ollero

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 13 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper examines different technologies that can be applied in the design and development of a ground control station for Unmanned Aerial Vehicles (UAVs) equipped with multimodal interfaces. Multimodal technologies employ multiple sensory channels/modalities for information transmission as well as for system control. Examples of these technologies could be haptic feedback, head tracking, auditory information (3D audio), voice control, tactile displays, etc. The applicability and benefits of those technologies is analyzed for a task consisting in the acknowledgement of alerts in an UAV ground control station composed by three

---

This work is partially supported by the CONET Network of Excellence (ICT-224053) funded by the European Commission under FP7, the ROBAIR Project funded by the Spanish Research and Development Program (DPI2008-03847) and the ATLANTIDA Project leaded by *Boeing Research & Technology Europe*.

I. Maza (✉) · F. Caballero · A. Ollero  
Robotics, Vision and Control Group, University of Seville,  
Avd. de los Descubrimientos s/n, 41092, Sevilla, Spain  
e-mail: imaza@cartuja.us.es

F. Caballero  
e-mail: caba@cartuja.us.es

A. Ollero  
e-mail: aollero@cartuja.us.es

R. Molina · N. Peña  
Boeing Research & Technology Europe, Canada Real de las Merinas,  
1-3, Bldg 4, 28042 Madrid, Spain

R. Molina  
e-mail: roberto.molina@boeing.com

N. Peña  
e-mail: nicolas.penaortiz@boeing.com

A. Ollero  
Center for Advanced Aerospace Technology (CATEC), Seville, Spain  
e-mail: aollero@catec.aero

screens and managed by a single operator. For this purpose, several experiments were conducted with a group of individuals using different combinations of modal conditions (visual, aural and tactile).

**Keywords** Unmanned Aerial Vehicles · Multimodal interfaces

## 1 Introduction

It is known that multimodal display techniques may improve operator performance in Ground Control Stations (GCS) for Unmanned Aerial Vehicles (UAVs). Presenting information through two or more sensory channels has the dual benefit of addressing high information loads as well as offering the ability to present information to the operator within a variety of environmental constraints. A critical issue with multimodal interfaces is the inherent complexity in the design of systems integrating different display modalities and user input methods. The capability of each sensory channel should be taken into account along with the physical capabilities of the display and the software methods by which the data are rendered for the operator. Moreover, the relationship between different modalities and the domination of some modalities over others should be considered.

Using multimodal technologies begins to be usual in current GCSs [7, 13, 14], involving several modalities such as positional sound, speech recognition, text-to-speech synthesis or head-tracking. The level of interaction between the operator and the GCS increases with the number of information channels, but these channels should be properly arranged in order to avoid overloading the operator.

In [19] some of the emerging input modalities for human-computer interaction (HCI) are presented and the fundamental issues in integrating them at various levels—from early “signal” level to intermediate “feature” level to late “decision” level are discussed. The different computational approaches that may be applied at the different levels of modality integration are presented, along with a briefly review of several demonstrated multimodal HCI systems and applications.

On the other hand, the intermodal integration can contribute to generate the illusion of presence in virtual environments if the multimodal perceptual cues are integrated into a coherent experience of virtual objects and spaces [1]. Moreover, that coherent integration can create cross-modal sensory illusions that could be exploited to improve user experiences with multimodal interfaces, specifically by supporting limited sensory displays (such as haptic displays) with appropriate synesthetic stimulation to other sensory modalities (such as visual and auditory analogs of haptic forces).

Regarding the applications in the UAVs field, [10] provides a survey on relevant aspects such as the perceptual and cognitive issues related to the interface of the UAV operator, including the application of multimodal technologies to compensate for the dearth of sensory information available.

The paper is structured as follows. In the next section, different technologies that can be applied in the design and development of a GCS for UAVs equipped with a multimodal interface are summarized in Section 2. Then, Section 3 describes a multimodal testbed developed by the authors along with a set of tests to measure the benefits under different modal conditions. Section 4 provides the results obtained in

several experiments performed with a group of individuals and analyzes the impact of the different modalities. Finally, the conclusions and future work section closes the paper.

## 2 Interactions between Operator and GCS

In the design of a GCS, two information flows are usually considered: *from GCS to operator*, presenting information about the UAV, the environment and the status of the mission, and *from operator to GCS* in the form of commands and actuations which are treated as inputs by the GCS software.

But there is a third flow of information which is not usually addressed in the design of the UAV's GCS; the information about the *operator's state* that can be gathered by sensors and processed by the GCS software. This channel could allow to have an adaptive GCS software, which can change the modality and format of the information depending on the state of the operator (tired, bored, etc). Furthermore, the information about the operator can be also used to evaluate and improve the interface. For instance, it is possible to register which screens are mainly used by the operator during a certain type of mission.

Next subsections are devoted to each of these information flows, summarizing several methods and devices which are usually applied.

### 2.1 Information Flow from GCS to Operator

Classical modalities in GCS to operator communications are visual information (mainly in monitors) and sound alerts. Last decades researchers have dedicated a significant effort to define the characteristics of such communications, taking into account the operator's capabilities and maximizing the information showed to the operator. Thus, effective data visualization and distribution is discussed in [26] and [22], where different models to measure the effectiveness of the visualization system are presented. Other researchers include the color, shape or size of the displays in their analysis.

Sound alerts have also been deeply studied, mainly applied to control stations in general. Intensity, frequency or loudness are some of the parameters taken into account to create comfortable and effective sound alarms. References [17] and [16] are good examples of sound alarm studies focused on civil aircrafts.

However, higher computational capabilities and the evolution of the communication systems raised new devices and techniques able to provide more complex information. The next paragraphs describe some of these new approaches.

#### 2.1.1 3D Audio

Concerning the aural modality, the 3D audio can improve the Situational Awareness (SA) of the operator. Three dimensional audio is based on a group of sound effects that attempt to widen the stereo image produced by two loudspeakers or stereo headphones, or to create the illusion of sound sources placed anywhere in a three dimensional space, including behind, above or below the listener. Taking into account the portability usual requirement for the ground stations, the use of a headset is usually preferred for the operator, instead of a set of speakers around him.

Thus, the objective of the 3D audio interface in a GCS is to provide multiple located sources of sound for the operator (the listener) to improve his SA while performing a given mission. In this way, the operator is able to recognize the presence of an alarm and also the origin of such alarm. This functionality is provided for example by a library called OpenAL [4] (Open Audio Library), which is a free software cross-platform 3D audio Application Programming Interface (API) designed for efficient rendering of multichannel three dimensional positional audio, and distributed under the LGPL license. This library has been used in our system implementation, which is described in Section 3.

### 2.1.2 Speech Synthesis

Considering also the audio channel, the speech synthesis technology has been also included in the system used in the experiments presented in this paper. Speech synthesis, also known as text-to-speech, is the artificial production of human speech. It can be implemented in software or hardware and basically can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units.

The quality of a speech synthesizer is judged by its similarity to the human voice, and by its ability to be understood. An intelligible text-to-speech program allows operators to listen complex messages, normally related with the state of commands, events or tasks currently carried out in the GCS. An example of these applications in the UAVs context can be found in the WITAS project [21].

A good example of free speech synthesis software is the Festival library [24]. It is a general multi-lingual speech synthesis system originally developed at Centre for Speech Technology Research (CSTR) at the University of Edinburgh. It offers a full text to speech system with various APIs, as well as an environment for development and research of speech synthesis techniques. It offers a general framework for building speech synthesis systems. As a whole it offers full text to speech through a number of APIs: from shell level, through a Scheme command interpreter, as a C++ library, from Java, and an Emacs interface. In the tests presented in this paper, Festival has been used as a C++ library and integrated into our multimodal software application.

### 2.1.3 Haptic Devices

Haptic technologies interface to the user via the sense of touch by applying forces, vibrations and/or motions to the operator. This mechanical stimulation can be applied to assist in the “creation” of virtual objects (objects existing only in a computer simulation), for control of such virtual objects, and to enhance the remote control of machines and devices (teleoperators).

In the particular case of GCSs, haptic devices add a new communication channel to the operator. The vibration of the device can be used as a stand alone alarm mechanism, or in combination with other sensory channels can increase the information provided to the user. For instance, if the activation of the haptic device is added to a currently played sound alarm, the operator will consider that the priority/criticality of such alarm has been increased.

## 2.2 Information Flow from Operator to GCS

Normally, the operator provides information to the GCS through mouse, touchpad or keyboard. However, other channels can be used to provide information to the software application running in the GCS.

### 2.2.1 Touch Screens

A touchscreen is a display which can detect the presence and location of a touch within the display area. The term generally refers to touch or contact to the display of the device by a finger or hand. The touchscreen has two main attributes. First, it enables the operator to interact with what is displayed directly on the screen, where it is displayed. Secondly, it lets the operator do so without requiring any intermediate device. Thus, touchscreens allows intuitive interactions between the operator and the GCS application.

Nevertheless, it is important to remark that touchscreen technology is usually poor in resolution if the operator uses his finger, i.e. the minimal size of the objects required to guarantee a proper interaction with the user must be bigger compared to a mouse or a touchpad. This is one of the main constraints to be considered in the design of the graphical interfaces to be used with touchscreens.

### 2.2.2 Automatic Speech Recognition

Speech recognition (also known as automatic speech recognition or computer speech recognition) converts spoken words to machine-readable input (for example, to key presses, using the binary code for a string of character codes). Speech recognition provides an easy and very effective way to command tasks to GCSs [7].

## 2.3 Operator's State

The operator's state is the third information flow mentioned above. It can be defined as the set of physiological parameters that allows to estimate the state of a human operator: heartbeat, temperature, transpiration, position, orientation, etc. All this information can be used by adaptive systems to improve the operator environment or to reduce the stress/workload of the operator.

There are plenty of studies examining how psychophysiological variables (e.g., electro-encephalogram, eye activity, heart rate and variability) change as a function of task workload (see [3, 18] or [25]), and how these measures might be used to monitor human operators for task overload [15] and/or used to trigger automated processes.

Next sections details some of the current technologies used in the operator's state estimation.

### 2.3.1 2DoF Head Tracking

As the operator's head concentrates his main sensorial capabilities, a very important tool to acquire the operator's state is the head position and orientation tracking. 2DoF head tracking applications and products are easy to find. Most of these products are based on image processing and marks/spots placed in the users head (or a hat). They also provide the two angle information used to move the mouse

from left/right and up/down. The following professional solutions can be highlighted: *Tracker Pro* [8], *Headmouse Extreme* [2] or *SmartNav 4 AT* [11].

A practical application of this technology could be a GCS software that provides the critical alerts on the screen which is being used by the operator when they occur. It can be also applied to evaluate the interaction between the human and the GCS during each mode of operation in terms of which information/screen is more relevant for the operator, etc.

### 2.3.2 2DoF Eye Tracking

If the GCS is composed by several screens it could be also necessary in many cases to track the head and the eye position in order to determine which screen is being used by the operator. However, products related with 2DoF eye tracking are scarce in the market. All of them are based on computer vision systems that analyzes the images gathered by a camera (normally mounted on the computer monitor). *EyeTech TM3* [23] is a good example.

### 2.3.3 6DoF Head Tracking

6DoF head tracking moves one step forward and allows estimating the complete position and orientation of the user's head in real time. Most of the existing methods make use of cameras and visual/IR patterns mounted on the operator's head.

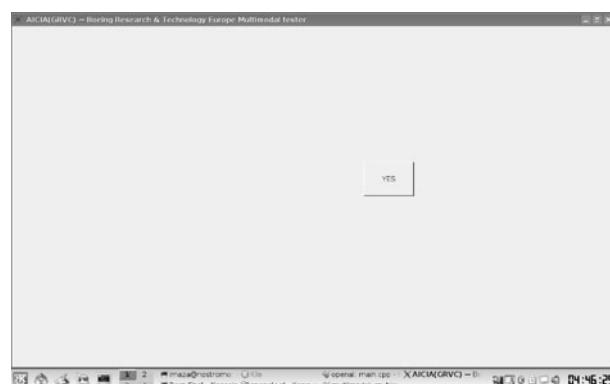
*TrackIr* [12], *Cachya* [20] and *FreeTrack* [5] represent the main options in the market. They use a 3D pattern visible in the infrared or visual band to estimate the position and orientation of the operator's head.

### 2.3.4 Body Motion Sensors

In order to register the behavior of the operator during a mission it could be also convenient to attach small sensors to his body to log the motion data. For example, it is possible to embed a wireless 3-axis accelerometer in each arm of the operator. The data registered can help to determine his current state (bored, tired, etc) and useful information as for example which arm is more used in each mode of operation and GCS configuration.

**Fig. 1** Multimodal technologies based system developed



**Fig. 2** Multimodal software application graphical interface

### 3 Multimodal Technologies Based System Developed

The applicability and benefits of multimodal technologies has been analyzed for a simple task consisting in the acknowledgement of alerts in an UAV ground control station composed by three screens and managed by a single operator. For this purpose, several experiments were conducted with a group of individuals using different combinations of modal conditions (visual, aural and tactile).

A software application integrating the different modalities has been developed and used in the tests. The experimental results are shown in this section whereas the corresponding analysis and conclusions are detailed in Section 4. This information can be used as a starting point in the design of the multimodal ground control station for UAVs.

#### 3.1 System Description

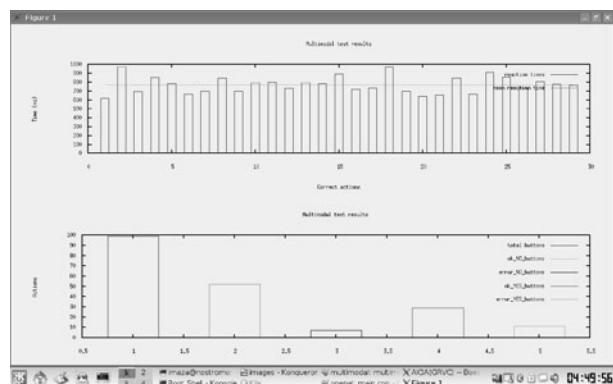
Figure 1 presents the system used to perform the multimodal experiments. This setup emulates a GCS for UAVs in which the operator can interact with the station through the following devices: three touch screens, three wireless haptic devices attached to the right hand, left hand and also on the chest of the operator, one optical mouse, one headset and stereo speakers. In addition, modules for Speech Synthesis and 3D Sound are included into the software application.

The application has been developed under Linux and makes use of different modalities to show the information to the operator. The graphical interface is composed by a single window (see Fig. 2) in which several buttons labeled as “Yes” or “No” appear in random positions. Only one button is present on the screen at any time and each button is displayed until it is pressed or until a programmable timeout

**Table 1** Operator right and wrong actions depending on the type of button which appears in the interface

Button	Right action	Wrong action
“Yes”	Press before timeout expires	Do not press before timeout expires
“No”	Do not press	Press

**Fig. 3** Graphical interface with the results of the test



( $T_{\text{yes}}$  or  $T_{\text{no}}$ ) expires. The duration of the experiment and the size of the buttons is also programmable.

The mission for the operator is quite simple: press **only** buttons labeled as “Yes” as soon as possible. Then, the right and wrong actions when each button appears on the screen are summarized in Table 1. For some values of the parameters, some wrong actions do not exist, i.e. if  $T_{\text{yes}} \rightarrow \infty$  there is no possible wrong action for the “Yes” buttons.

Both type of buttons have the same grey color, which is also the same color used in the background of the window. Therefore, when a button appears in the perimeter of the field of view, it is hard to realize for the user that it is there. This bad feature has been intentionally left in the application to emphasize the benefits of other modalities different from the visual one.

Once a test has finished, several performance parameters are computed and showed automatically on the screen. Figure 3 is an example of the interface with the results of a given experiment. In the top subfigure, the reaction time of the operator for each right action (corresponding to “Yes” buttons pressed before  $T_{\text{yes}}$ ) is shown in milliseconds. The mean reaction time ( $\bar{T}_{\text{yes}}$ ) is also represented with an horizontal line. In the subfigure below, the total number of buttons ( $n$ ), and the number of right and wrong actions for each button ( $n_{\text{right\_yes}}$ ,  $n_{\text{right\_no}}$ ,  $n_{\text{wrong\_yes}}$  and  $n_{\text{wrong\_no}}$ ) are represented with bars. Table 2 shows a summary of the values presented in Fig. 3.

The system developed allows integrating visual, aural and tactile modalities into the GCS. Their integration in the developed software have been carried out as follows:

- Speech synthesis: Once each button appears on the screen, its label is told to the operator.
- 3D audio: Depending on the location of the button on the window (left, right or middle), the source of audio corresponding to its label is generated on the left, on the right or in front of the operator respectively.

**Table 2** Summary of the values represented in Fig. 3

$T$ (sec)	$\bar{T}_{\text{yes}}$ (ms)	$n$	$n_{\text{right\_yes}}$	$n_{\text{right\_no}}$	$n_{\text{wrong\_yes}}$	$n_{\text{wrong\_no}}$
90	773.28	99	29	52	11	7

**Table 3** Summary of the tests designed along with the identifiers that will be used later to make reference to them

Experiment nr.	Description	Identifier
#1	Mouse interface only	Mouse
#2	Touch screen interface only	TS
#3	Touch screen and speech synthesis	TS+speakers
#4	Touch screen and 3D audio	TS+3D
#5	Touch screen and tactile interfaces	TS+vibrator
#6	Touch screen, 3D audio and tactile interfaces	TS+3D+vibrator
#7	Touch screen interface test repetition	TS2

- **Vibrator:** The wireless vibrator is activated every time a “Yes” button appears on the screen. Moreover, depending on the location of the button on the window (left, right or middle), the device on the left, on the right or in the middle vibrates respectively.

### 3.2 Tests Performed and Results

Prior to the different modalities tests, several sizes for the rectangular buttons are used in many tests with the touch screens. The goal is to determine the minimum size for the buttons which can “guarantee” a correct operation with the application. This minimum size is estimated to be approximately  $2.8 \times 2.6$  cm.

The tests described in the next subsections have been performed using the multimodal software previously presented. A short video with a summary of the experiments can be found at [9]. The values selected for the parameters of the software application have been the following:

- Full duration of each test:  $T = 8$  min.
- Size of the buttons:  $3.0 \times 2.8$  cm for the central screen and  $2.8 \times 2.6$  cm for the left and right screens.
- Timeout period of the buttons:  $T_{\text{yes}} \rightarrow \infty$  and  $T_{\text{no}} = 1.6$  sec respectively.

**Fig. 4** In the experiment #1 the operator is only allowed to use the mouse interface



**Table 4** Summary of the results for the experiment #1

Individual	<i>n</i>	<i>n</i> <sub>right_yes</sub>	<i>n</i> <sub>right_no</sub>	<i>n</i> <sub>wrong_no</sub>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
#1	334	166	168	0	1288.2	375.3
#2	316	165	151	0	1462.3	537.7
#3	325	162	163	0	1377.2	349.0
#4	341	172	169	0	1239.7	373.1
#5	358	178	179	1	1098.7	322.6
#6	368	195	173	0	1066.3	256.3
#7	350	170	180	0	1151.9	303.7
#8	342	172	170	0	1230.7	382.6
#9	357	171	186	0	1093.6	300.2

On the other hand, the tests have been done by nine people with ages between 20 and 30 years old (3 women and 6 men), registering their performance and opinions.

Table 3 shows a summary of the seven tests designed for the multimodal station. Each of those tests is detailed in the next subsections, including the results obtained by the different individuals.

### 3.2.1 Experiment #1: Mouse Interface

In this test, the operator can only use the mouse to press the “Yes” buttons appearing on the screens (see Fig. 4). His reaction time and the number of right/wrong actions are measured.

The results obtained by each individual are detailed in Table 4. It should be pointed out that all of them were used to work with the mouse.

### 3.2.2 Experiment #2: Touch Screen Interface

This test is like the previous one, but using the touch screen interface. It allows to compare both input technologies in order to evaluate which one is better suited for the station considered. The more efficient input method (mouse or touch screen) will be used in the following experiments.

Table 5 shows results better than those obtained with the mouse interface. In order to quantify the benefit of the touch screens, the percentage of reduction in the mean

**Table 5** Summary of the results for the experiment #2

Individual	<i>n</i>	<i>n</i> <sub>right_yes</sub>	<i>n</i> <sub>right_no</sub>	<i>n</i> <sub>wrong_no</sub>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
#1	343	162	180	1	1210.2	442.2
#2	338	174	164	0	1271.2	339.9
#3	348	167	181	0	1171.9	279.9
#4	359	185	174	0	1110.7	305.6
#5	356	168	188	0	1097.9	251.9
#6	362	190	172	0	1105.4	290.7
#7	369	185	184	0	1032.7	242.7
#8	371	189	182	0	1021.9	286.1
#9	371	195	175	1	1042.5	265.8

**Table 6** Summary of the results for the experiment #3

Individual	<i>n</i>	<i>n<sub>right_yes</sub></i>	<i>n<sub>right_no</sub></i>	<i>n<sub>wrong_no</sub></i>	$\overline{T_{yes}}$ (ms)	$\sigma$ (ms)
#1	364	182	182	0	1065.5	373.5
#2	348	165	183	0	1160.5	268.4
#3	363	179	183	1	1068.9	254.1
#4	370	184	186	0	1030.8	241.7
#5	371	182	189	0	1001.2	232.2
#6	372	181	191	0	991.0	194.9
#7	389	200	188	1	921.7	206.1
#8	393	216	176	1	943.5	277.6
#9	381	202	179	0	981.2	266.9

reaction time ( $\overline{\Delta T_{yes}}$ ) and also in the standard deviation of the reaction times ( $\overline{\Delta\sigma}$ ) have been computed for the whole population:

$$\overline{\Delta T_{yes}} = +9.33\%, \quad \overline{\Delta\sigma} = +19.73\% \quad (1)$$

Therefore, the touch screen interface has been determined to be better suited for the intended application than the mouse. Then, the touch screens is the input system adopted for the following experiments. Nevertheless, it should be pointed out that both with the mouse and the touch screens, the head of the operators was constantly moving from one screen to another searching for buttons. Then, the required effort to achieve low reaction times was quite high.

### 3.2.3 Experiment #3: Speech Synthesis

In this experiment, once each button appears on the screen, its label is told to the operator through the speakers. Therefore, two modalities (visual and aural) are involved simultaneously and the potential benefits can be analyzed.

In the interviews after the tests, it was mentioned that the workload is reduced with the speech synthesis as far as the operator can be relaxed until the “Yes” message is received. Then, it was observed that the head was more or less static if several “No” buttons appeared consecutively. Once a “Yes” message was heard, the operator moved his head from one screen to another searching for the “Yes” button (Table 6).

**Table 7** Summary of the results for the experiment #4

Individual	<i>n</i>	<i>n<sub>right_yes</sub></i>	<i>n<sub>right_no</sub></i>	<i>n<sub>wrong_no</sub></i>	$\overline{T_{yes}}$ (ms)	$\sigma$ (ms)
#1	364	168	196	0	1020.4	244.5
#2	344	157	187	0	1173.5	183.8
#3	376	209	165	2	1048.7	250.3
#4	382	193	188	1	953.2	241.1
#5	374	182	192	0	973.8	190.2
#6	376	190	186	0	982.7	184.0
#7	388	208	180	0	954.6	243.0
#8	380	173	207	0	891.6	181.4
#9	380	183	196	1	939.3	254.3

**Table 8** Summary of the results for the experiment #5

Individual	<i>n</i>	<i>n</i> <sub>right_yes</sub>	<i>n</i> <sub>right_no</sub>	<i>n</i> <sub>wrong_no</sub>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
#1	364	181	183	0	1062.0	279.5
#2	352	157	195	0	1094.9	154.2
#3	367	182	185	0	1041.4	235.3
#4	392	216	175	1	959.7	244.9
#5	372	168	204	0	956.4	226.2
#6	384	201	183	0	958.1	190.5
#7	375	191	184	0	1001.6	226.6
#8	382	188	192	2	948.5	202.3
#9	379	176	202	1	922.1	233.2

### 3.2.4 Experiment #4: 3D Audio Interface

This test is like the previous one, but adding the 3D audio technology. Depending on the location of the button on the screens (left, right or middle), the source of audio corresponding to its label is generated synthetically on the left, on the right or in front of the operator respectively through the headset. The goal is to evaluate the potential benefits of the 3D audio w.r.t. the conventional audio.

The results obtained are shown in the Table 7 and compared with the speech synthesis alone, it can be seen than the performance is better. In fact, it could be observed during the experiments that the individuals pointed their head directly on the right screen after hearing the “Yes” message. Then, the workload was lower due to two different factors:

- No need to pay attention while hearing “No” messages.
- Once a “Yes” button appeared, no need to search for the button from one screen to another (focus immediately on the screen with the “Yes” button instead).

### 3.2.5 Experiment #5: Tactile Interfaces

In this case, three wiimotes are used along with the touch screens. The devices are attached to the left and right arms, and also on the chest. The wiimote vibrator is

**Table 9** Summary of the results for the experiment #6

Individual	<i>n</i>	<i>n</i> <sub>right_yes</sub>	<i>n</i> <sub>right_no</sub>	<i>n</i> <sub>wrong_no</sub>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
#1	375	198	177	0	1017.3	336.8
#2	360	170	190	0	1061.0	215.8
#3	368	197	171	0	1068.5	258.5
#4	390	189	200	1	881.9	209.2
#5	387	191	196	0	910.7	204.4
#6	387	196	191	0	929.2	158.8
#7	384	202	182	0	963.8	220.7
#8	393	200	193	0	879.3	197.4
#9	389	209	180	0	953.7	241.4

**Table 10** Summary of the results for the experiment #7

Individual	<i>n</i>	<i>n<sub>right_yes</sub></i>	<i>n<sub>right_no</sub></i>	<i>n<sub>wrong_no</sub></i>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
#1	360	174	185	1	1073.6	325.1
#2	355	189	166	0	1154.7	304.7
#3	345	171	174	0	1202.6	299.3
#4	362	176	186	0	1067.9	304.5
#5	366	197	169	0	1092.4	380.6
#6	364	171	193	0	1025.0	257.1
#7	365	172	193	0	1014.0	232.2
#8	369	173	191	5	1023.2	281.7
#9	384	197	186	1	953.0	232.0

activated every time a “Yes” button appears on the screen. Moreover, depending on the location of the button on the window (left, right or middle), the wiimote on the left, on the right or on the chest vibrates respectively.

Table 8 shows values which are quite similar in mean to those obtained in the last experiment with the 3D audio interface. The reason is that the kind of benefits that the vibrators provide are essentially the same provided by the 3D audio:

- No need to pay attention while there is no vibration.
- Once a vibrator is activated, no need to search for the button from one screen to another (focus immediately on the screen with the “Yes” button instead).

### 3.2.6 Experiment #6: Integrated 3D Audio and Tactile Interfaces

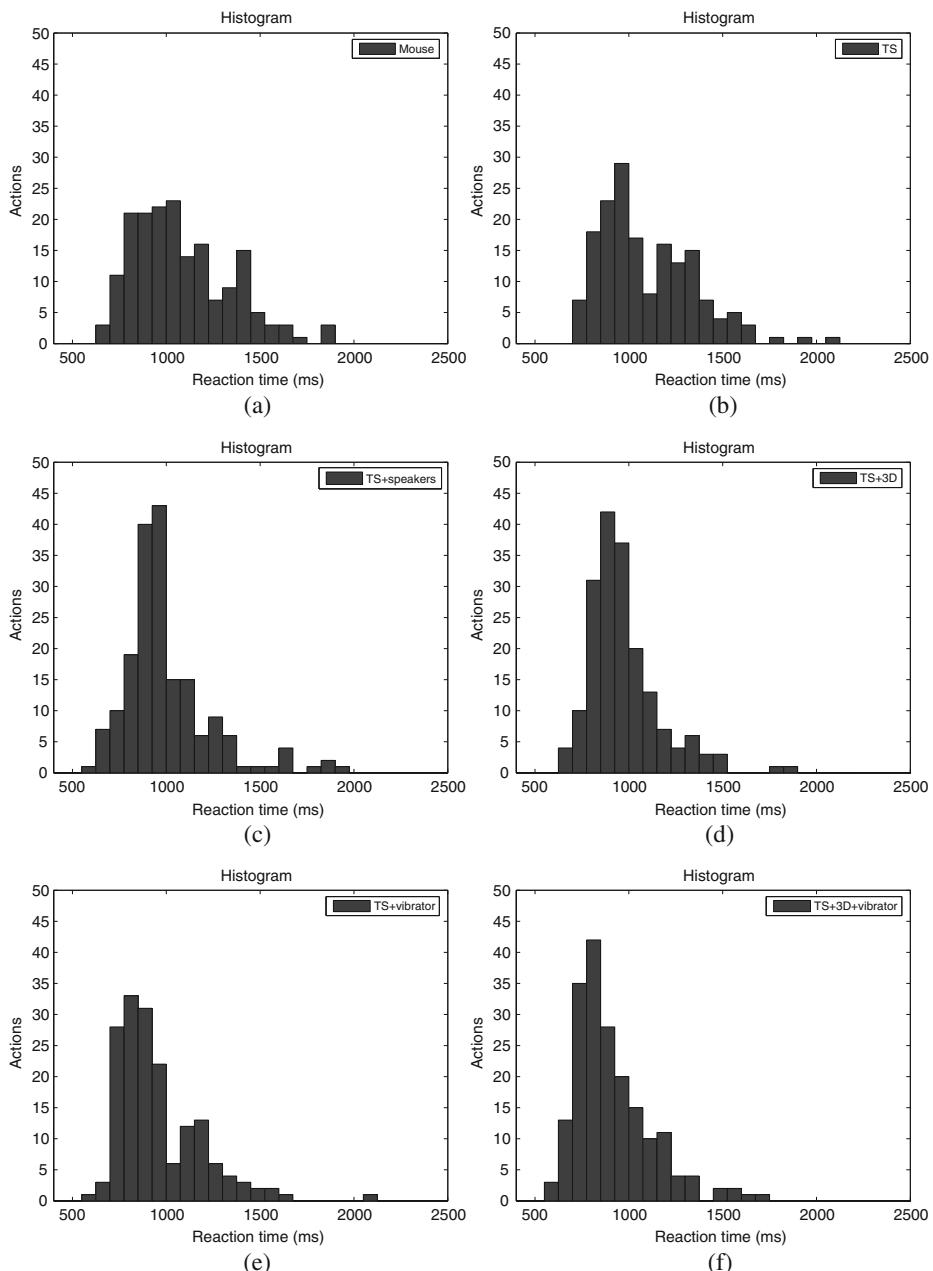
This test is a combination of the modalities involved in the last two experiments. The operator receives redundant information from the 3D audio and tactile interfaces. Then, depending on the location of the button on the screens (left, right or middle):

- the source of audio corresponding to its label is generated synthetically on the left, on the right or in front of the operator respectively through the headset, and
- if the button is a “Yes”, the wiimote on the left, on the right or on the chest vibrates respectively.

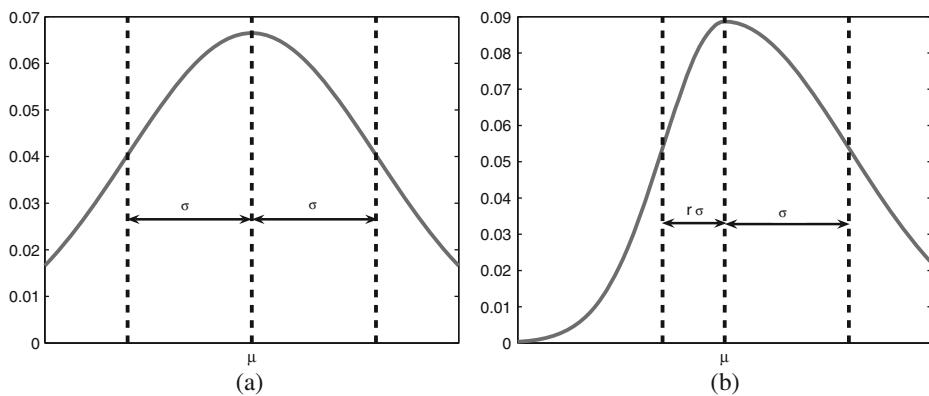
**Table 11** Summary of the results for individual #5

Experiment	<i>n</i>	<i>n<sub>right_yes</sub></i>	<i>n<sub>right_no</sub></i>	<i>n<sub>wrong_no</sub></i>	$\bar{T}_{yes}$ (ms)	$\sigma$ (ms)
Mouse	358	178	179	1	1098.7	322.6
TS	356	168	188	0	1097.9	251.9
TS+speakers	371	182	189	0	1001.2	232.2
TS+3D	374	182	192	0	973.7	190.2
TS+vibrator	372	168	204	0	956.4	226.1
TS+3D+vibrator	387	191	196	0	910.7	204.3
TS2	366	197	169	0	1092.4	380.5

In the Table 9, it can be observed that the results are slightly better than those presented in the previous two experiments. Therefore, it seems that the redundant information from the audio and tactile interfaces contributes to improve the performance of the operator.



**Fig. 5** Individual #5: Histograms with the number of correct actions in each reaction time interval for the different experiments (a–f)



**Fig. 6** Univariate Gaussian (a) and univariate asymmetric Gaussian (b)

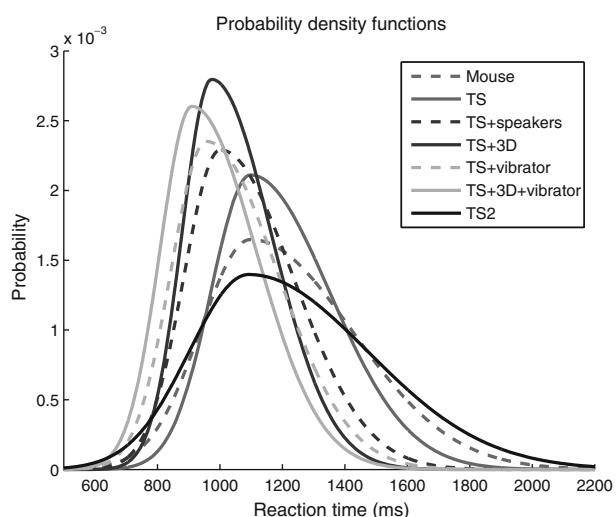
### 3.2.7 Experiment #7: Touch Screen Interface Repetition

The goal of this test is to check if the learning process of the user has any impact on the results. To fulfill this purpose, the individual is requested to repeat the test using only the touch screen interface after completing all the previous experiments. Comparing the results obtained in Table 10 with those corresponding to the second experiment (see Table 5), no significant improvement from the learning process arises.

## 4 Analysis of the Results

In order to compare in a more exhaustive manner the different technologies involved in the experiments, we will focus on the results from one individual. Table 11 shows several performance parameters of individual #5 in all the experiments.

**Fig. 7** Individual #5 reaction time probability density functions (using an approximation based on the univariate asymmetric Gaussians (UAGs) with  $r = 0.5$ )



On the other hand, Fig. 5 contains six histograms corresponding to the first six experiments (from Exp. #1 to #6) with the number of correct actions in several reaction time intervals. From those histograms the idea was to find a probability density function that could approximate them. The approach adopted is depicted in the next subsection.

#### 4.1 Probability Density Functions

Taking into account the histograms from the experiments and due to the nature of the measured values, it seems reasonable to use Gaussian distributions as an analytical approach for the results. However, the shape of the histograms computed is not symmetric with respect to the mean value (the decrease at the left is more abrupt than at the right of the mean value). Therefore, it has been considered that the probability model of the *asymmetric Gaussians* (AG) [6], which can capture temporal asymmetric distributions, could outperform Gaussian models.

Let  $\chi$  be the random variable associated to the reaction times measured in the experiments presented before. To indicate that a real-valued random variable  $\chi$  is normally distributed with mean  $\mu$  and variance  $\sigma^2 \geq 0$ , we write

$$\chi \sim \mathcal{N}(\mu, \sigma^2) \quad (2)$$

The continuous probability density function of the normal distribution is the Gaussian function

$$\varphi_{\mu, \sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (3)$$

where  $\sigma > 0$  is the standard deviation and the real parameter  $\mu$  is the expected value.

We now introduce an asymmetric Gaussian (AG) model with the following distribution:

$$\varphi_{\mu, \sigma^2, r}(x) = \frac{2}{\sigma(r+1)\sqrt{2\pi}} \begin{cases} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) & \text{if } x > \mu, \\ \exp\left(-\frac{(x-\mu)^2}{2r^2\sigma^2}\right) & \text{otherwise} \end{cases} \quad (4)$$

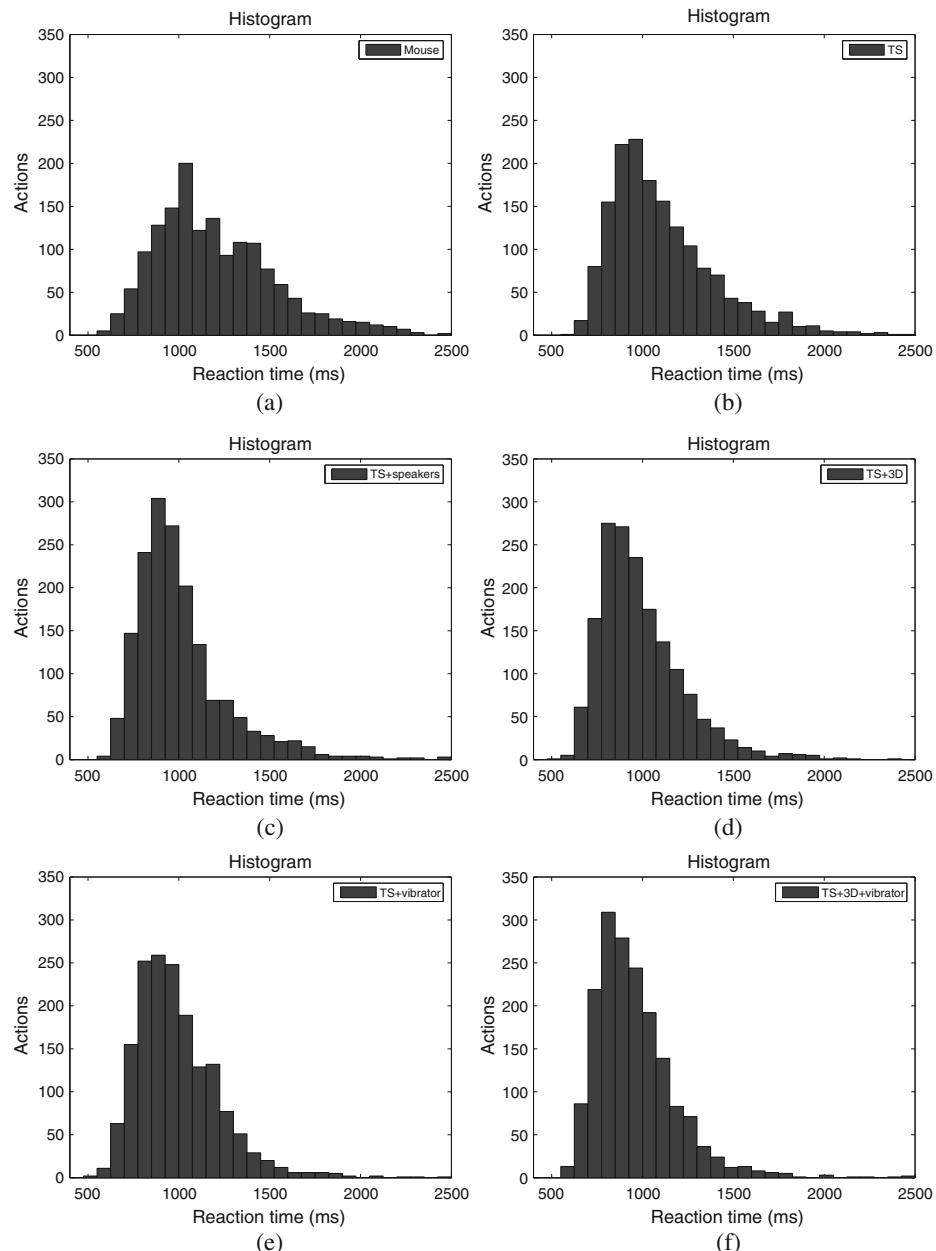
with  $\mu$ ,  $\sigma$  and  $r$  as parameters. We term the density model (4) *univariate asymmetric Gaussian* (UAG). It is shown that UAG have an asymmetric distribution by the Fig. 6b, where the density function is plotted. In addition, UAG is an extension of a Gaussian since UAG with  $r = 1$  is equivalent to the Gaussian distribution.

Then, the next step was to approximate each histogram by an UAG distribution. For example, for the histograms in Fig. 5, the values of  $\mu$  and  $\sigma$  have been already

**Table 12** Summary of the improvements in mean with respect to the results of Experiment #2 (TS): percentage of reduction in the mean reaction time ( $\Delta\bar{T}_{\text{yes}}$ ) and also in the standard deviation of the reaction times ( $\Delta\sigma$ )

	Mouse	TS	TS+speakers	TS+3D	TS+vibr	TS+3D+vibr	TS2
$\Delta\bar{T}_{\text{yes}}$ (%)	-9.33	0	8.89	11.18	10.97	13.78	4.41
$\Delta\sigma$ (%)	-19.73	0	13.91	24.93	24.46	23.68	1.04

computed (see Table 11). Selecting  $r = 0.5$  and plotting the UAGs corresponding to the first six experiments in the same figure, it is possible to compare easily the different modalities tested (see Fig. 7).



**Fig. 8** Histograms with the number of correct actions in each reaction time interval for the whole population during the different experiments (a–f)

## 4.2 Comparative Results among Technologies

After collecting the full set of data from all the individuals in all the experiments, it has been processed in order to obtain a general comparison among the technologies involved in the experiments.

In a first step, the improvement in the mean reaction time and in its standard deviation has been computed for all the individuals participating in the experiments (see Table 12). This improvement has been expressed in percentage and computed with respect to the results in the Experiment #2, in which only the touch screens were used (a negative value in the percentage means that the performance was worse).

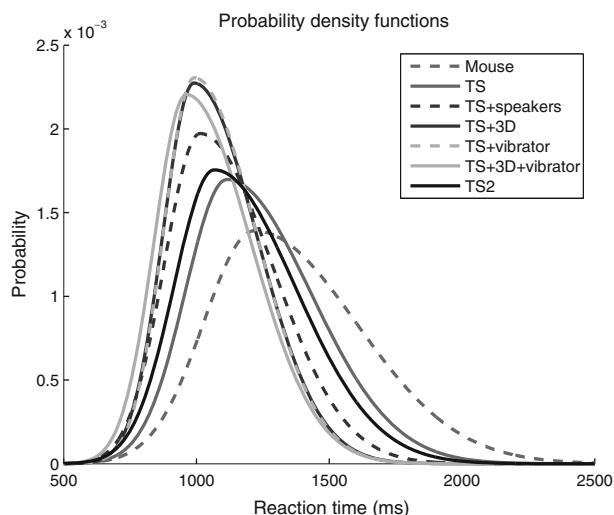
It can be seen that the progressive introduction of better multimodal technologies from the first experiment to the sixth one improves the performance of the operator. On the other hand, when equivalent technologies are used (i.e. 3D audio or vibrators), the results obtained in mean are quite similar (although each individual could show preference for one of them).

It should be pointed out that there is a “minimum” response time due to the limitations of the operating system and the electronic components and interfaces involved in the system. This minimum response time has been estimated to be approximately 100 ms. Then, if we remove this interval from the computed mean reaction times, the percentages of improvement presented in Table 12 would have higher values.

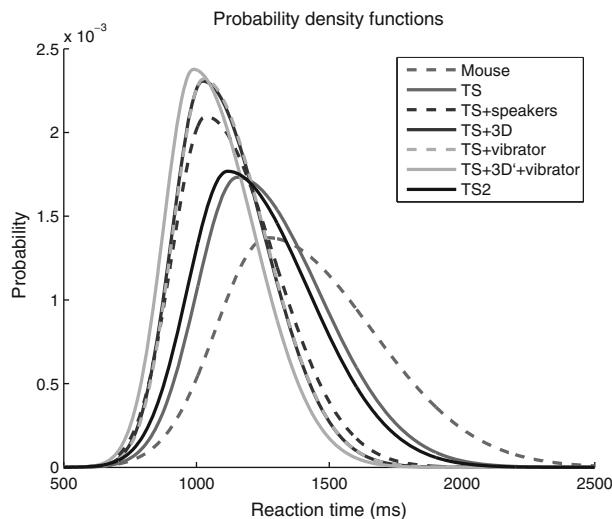
The histograms for the whole population in the experiments from #1 to #6 are shown in Fig. 8. Comparing this figure with the histograms of the individual #5, it can be seen that when the number of samples increases, the shape of the histograms is more similar to the UAG distribution adopted for the analysis.

Then, using the values of  $\mu$  and  $\sigma$  for the whole population and with  $r = 0.5$  the UAG distributions for the experiments from #1 to #6 are computed and plotted together in Fig. 9. This figure allows to compare the impact of each modality for the whole population at a glance. The Gaussians move from right to left as we use better

**Fig. 9** Reaction times probability density functions for the whole population in the different experiments



**Fig. 10** Reaction times probability density functions for the whole population in the different experiments considering only the transitions from one screen to another



modalities in the interface, because the mean reaction times are lower. Moreover, the shape of the Gaussians is narrower also from right to left as far as the standard deviation is lower.

Finally, during the experiments it was also registered the screen where each “Yes” button was pressed by the operator, allowing us to compute the reaction times when a transition from one screen to another happened. Using this information, the UAG distributions for the reaction times of the transitions were calculated (see Fig. 10). The Gaussians are slightly displaced to the right with respect to Fig. 9 as expected (the mean reaction times are higher for the transitions) and the benefit of adding modalities is clearer.

## 5 Conclusions and Future Developments

Multimodal display techniques may improve operator performance in the context of a Ground Control Station (GCS) for Unmanned Aerial Vehicles (UAVs). Presenting information through two or more sensory channels has the dual benefit of addressing high information loads as well as offering the ability to present information to the operator within a variety of environmental constraints.

This paper has explored different technologies that can be applied in the design and development of a GCS for UAVs equipped with a multimodal interface. The applicability and benefits of those technologies has been analyzed for a task consisting in the acknowledgement of alerts in an UAV ground control station composed by three screens and managed by a single operator. The system integrated visual, aural and tactile modalities and multiple experiments have shown that the use of those modalities has improved the performance of the users of the application.

Regarding the multimodal application used to obtain the results presented in this paper, there are several possible improvements. One of them would be to compute the exact position of each button on the screen when it is pressed. It will allow to

estimate the stochastic relation between the reaction times, the different modalities and the distance between buttons.

On the other hand, the wiimote devices were used in the experiments as wireless vibrators to signal the alarms. But their internal accelerometers can also provide information about the motion of the arms of the operator during the mission, allowing to measure the level of stress for instance.

Finally, it could be interesting to integrate a head-tracking system for the operator in the platform. This system will allow to compute an estimation of the screen at which the head of the operator is pointing at. This information can be used to show each alarm in the screen where the attention of the user is focused, and evaluate its benefits for the operation. Additionally, it can be used along with other body sensors to evaluate the state of the user (level of attention, stress, etc.).

**Acknowledgements** The authors would like to thank the *Boeing Research & Technology Europe* company for their financial and technical support.

## References

1. Biocca, F., Jin, K., Choi, Y.: Visual touch in virtual environments: an exploratory study of presence, multimodal interfaces, and cross-modal sensory illusions. *Presence: Teleoperators and Virtual Environments* **10**(3), 247–265 (2001)
2. Origin Instruments Corporation: Headmouse extreme. <http://www.orin.com/access/headmouse/> (2009)
3. Craven, P., Belov, N., Tremoulet, P., Thomas, M., Berka, C., Levendowski, D., Davis, G.: Foundations of augmented cognition, chap. cognitive workload gauge development: comparison of real-time classification methods, pp. 75–84. Springer, New York (2006)
4. Creative Labs: OpenAL: cross-platform 3D audio library. <http://www.openal.org/> (2009)
5. Free Software Foundation: FreeTrack. <http://www.free-track.net/english/> (2009)
6. Kato, T., Omachi, S., Aso, H.: Asymmetric gaussian and its application to pattern recognition. In: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, pp. 405–413. Springer, London (2002)
7. Lemon, O., Bracy, A., Gruenstein, A., Peters, S.: The WITAS multi-modal dialogue system I. In: Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH), pp. 1559–1562. Aalborg, Denmark (2001)
8. Madentec: Tracker Pro. <http://www.madentec.com/products/tracker-pro.php> (2009)
9. Maza, I., Caballero, F.: Video summarizing the experiments reported in this paper. [http://grvc.us.es/JINT\\_multimodal](http://grvc.us.es/JINT_multimodal) (2009)
10. McCarley, J.S., Wickens, C.D.: Human factors implications of UAVs in the national airspace. Tech. Rep. AHFD-05-5/FAA-05-1, Institute of Aviation, Aviation Human Factors Division, University of Illinois at Urbana-Champaign (2005)
11. NaturalPoint: SmartNav 4 AT. <http://www.naturalpoint.com/smartnav/> (2009)
12. NaturalPoint: TrackIR 4. <http://www.naturalpoint.com/trackir/02-products/product-TrackIR-4-PRO.html> (2009)
13. Ollero, A., Garcia-Cerezo, A., Gomez, J.: Teleoperacion y Telerobotica. Pearson Prentice Hall, Englewood Cliffs (2006)
14. Ollero, A., Maza, I. (eds.): Multiple heterogeneous unmanned aerial vehicles, chap. teleoperation tools, pp. 189–206. Springer Tracts on Advanced Robotics. Springer, New York (2007)
15. Orden, K.F.V., Viirre, E., Kobus, D.A.: Foundations of Augmented Cognition, chap. Augmenting Task-Centered Design with Operator State Assessment Technologies, pp. 212–219. Springer, New York (2007)
16. Patterson, R.D.: Guidelines for Auditory Warnings on Civil Aircraft. Civil Aviation Authority, London (1982)
17. Peryer, G., Noyes, J., Pleydell-Pearce, K., Lieven, N.: Auditory alert characteristics: a survey of pilot views. *Int. J. Aviat. Psychol.* **15**(3), 233–250 (2005)

18. Poythress, M., Berka, C., Levendowski, D., Chang, D., Baskin, A., Champney, R., Hale, K., Milham, L., Russell, C., Seigel, S., Tremoulet, P., Craven, P.: Foundations of Augmented Cognition, chap. Correlation between expected workload and EEG indices of cognitive workload and task engagement, pp. 75–84. Springer (2006)
19. Sharma, R., Pavlovic, V.I., Huang, T.S.: Toward multimodal human-computer interface. Proc. IEEE **86**(5), 853–869 (1998)
20. Cachya Software: Cachya. <http://www.cachya.com/esight/overview.php> (2009)
21. Stanford University: WITAS Project multi-modal conversational interfaces. <http://www-csli.stanford.edu/semlab-hold/witas/> (2009)
22. Sweller, J.: Visualisation and instructional design. In: Proceedings of the International Workshop on Dynamic Visualizations and Learning (2002)
23. EyeTech Digital Systems: EyeTech TM3. <http://www.eyetechds.com/index.htm> (2009)
24. University of Edinburgh: The festival speech synthesis system. <http://www.cstr.ed.ac.uk/projects/festival/> (2009)
25. Wilson, G., Russell, C.: Real-time assessment of mental workload using psychophysiological measures and artificial neural networks. In: Human Factors, pp. 635–643 (2003)
26. Zhu, Y.: Advances in Visual Computing, chap. Measuring Effective Data Visualization, pp. 652–661. Springer, New York (2007)

## Multi-UAV Simulator Utilizing X-Plane

Richard Garcia · Laura Barnes

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 6 October 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper describes the development of a simulator for multiple Unmanned Aerial Vehicles (UAVs) utilizing the commercially available simulator X-Plane and Matlab. Coordinated control of unmanned systems is currently being researched for a wide range of applications, including search and rescue, convoy protection, and building clearing to name a few. Although coordination and control of Unmanned Ground Vehicles (UGVs) has been a heavily researched area, the extension towards controlling multiple UAVs has seen minimal attention. This lack of development is due to numerous issues including the difficulty in realistically modeling and simulating multiple UAVs. This work attempts to overcome these limitations by creating an environment that can simultaneously simulate multiple air vehicles as well as provide state data and control input for the individual vehicles using a heavily developed and commercially available flight simulator (X-Plane). This framework will allow researchers to study multi-UAV control algorithms using realistic unmanned and manned aircraft models in real-world modeled environments. Validation of the system's ability is shown through the demonstration of formation control algorithms implemented on four UAV helicopters with formation and navigation controllers built in Matlab/Simulink.

**Keywords** Unmanned air vehicles · Simulator · X-Plane

---

R. Garcia (✉)

Army Research Laboratory, Aberdeen Proving Ground, Aberdeen, MD 21005, USA  
e-mail: richard.d.garcia@arl.army.mil

L. Barnes

Automation and Robotics Research Institute, University of Texas at Arlington,  
Arlington, TX, USA

## 1 Introduction

The significant increase of UAVs can be attributed to many factors including their ability to extend situational awareness and remove the human from dangerous situations. In fact, there are currently over 950 UAVs operating in Iraq alone logging more than 14,000 h of operation per month [1]. Unfortunately, these vehicles act as a force divider where a job that usually takes one or two soldiers to complete alternatively takes ten soldiers to accomplish using a single UAV. This fact can be attributed to lack of autonomy, inefficient human-computer interfaces, and inabilities to coordinate efforts. This research attempts to help alleviate the latter of these issues by allowing the group of vehicles to act as one entity creating a force multiplier.

In practice UAVs are largely disconnected and act as distinct entities sharing little if any information between systems. This lack of cooperation and coordination creates a highly inefficient and possibly dangerous environment. By coordinating the control of multiple UAVs they can easily be tasked to create safe and optimal flight paths, work as groups to solve large complex task, and act as backup systems for failed vehicles which is estimated to occur every 80–110 h of operation [2].

The first step following the development of new and creative algorithms is to thoroughly test them. Initial testing for most types of vehicles today is handled through vigorous simulations. Unfortunately, very few simulators exist that are capable of accurately simulating multiple air vehicles in real-time. To date, software that is capable of simulating multiple air vehicles is typically developed and utilized in-house. These simulation systems typically provide either a small number of highly accurate vehicle models operating in a highly confined environment model [3–5] or provide simplified models that decrease the computational load on the simulator [6, 7]. Both types of simulation systems require the end user to develop vehicle models and very rarely include realistic world models capable of simulating outside effects such as wind, rain, thermals and microburst.

Developing vehicle and world models is a very intensive and time consuming task and many times replicates models that have already been designed for commercial off-the-shelf (COTS) simulators. In general, COTS simulators provide a database of both world and vehicle models that many times have decades of dedicated development and refinement. COTS simulators also have the advantage of highly refined graphical outputs, special optimizations for RAM and video adapters, and many times contain Federal Aviation Administration (FAA) certified versions.

Beyond being able to supply heavily developed and certified models, COTS simulators can provide a public median for allowing researchers to easily replicate work performed by other facilities. For example, this work focuses on the COTS simulator X-Plane. X-Plane has also been utilized for hardware in the loop testing using [8], UAS failure control in [9], and for heads up displays in [10]. By adapting a COTS simulator to support multi-vehicle simulations the integration of diverse research areas becomes more viable.

## 2 Simulator Hardware

At a high level, the simulation environment is a cluster of individual machines capable of supplying sufficient computational and graphical power to collectively

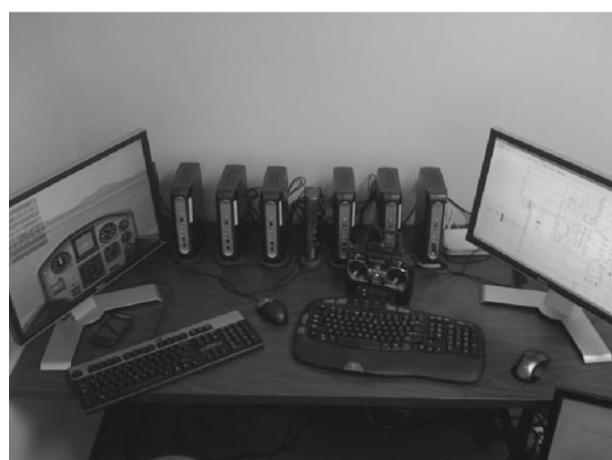
simulate, control, and visualize multiple aerial vehicles. This is accomplished by allowing each individual machine to act as a vehicle. Each vehicle, or machine, is responsible for simulating its own dynamics and kinematics, collecting internal state data, communicating its state data with outside machines and implementing externally received control inputs, all of which are performed by X-Plane. This configuration directly lends itself to hardware-in-the-loop simulation where physical hardware, be it sensors or computational devices, are simply attached to or replace the specific machine.

Although infinite variations of hardware exist to configure a functional system, special attention must be given towards hardware to provide the desired results. More specifically, the hardware must meet the minimal criteria for operating the COTS simulator as well as communicating data in a rapid effective manner. For the purposes of this research, the minimal criterion was defined as the ability to graphically operate the simulator at a rate greater than or equal to 50 Hz. This criteria was selected because the utilized simulator's graphical rate is directly connected to state and control data I/O (Input/Outputs).

The hardware configuration for the simulation environment, as seen in Fig. 1, has the following characteristics:

- Simulation Machines:
  - Quantity: 7
  - Motherboard: Mini-ITX
  - CPU: 2.8 GHz Duo
  - Graphics: NVidia GeForce 7100
  - RAM: 4 Gb
- Control Machine:
  - Quantity: 1
  - Motherboard: Mini-ITX
  - CPU: 2.4 GHz Duo
  - Graphics: Intel Integrated
  - RAM: 2 Gb

**Fig. 1** Picture of the simulation environment



- Network:
  - Switch: 1,000 MHz
  - Router: 1,000 MHz

The rational for choosing this hardware configuration is multi-fold. First, the use of the mini-ITX motherboard provides the ability to utilize state of the art processors and maximize RAM size while providing a small footprint. Second, by utilizing a cluster of processing systems decentralized control can be directly implemented into the simulator. Third, the NVidia graphics adapter provides the required capabilities to sufficiently operate the simulator. It should be noted that the integrated Intel video adapter on the control machine was incapable of providing this requirement. Fourth, the high speed switch allows the system to maximize the number of simulated vehicles that can be controlled at once. This allows researchers to test the scalability of their algorithms, discussed later. Last, the low cost of the assembled environment, approximately \$3,500 USD.

### 3 Simulator Software

#### 3.1 X-Plane Simulator

The simulation environment is built around the COTS simulator X-Plane. The benefits of using X-Plane are:

- FAA Certifications: X-Plane can provide Federal Aviation Administration (FAA) certified simulation and vehicle models. This allows researchers to achieve high levels of confidence in simulation results.
- Operating Systems (OS) Flexibility: The X-Plane simulator is capable of running on Linux, Windows, or Macintosh OS.
- Database of Vehicle Models: Thousands of manned and unmanned vehicle models are freely available for download at multiple internet sites [11]. Although not necessarily certified, these models provide quick starting points for testing and can be easily modified using tools supplied with X-Plane. In addition, researchers can develop their own vehicle models for X-Plane.
- Highly Developed World Models: This simulator has been receiving constant refinements for over a decade. The world model used in X-Plane is able to simulate, cloud cover, rain, wind, thermals, microburst, and fog to name a few.
- Precedence of Success: X-Plane has been used as the basis for designing and testing control algorithms that have ultimately been proved viable on actual hardware in field experimentation [12].
- Visualization: X-Plane is designed to provide a visual collaboration of up to ten vehicles. This means that up to ten vehicles can be visualized simultaneously from a single display allowing for visual debugging and data collection to be performed without the need for extra software.

The main limitation of X-Plane, as with most COTS simulators, is the inability to run multiple instances of itself on a single machine. This limitation goes beyond

simply not being able to start two identical processes simultaneously. This work originally attempted to bypass this limitation in an attempt to utilize as few machines as possible. Experiments were performed utilizing Linux and Windows OSs as well as OS emulators, VMWare, VirtualBox and Microsoft Virtual PC. All attempts to bypass this limitation failed for various reasons most of which were related directly to the graphics adapter or emulated graphics adapter.

To account for this limitation, the simulation environment is built around a cluster of small low cost machines. These machines are interconnected through a switch which allows them to pass state and control data to any or all machines on the network. This design gives rise to a possible bottleneck where data transmissions flood the network preventing adequate data transfer. This essentially becomes a scalability issue and is approached later in this section.

Due to the design of the simulation environment vehicle navigation and control can be distributed to multiple machines or performed on a single machine. For distributed type control, each individual machine could be responsible for its own control input or some subset of machines could collectively be calculating and distributing control to all simulators. In this work a single control machine is dedicated for calculating and distributing control inputs for all simulation machines. Specific details of the navigation and formation algorithms used to validate the simulation environment are presented in the following sections.

X-Plane's standard method for communicating with external processes and machines is User Datagram Protocol (UDP). UDP provides a minimal overhead communication method for passing data between nodes. Unlike Transmission Control Protocol (TCP), UDP is a non-guaranteed protocol and gives no assurances that data packets will arrive in order or at all. UDP is designed to minimize bandwidth usage but presents a possible problem resulting from corrupt data. Although this problem does exist and may need to be addressed given specific system designs, degradation of control and simulation has been unseen in X-Plane experimentations.

One issue that has yet to be mentioned is the scalability of the simulation environment. The two obvious bottlenecks for the described simulation environment are flooding of the network bandwidth and computational limitations of a centralized control machine. The computational ability of the centralized control machine is directly related to the method of control being tested. If the control machine is unable to perform the necessary calculations then the utilized method, or algorithms, may not scale well. This can be verified by using hardware-in-the-loop and should be considered experimentation all in itself. Ultimately, if the researcher is unconcerned with scalability on the current processor, the controlling machine can simply be upgraded to a machine capable of the desired computations.

The network bottleneck is ultimately a matter of understanding the hardware limitations as compared to the size and rate of messages passing through the network. Simply stated, as the number of simulated vehicles increases the available bandwidth on the network will decrease. By utilizing a network switch this simulation environment avoids packet collision and is available to take advantage of the hardware's full duplex communication median. Thus, the network scalability of the system is limited by the maximum inflow or outflow from any single node on the network. In the case of centralized control, the control machine will be the obvious bottleneck for communication as all simulation nodes will communicate directly with this one machine. Since both state data and control packets are identical in size, the number

of vehicles that can be simulated without exceeding the network's bandwidth can be calculated using:

$$N_{sim} = \frac{R_{net}}{(P_{head} + N_{msg} * D_{size}) * F} \quad (1)$$

where  $R_{net}$  is the speed of the switch,  $F$  is the frequency of data packets and the UDP packet size is made up of the packet header,  $P_{head}$ , the size of an individually selected state data item,  $D_{size}$ , and the number of state data items selected,  $N_{msg}$ . Note that this calculation assumes optimal functionality of both the network and cluster. For the hardware described in the previous section using our method for controlling multiple helicopters, described in the next section, these values are:  $R_{net} = 1,000$  Mb,  $P_{head} = 40$  b,  $N_{msg} = 7$ ,  $D_{size} = 288$  b, and  $F = 50$ . Using this data the maximum number of vehicles that can be simulated without exceeding bandwidth limitations is 9,727 vehicles. Realistically, control calculation time and minor delays in network traffic would reduce this number but will leave ample room for growth for large scale simulation.

It should be noted that X-Plane communication packets as well as data rates are user selectable and can vary from position and velocity data to carburetor temperatures and fuel pressure. Also, data rates cannot exceed the rendering speed of the simulator. Thus, if the simulator is only rendering frames at 25 Hz then state data will be limited to 25 Hz. Rendering rates can be improved by limiting rendering options in the simulator, decreasing visibility, and removing weather items such as cloud cover.

### 3.2 Control Software

Control software for the simulation environment is fundamentally unlimited. Any software capable of directly or indirectly receiving and transmitting UDP data can be seen as a viable option for implementing control algorithms. As such, the remainder of this section details the Matlab/Simulink model used to validate the simulation environment.

The Simulink model used to validate the simulation environment can be broken down into three major groups:

- Mission Controller: This module is designed to control the overall position of the group by selecting a setpoint in space from which the vehicles formation will originate. Advancement from setpoint to setpoint is determined by comparing each vehicle's position to its individual desired position.
- Formation Controller: This module is responsible for calculating the individual setpoints for each vehicle based on the desired formation parameters and setpoint provided by the mission controller. Individual setpoints include a desired altitude and heading as well as a 2-D vector representing a magnitude of error and a direction of travel.
- Vehicle Controller: The vehicle control module is responsible for the stabilization and navigation of each individual vehicle. As such, there will be a distinct vehicle control module for every simulated vehicle in X-Plane. This module includes the control algorithms, state data computation algorithms, and the X-Plane communication function.

These three major modules collectively make up the upper level of the Simulink model. A top-level view of a Simulink model for four vehicles can be seen in Fig. 2. It should be noted that the only identifier used to distinguish between individual vehicles is the port number, which is passed to each Vehicle Controller module. This identifier is only used by the X-Plane communication function and simply identifies the port number on which a specific simulator will be sending data.

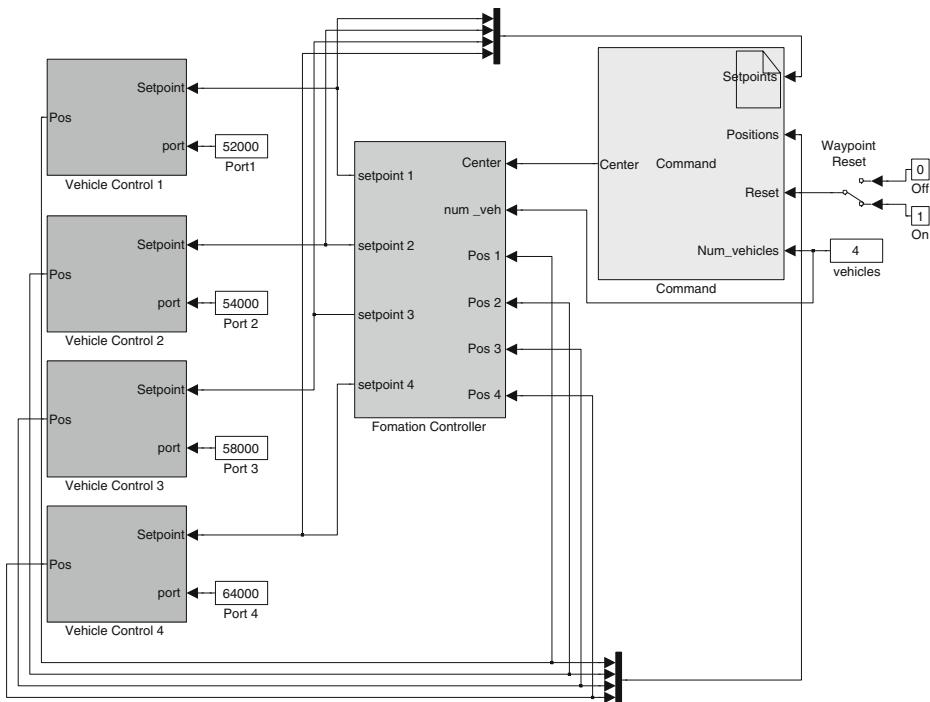
## 4 Controller

### 4.1 Formation Controller

In order to describe the formation controller, suppose that the UAVs need to maintain an elliptical ring configuration at a fixed altitude, defined loosely by the ring's dimensions and the center of mass. Using a properly generated field, members of the swarm can be attracted to the loose elliptical ring. This formation can be described using a sequence of three concentric ellipses with center  $(x_c, y_c)$ . Figure 3 depicts three elliptical rings with center  $(x_c, y_c)$ . The ultimate goal is to attract swarm members to the center elliptical ring  $R^*$  described as the set of points  $(x, y) \in \mathbb{R}^2$  satisfying:

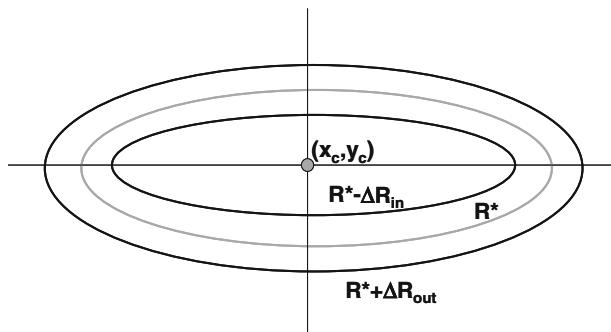
$$R^{*2} = (x - x_c)^2 + \gamma (y - y_c)^2 \quad (2)$$

where  $(x_c, y_c)$  is the center and  $\gamma$  is the axis ratio.



**Fig. 2** Top-level of the simulink model

**Fig. 3** Elliptical attraction band for the swarm



The general formation controller is described by:

$$V(x, y, t) = \sum_1^N w_i(x, y, t) V_i(x, y, t) \quad (3)$$

where  $V(x, y, t)$  gives the velocity of the swarm at a particular time and place. Each of the vectors  $V_i(x, y, t)$  is associated with different fields and  $w_i(x, y, t)$  are weights of the overall contribution of the  $i$ th vector. In general, the field  $V_i(x, y, t)$  is the weighted sum of  $N$  different vectors, each of which is acting on the swarm. In the case of this work, three different vector fields are utilized: one attracts UAVs to the elliptical band from points outside the elliptical region; one pushes UAVs away from the center towards the desired band; and one controls the movements of the UAVs within the band.

The potential field based controller uses a small number of physically relevant weights,  $w_i$ , and vectors  $v_i$  that attract UAVs to a neighborhood of the  $R^*$  ellipse. This neighborhood is shown in Fig. 3. The parameters  $R_{in}$  and  $R_{out}$  denote the inside and outside boundaries of the  $R^*$  neighborhood, respectively, as also depicted in Fig. 3. The desired vector fields will ‘trap’ the UAVs in these bands. Typically, this is a very narrow band of allowable space for the UAVs with a controllable width of  $\Delta R_{in} + \Delta R_{out}$  where:

$$R_{in} = R^* - \Delta R_{in} \quad (4)$$

$$R_{out} = R^* + \Delta R_{out} \quad (5)$$

The vector field is constructed utilizing the normalized gradient. For every  $(x, y)$ , the gradient field vector has the form:

$$V_i(x, y) = \begin{cases} W_i(x, y) \frac{1}{L(x, y)} \begin{pmatrix} (x - x_c) \\ \gamma(y - y_c) \end{pmatrix} & \text{for } (x, y) \neq (x_c, y_c) \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{for } (x, y) = (x_c, y_c) \end{cases} \quad (6)$$

where:

$$L(x, y) = \sqrt{(x - x_c)^2 + \gamma^2(y - y_c)^2} \quad (7)$$

The vector  $\frac{1}{L(x,y)} \begin{pmatrix} (x - x_c) \\ \gamma (y - y_c) \end{pmatrix}$  is a unit vector that provides the direction of the vector at  $(x, y)$ . The function  $w(x, y)$  provides the magnitude of the vector at that point. Notice that for any  $(x, y)$ , this vector points *away* from the center of the ellipse.

In the defined vector field, particles starting within the  $R^* - \Delta R_{in}$  ellipse with:

$$R^* = \sqrt{(x - x_c)^2 + \gamma^2 (y - y_c)^2} \quad (8)$$

move out from the center until they reach the  $R^*$  neighborhood. UAVs starting outside the  $R^* + \Delta R_{out}$  ellipse move towards the center until they reach the  $R^*$  neighborhood. Eventually all UAVs will be trapped within the neighborhood given by:

$$(R^* - R_{in}) \leq r \leq (R^* + R_{out}) \quad (9)$$

Avoidance of individual swarm members including their dispersion about the formation is controlled by another weight,  $w_i$ , which limits how close UAVs are allowed to be to one another. This is a user-defined parameter,  $\Delta R_{avoid}$ . The formation controller is described in further detail in [13–15].

#### 4.2 Vehicle Controller

Stabilization and navigation of the individual vehicles is performed using a previously developed fuzzy logic controller. This was done to simplify the simulation environment's verification process by utilizing controllers that were already known to function properly and had thousands of hours of testing using the simulator X-Plane. These controllers are thoroughly detailed in [12] and therefore will only be summarized here.

Fuzzy logic was chosen to implement the helicopter controllers due to its inherent ability to handle minor errors, sensor noise, and contradictory inputs. Four distinct and uncoupled fuzzy controllers are utilized to both stabilize and navigate the helicopters. These individually control the roll, pitch, heading and collective of the vehicle. Throttle is assumed to be controlled by the simulator's throttle governor. The four fuzzy controllers utilize Sugeno constant fuzzy logic and a weighted average defuzzification method where the weight for all rules is equal to one. All rules for the controllers are based on the 'and' method and use membership products to determine the strength of each rule.

Although the fuzzy rule base was not modified from [12], there was minor editing performed on two input variables to allow the controllers to conform to desired multi-vehicle control. More specifically, the formation control algorithms were designed to provide vehicles with potential field vectors guiding them into formation. The fuzzy controllers described in [12] were not designed with this type of input in mind. Although this presented an initial issue, it was determined that the potential field vector could be seen as representing a unitless position error. Assuming that this unitless position error could be approximated in units of length provides the expected input into the fuzzy controllers.

This conversion is done by separating the X and Y vector components of the potential field vector, multiplying them by a constant and rotating them to the local coordinate system of the vehicle,

$$P_E = (-V_x^* \cos(-\psi) + V_y^* \sin(-\psi))^* c \quad (10)$$

and

$$R_E = (-V_x^* - \sin(-\psi) + V_y^* \cos(-\psi))^* c \quad (11)$$

where  $V_x$  and  $V_y$  are the magnitude of the potential field vector on the x and y axis respectively,  $\psi$  is the Euler yaw angle of the vehicle, and  $c$  is a constant used to adjust the magnitude of the conversion. The magnitude constant,  $c$ , is simply a way of linearly adjusting how much error a vector magnitude has. For the purposes of the work the magnitude constant was set to seven. This value allowed the small range of the potential vector to be converted to a positional error with seven times the range. It should be noted that increasing the magnitude constant increases the navigation control of the vehicle and decreases the vehicle's time to acquire a desired position. Due to the linear nature of the magnitude constant, an increase in its value also lead to a steady state sinusoidal error when the vehicle was near its desired position. This was deemed a limitation of the conversion equation and can be reduced by utilizing a non-linear equation to increase the range of the conversion. Although easily accomplished it was deemed unnecessary for this work.

## 5 Simulations

In order to demonstrate and validate the multi-UAV simulator, simulations were run with four simulated UAVs. The model RC UAV utilized for simulation was freely downloaded from the web and slightly tuned to mimic the Maxi Joker II electric helicopter.

### 5.1 Hovering Formation

In the first simulation, all UAVs initially start at ground level in random positions and are given a static center ( $x_c, y_c$ ). The desired altitude is fixed as we are only concerned with formation at constant altitude for these simulations. Once the system is initialized the UAVs takeoff and immediately begin to surround the provided center point in an ellipse formation with the parameters given in Table 1.

Figure 4 depicts the X-Plane simulation environment with the four UAVs hovering in formation. In Fig. 5, the final positions of the UAVs in the formation are shown. Figure 6 plots the distance from the center of the formation over time. The UAVs remain at a consistent distance from ( $x_c, y_c$ ) with minor perturbations due to

**Table 1** Formation parameters

Formation	$R^*$	$\gamma$	$\Delta R_{in}$	$\Delta R_{out}$	$\Delta R_{avoid}$
Ellipse	100	1.0	20	20	100
Flocking	50	1.0	—	—	20

**Fig. 4** Screenshot of X-Plane during formation control experiment

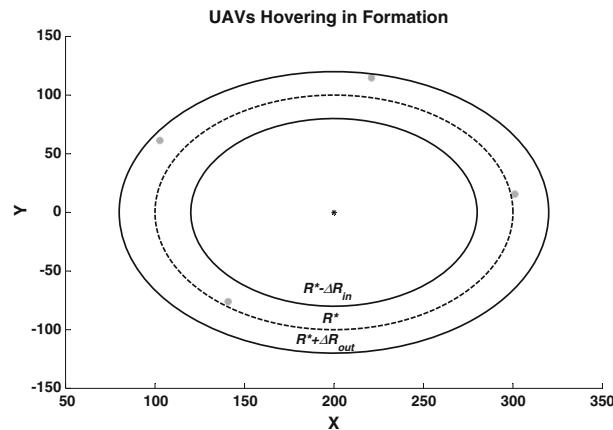


untrimmed mechanical offsets and the low resolution of single floating point GPS coordinates, approximately 0.8 m [16]. Figure 7 shows the distance between UAVs. From Figs. 5 and 8, it can be seen that the UAVs are evenly distributed about the formation.

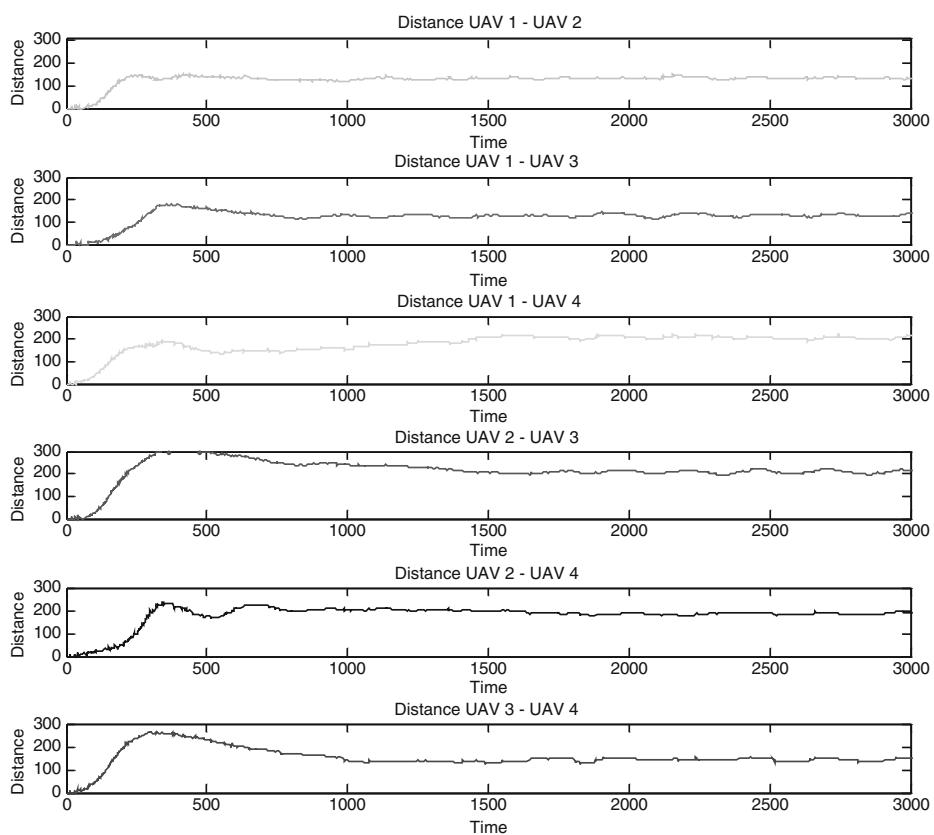
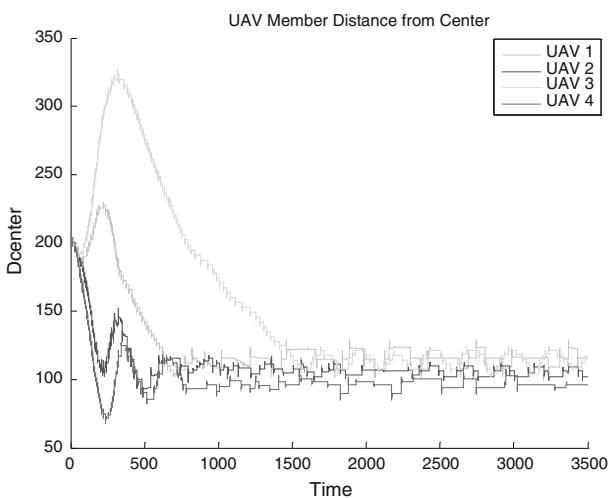
## 5.2 Multi-UAV Flocking

The flocking behavior was demonstrated by turning off the weights,  $w_i$ . Figure 8 demonstrates the swarm following a trajectory at different times. The UAVs follow

**Fig. 5** Four UAVs hovering in formation

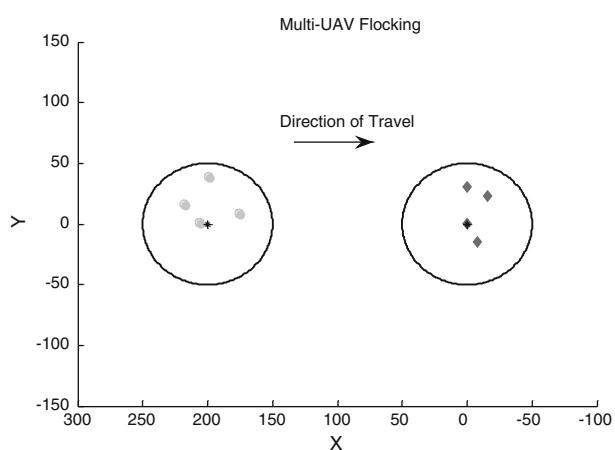


**Fig. 6** Distance from formation center over time

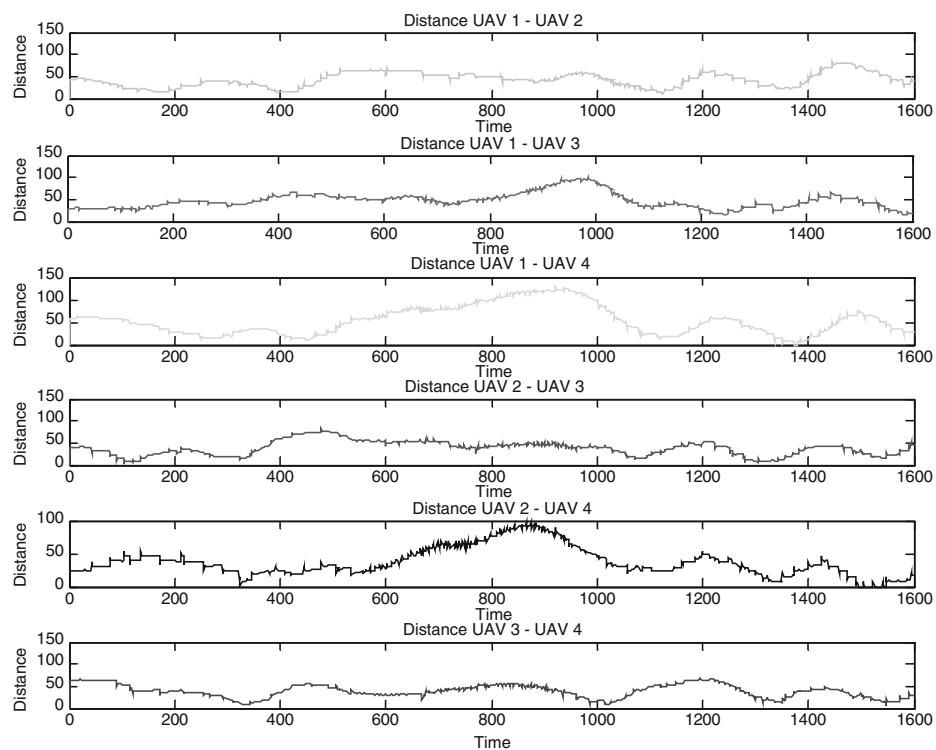


**Fig. 7** Distance between UAVs

**Fig. 8** Four UAVs demonstrating flocking behavior



the trajectory flocking inside the boundaries of the ellipse. The parameters utilized are shown in Table 1. Since we are not concerned with defining elliptical bands, only the  $R^*$  denoting the ellipse of interest is defined. Figure 9 shows that UAVs stayed dispersed while traversing the trajectory.



**Fig. 9** Distance between UAVs with flocking behavior

## 6 Conclusions and Future Work

This research provides the insight and justification for utilizing COTS simulation products for multi-vehicle control and coordination. It also shows how this type of system design is fundamentally applicable to distributed control and hardware in the loop simulation. This idea is proven viable with currently available software and hardware. Overall, the main advantages of this type of simulation environment are: access to a database of thousands of vehicle models, access to highly developed world models, ease of development for new vehicle models, highly detailed real-time graphical representations, and simplified research replication/validation.

One item of immediate future work includes deriving a nonlinear vector conversion for the potential field vector into the fuzzy controllers. This update will allow navigation control to be increased without creating noticeable sinusoidal errors when the vehicle is very close to its desired position. Long term future work includes distributed control, 3-D adaptable formation control, failure tolerance and urban path identification for ground vehicle support.

## References

1. Baldor, L.C.: U.S. use of UAVs in Iraq surges. In: Prompting Turf War, vol. 2009 (2007)
2. Cook, C.: Perspectives on Acquisition, Test, and Early Fielding of UAV Systems. Department of Operational Test and Evaluation (2008)
3. Kim, D.-M., Kim, D., Kim, J., Kim, N., Suk, J.: Development of near-real-time simulation environment for multiple UAVs. In: International Conference on Control, Automation and Systems, Seoul, Korea (2007)
4. Rasmussen, S.J., Chandler, P.R., Veridian, W.: MultiUAV: a multiple UAV simulation for investigation of cooperative control (2002)
5. Goktogan, A.H., Nettleton, E., Ridley, M., Sukkarieh, S.: Real time multi-UAV simulator. In: IEEE International Conference on Robotics and Automation (2003)
6. Beer, B.D., Lewis, M.: Lightweight UAV simulation for use in multi-agent human-in-the-loop experiments. In: Proceedings of the European Concurrent Engineering Conference, EUROSIS, pp. 51–56 (2007)
7. Xu, D., Borse, P., Grigsby, K., Nygard, K.E.: A petri net based software architecture for UAV simulation. In: Proceedings of Software Engineering Research and Practice (SERP04), pp. 227–232 (2004)
8. Ali, K., Carter, L.: Miniature-autopilot evaluation system. *J. Comput. Sci.* **4**, 30–35 (2008)
9. Garcia, R.D., Valavanis, K.P., Kandel, A.: Fuzzy logic based autonomous unmanned helicopter navigation with a tail rotor failure. In: 15th Mediterranean Conference on Control and Automation, Athens, Greece (2007)
10. Ertem, M.C.: An airborne synthetic vision system with HITS symbology using X-Plane for a head up display. In: Digital Avionics Systems Conference, DASC (2005)
11. “Downloads.” vol. 2009: X-Plane.Org (2009)
12. Garcia, R.D.: Designing an autonomous helicopter testbed: from conception through implementation. In: Computer Science Engineering, vol. Ph.D., p. 305. University of South Florida, Tampa (2008)
13. Barnes, L., Fields, M.A., Valavanis, K.: Unmanned ground vehicle swarm formation control using potential fields. In: 15th Mediterranean Conference on Control and Automation, pp. 1–8 (2007)
14. Barnes, L., Fields, M.A., Valavanis, K.: Heterogeneous swarm formation control using bivariate normal functions to generate potential fields. *International Transactions on Systems Science and Applications* **2**, 346–359 (2007)
15. Barnes, L., Garcia, R., Fields, M.A., Valavanis, K.: Adaptable Formations utilizing heterogeneous unmanned systems. In: SPIE Defense and Security Conference (2009)
16. Reddy, M., Iverson, L.: GeoVRML 1.1 - Concepts. Available from: <http://www.ai.sri.com/geovrml/1.1/doc/concepts.html> (2002)

# UAS Flight Simulation with Hardware-in-the-loop Testing and Vision Generation

Jeffery Saunders · Randal Beard

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 27 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** UAS flight simulation for research and development is a difficult problem because each airframe requires accurate physical models, control systems, an organized method of testing new control systems, virtual cameras for vision-based control, and methods of testing new control in the transition from simulation to flight tests. In an environment where researchers are temporary, such as a university, a standard research and development platform with these properties expedites prototyping and prevents code loss when an employee leaves. We develop a research simulation which conforms to all of these properties inside a Matlab environment. A series of mex functions provide connections to the autopilot for hardware-in-the-loop testing, graphical interfaces, and vision processing. The option to write C mex functions offers a seamless method of porting code to embedded systems, minimizing coding errors. We demonstrate fast prototyping by showing flight test data where the simulation provided virtual vision data to avoid virtual obstacles.

**Keywords** Flight simulation · Micro air vehicle · Unmanned air system · Matlab · Hardware in the loop

## 1 Introduction

Unmanned Air Systems (UASs) take on tasks to replace piloted aircraft in situations where human intervention is dangerous, too expensive, or infeasible. In military tasks, they remove danger to human pilots by flying surveillance, reconnaissance,

---

J. Saunders · R. Beard (✉)  
Electrical and Computer Engineering,  
Brigham Young University, Provo, UT 84602, USA  
e-mail: beard@byu.edu

J. Saunders  
e-mail: saunders.jeff@gmail.com

and attack missions without human involvement. Domestic missions include forest fire monitoring, search and rescue, and surveillance. Very often, small, relatively inexpensive UASs accomplish these tasks. UASs not only remove human risk, but lower the overall costs involved in mission tasks.

The high number of potential applications and the low cost of small UASs have kindled research and development in small autopilots. The decrease in size of the required sensors and integrated circuits have allowed the development of light-weight autopilots capable of navigating airframes with wingspans as small as 6 inches [1, 2]. The sensors available for large aircraft are not feasible on small UASs because of payload limitations. They require small sensors that are typically less accurate. Small UASs typically have fast dynamics, making accurate sensors even more important. While the technology of sensors improve, inner loop autopilot control continues to develop using alternative sensors such as computer vision [4]. Our objective is to create a simulation platform that facilitates development of these technologies.

The purpose of this paper is to describe a simulation environment that expedites prototyping by providing mechanisms to test new guidance, navigation, and control algorithms on various types of UASs, including vision-based control. The base simulation is simple, but offers enough features to aid in development of complex algorithms.

The University of Florida developed a simulation testbed for rapid vision processing [3]. Their goal was to accelerate the process of development of vision-based control inside simulation and smoothly transition the control to flight tests. However, their system required extensive hardware resources. Our goals are similar, but to do so on a single laptop computer. The objective is to create a simulation environment where vision-based control can be rapidly developed, tested with hardware in the loop, and rapidly transitioned to flight tests. We briefly discuss an application in which this was accomplished and describe the associated flight [6].

There are five requirements for the simulator:

1. Modularity to support different airframes and research,
2. Rapid prototyping,
3. Hardware-in-the-loop testing,
4. Graphical interface with camera simulation for vision processing, and
5. Code standardization to prevent code loss when employees leave.

Each of these requirements are discussed in the paper. Section 2 presents the structure of the simulation code. Section 3 discusses the physical subsystem and gives an example of a fixed-wing airframe. The guidance and navigation systems, including hardware-in-the-loop testing, is discussed in Section 4 followed by the graphical interface in Section 5. An example flight test in which we used hardware-in-the-loop vision processing is shown in Section 6 with a conclusion in Section 7.

## 2 Overview

A requirement of the simulator is modularity. At Brigham Young University, we have a wide variety of airframes, including fixed wing aircraft, quadrotors,

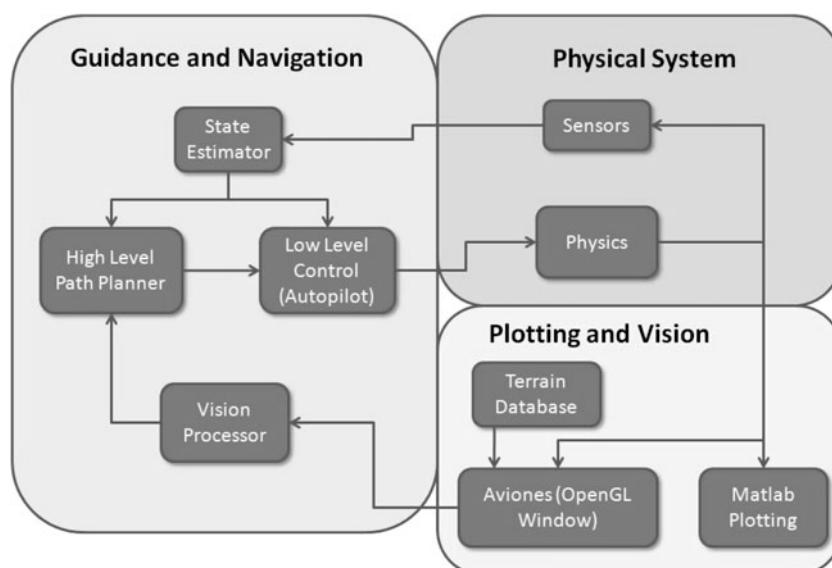
helicopters, and vertical take-off and land (VTOL) aircraft. We want to incorporate each of these different airframes into the simulator with minimal change, including physics and autopilot capabilities. The simulator is encoded in Matlab's Simulink.

Simulink has many beneficial characteristics for a simulator using multiple airframes and autopilots including:

1. A scripting interface for fast development,
2. A large set of engineering script libraries including control and computer vision,
3. A customizable library for simulation blocks that can be used for different airframes,
4. Simple data plotting functions,
5. An interface for C++ code (mex functions).

The block diagram of the simulation structure is shown in Fig. 1, which shows three subsystems. The first subsystem is the Guidance and Navigation System which includes the State Estimator, the High Level Path Planner, the Autopilot, and the Vision Processor required.

The second subsystem is the Physical System which includes the sensors onboard the UAS and the physics of the airframe. This subsystem is important when changing airframes. A fixed-wing aircraft has different flight characteristics than a quadrotor. To make this simulation universal, we store the physics engine of each airframe in the Simulink library. Each physics engine can be dragged from the Simulink library to the simulation.



**Fig. 1** The simulation is subdivided into three subsystems. Each subsystem contains Simulink blocks that can be interchanged for different autopilots, airframes, etc. as required

The third subsystem is Plotting and Vision which has two purposes. The first purpose is to display graphs, plots, and data as required. The second purpose is the graphical display called Aviones. Aviones displays the UAS in a terrain and simulates a camera for computer vision.

Our end goal is not to produce a good implementation in simulation, but to produce flight results from the theory using code developed in simulation. Many embedded systems are coded using C/C++. To implement an autopilot on a microprocessor, the Matlab script must be converted to C code. The conversion to C code also opens possibilities for bugs and problems that were not present in the script version. To aid in the transition, Matlab has a C/C++ interface called mex functions. The script can be written in C/C++ code and tested in the simulation environment without changing any other details. This limits the programming bugs and errors present in the embedded system.

In an educational environment, employment length is relatively short. In that time, employees implement their research on the current autopilot platform, or in a specialized simulation. The code is rarely reusable and often poorly documented. Code loss is prevented in a standard simulation platform and it provides a means of documentation. Each employee provides a section of code in a Simulink block that corresponds to their research, ideally using a script implementation and a mex function implementation. The state variables are identical for each code implementation, allowing each block to be added to other copies of the simulation. Each Simulink block includes a section for documentation. When a code implementation is required for another project, we use the Simulink block previously created. This standardized method minimizes code loss.

### 3 Physics

The Physics section simulates the physical system of the airframe. Matlab plays a key role in helping a researcher write and debug the code of the physical system. In the case of a fixed-wing aircraft, there are typically 12 vehicle states and 2 camera states which the simulator uses as a standard state vector. They are inertial position ( $p_n, p_n, p_d$ ), body frame velocity ( $u, v, w$ ), roll  $\phi$ , pitch  $\theta$ , yaw  $\psi$ , angular accelerations ( $p, q, r$ ), and the azimuth  $\alpha_a$  and elevation  $\alpha_{el}$  of the camera. The force equations we use are

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = mg \begin{pmatrix} -\sin \theta \\ \cos \theta \sin \phi \\ \cos \theta \cos \phi \end{pmatrix} + \frac{1}{2} \rho V_a^2 S \begin{pmatrix} C_X(\mathbf{x}, \delta) \\ C_Y(\mathbf{x}, \delta) \\ C_Z(\mathbf{x}, \delta) \end{pmatrix}, \quad (1)$$

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2} \rho V_a^2 S \begin{pmatrix} \frac{b}{2} C_X(\mathbf{x}, \delta) \\ \bar{c} C_Y(\mathbf{x}, \delta) \\ \frac{b}{2} C_Z(\mathbf{x}, \delta) \end{pmatrix}, \quad (2)$$

where  $C_*$  are functions to determine effective lift,  $\mathbf{x}$  is the states,  $\rho$  is the fluid density,  $S$  is the incidence area of the wing, and  $\delta$  is the surface deflections. The differential equations for a fixed-wing aircraft using those forces are

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_h \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta s\psi - s\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\theta \psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad (3)$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -g \sin \theta \\ g \cos \theta \sin \phi \\ g \cos \theta \cos \phi \end{pmatrix} + \frac{\rho V_a^2 S}{2m} \begin{pmatrix} C_X(\mathbf{x}, \delta) \\ C_Y(\mathbf{x}, \delta) \\ C_Z(\mathbf{x}, \delta) \end{pmatrix}, \quad (4)$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \quad (5)$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_4(p^2 - r^2) \\ \Gamma_6 pq - \Gamma_1 qr \end{pmatrix} + \frac{\rho V_a^2 S}{2} \begin{pmatrix} \frac{b}{2} (\Gamma_3 C_l(\mathbf{x}, \delta) + \Gamma_4 C_n(\mathbf{x}, \delta)) \\ \frac{\bar{c}}{J_y} C_m(\mathbf{x}, \delta) \\ \frac{b}{2} (\Gamma_4 C_l(\mathbf{x}, \delta) + \Gamma_7 C_n(\mathbf{x}, \delta)) \end{pmatrix}. \quad (6)$$

where  $\Gamma_*$  and  $J_y$  are functions of the inertia matrix, and  $C_n$ ,  $C_\ell$ ,  $C_n$  are moment coefficients,  $g$  is the gravitational constant, and  $m$  is the mass of the aircraft.

We implemented these equations in Matlab script for debugging purposes, and in a mex function to lower processing time. Parameters such as lift and drag coefficients are available in a Simulink block mask and they are changeable as necessary without changing code. Each airframe has a physics block in the Simulink library. The state vector is standard for each airframe, allowing each airframe physics to be used in different simulations.

## 4 Guidance and Navigation

The guidance and navigation section of the simulator includes the low level control, high level control, vision processing, and sensor filters.

### 4.1 Autopilot

The autopilot is the low level inner loops to maintain altitude, heading, roll, pitch, heading rate, etc. Commercial autopilots commonly use a series of PID loops for low level control of an aircraft [1]. Simulink can implement an autopilot using either a Matlab script or a C++ mex function. The scripting interface offers an excellent method to implement and test new ideas. Scripting requires no compilation, allows

breakpoints, and provides complex mathematical operations with simple commands. After theoretical analysis is complete, the Matlab script can be ported to a mex function before porting it to an autopilot. The mex function provides a method of testing C code in simulation to minimize errors before moving it to an autopilot system.

#### 4.2 High Level Path Planning

A variety of high level path planners can be developed. For example, a waypoint path planner creates waypoint paths that cause the UAV to fly a series of waypoints [5]. The high level planner provides roll, airspeed, and climb rate commands to the autopilot.

#### 4.3 Vision Processing

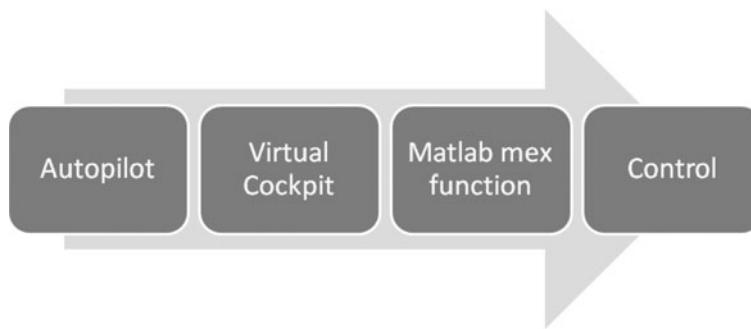
Matlab has a wide variety of vision processing functions available through its scripting interface. The vision processing Simulink block can use the Matlab vision processing functions for fast development of required vision processor. The Aviones graphics block simulates a camera and sends the pixel data to the vision processing block. The pixel data from Aviones is treated as pixel data from an actual camera, even adding noise is possible. Vision processing on the data is different for each application, but the development process is simplified by script implementation. The Matlab functions offer a method to check theory validity without C-code.

For flight, a vision processing implementation in C is usually required. To ease the process of converting from Matlab script, we use a mex function with OpenCV support. OpenCV is an open source vision processing library. OpenCV has many powerful vision functions. Using OpenCV in a mex function greatly reduces the time required in converting from script to C-code. The mex function is used in the same way as a script file, allowing the C implementation to be tested in the simulation before using it in a separate program.

#### 4.4 Hardware in the Loop

Before any flight, a control platform needs to be tested in an environment that is as close to actual flight as possible. We use the embedded autopilot with simulated physics to test the platform. In our case, the physics engine is also located on the autopilot. An RF modem transmits the telemetry data from the autopilot to the ground station. A computer program called Virtual Cockpit collects the data. A TCP/IP connection sends the telemetry data to other programs as required, including Matlab through a mex function. The simulation can use the telemetry information for any hardware-in-the-loop testing required, such as vision simulation with Aviones and vision processing with the script interface or OpenCV. Running hardware-in-the-loop using Matlab creates a vision simulation that normally isn't available for hardware-in-the-loop. This option speeds up the development cycle (Fig. 2).

The hardware-in-the-loop mechanism is also available for flight tests. While another program can be made for monitoring and vision processing to separate simulation from hardware-in-the-loop and flight tests, the option of leaving control



**Fig. 2** Telemetry information flows from the autopilot to Virtual Cockpit via an RF modem, to Simulink via a TCP/IP connection

in Matlab speeds up development. Using Matlab for vision processing and control provides an option of using simulated vision during a flight test, increasing safety and allowing observance of flight characteristics in the control loops.

## 5 Graphical Interface

### 5.1 Matlab Plotting

Matlab has a large set of plotting functions useful for debugging, tuning, and evaluation. Debugging and plotting routines are written by employees. Many of the routines are useful and added to the Simulink library for use in other projects.

### 5.2 Aviones

We use another plotting interface called Aviones which was developed at BYU. Aviones is an OpenGL 3D environment that uses terrain elevation maps overlayed by satellite images of the terrain. The result is a virtual terrain matching any terrain for which elevation data is available. An example is shown in Fig. 3a. This environment is available in Matlab through a mex function. The first call to the function initializes the window with the desired size, frame rate, terrain, etc. Another call to the function updates the state of the UAS. The result is simulated video of the UAS moving through the terrain.

The viewpoint in Aviones is adjustable. There are four viewpoints available: chase, camera, satellite, and groundtrack. The chase view is a viewpoint one meter behind the UAS with the same yaw angle as the UAS. Figure 3a shows the chase view. The satellite view is simply a high altitude looking down on the terrain. The groundtrack views the UAS from the launch position. The most useful view for vision directed control is the camera view. The camera view simulates a camera at the elevation and azimuth angles given in the state vector. The pixel data from the camera view is sent to the video processing Simulink block for vision processing.



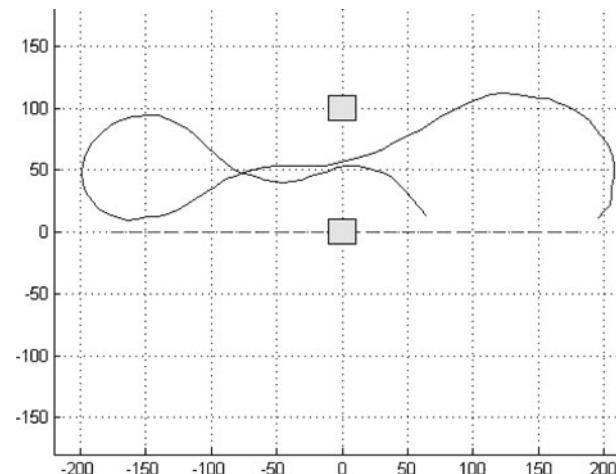
**Fig. 3** Aviones generates viewpoints at various positions. The pixel data from any of them can be used for vision processing. **a** UAS chase view. **b** UAS camera view

## 6 Flight Tests

We conduct flight tests on a regular basis. Some of these flight tests involve our obstacle avoidance algorithms. Obstacle avoidance is dangerous to test. If an obstacle is not detected properly, the airframe may crash into the obstacle, possibly damaging the airframe and the obstacle. In addition, most obstacles are close to the ground, requiring the UAS to fly at low altitudes and relying on accurate altitude measurements to prevent it from colliding with the ground. These dangers can be alleviated by testing first using hardware-in-the-loop vision provided by the simulator described in this paper. The graphical interface section generates camera data, Matlab processes it, and sends control commands to the autopilot in flight. This allows the UAS to fly at high altitudes, in an obstacle free area, while testing obstacle avoidance routines. If performance is adequate, then a flight test with real obstacles may be warranted.

Obstacle avoidance with hardware-in-the-loop vision has been conducted many times with this simulation architecture. An example is shown in Fig. 4. In this

**Fig. 4** The UAS avoids virtual obstacles using virtual vision with the hardware-in-the-loop simulation. The *solid line* is the UAS path, the *dashed line* is the waypoint path, and the *yellow blocks* are obstacles



example, we use the obstacle avoidance routines described in Ref [6]. As the vision processing detects the obstacles, it moves the obstacles to the edge of the camera field-of-view to avoid them. For two obstacles, each obstacle is moved to opposite sides of the camera field-of-view, thus the UAS maneuvers between the obstacles. The flight was conducted safely at an altitude of 100 meters in an obstacle free area. This flight test allowed us to monitor flight performance, tune gains, and determine if a low altitude test with real obstacles is feasible. All control code was written in Matlab script, unchanged from the purely simulated tests.

## 7 Conclusion

In this paper, we have described a simulator with the following properties,

1. Modularity to support different airframes and research,
2. Rapid prototyping,
3. Hardware-in-the-loop testing,
4. Graphical interface with camera simulation for vision processing,
5. Code standardization to prevent code loss when employees leave,
6. Support for flight testing with vision generation.

The simulator is a significant factor in fast development and prototyping of new research. It contributes to safe flight tests in situations where flight tests are dangerous by allowing testing at high altitudes and generated data. It also provides standardization.

## References

1. Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., Goodrich, M.A.: Autonomous vehicle technologies for small fixed-wing UAVs. *J. Aerosp. Comput. Inform. Commun.* **2**, 92–102 (2005)
2. Grasmeyer, J.M., Keenon, M.T.: Development of the black widow micro air vehicle. In: 39th AIAA Aerospace Sciences Meeting and Exhibit (AIAA Paper No. 2001-0127) (2001)
3. Grzywna, J.W., Jain, A., Plew, J., Nechyba, M.C.: Rapid development of vision-based control for MAVs through a virtual flight testbed. In: International Conference on Robotics and Automation (2005)
4. Kaiser, K., Gans, N., Dixon, W.: Localization and control of an aerial vehicle through chained, vision-based pose reconstruction. In: American Control Conference, pp. 5934–5939 (2007)
5. Nelson, D.R., Barber, B., McLain, T.W., Beard, R.W.: Vector field path following for miniature air vehicles. *IEEE Trans. Robot.* **23**, 519–529 (2007)
6. Saunders, J., Beard, R.: Reactive vision based obstacle avoidance with camera field of view constraints. In: Guidance, Navigation, and Control Conference (2008)

## Multi-UAV Cooperation and Control for Load Transportation and Deployment

I. Maza · K. Kondak · M. Bernard · A. Ollero

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 13 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper deals with the cooperation and control of multiple UAVs with sensing and actuation capabilities. An architecture to perform cooperative missions with a multi-UAV platform is presented. The interactions between UAVs are not only information exchanges but also physical couplings required to cooperate in the joint transportation of a single load. Then, the paper also presents the control system for the transportation of a slung load by means of one or several helicopters. Experimental results of the load transportation system with one and three helicopters are shown. On the other hand, the UAVs considered in the platform can also deploy small objects, such as sensor nodes, on different locations if it is required. This feature along with the whole platform architecture are illustrated in the paper with a real multi-UAV mission for the deployment of sensor nodes to repair the connectivity of a wireless sensor network.

---

This work was partially funded by the European Union AWARE Project (IST-2006-33579), the CONET Network of Excellence (INFSO-ICT-224053) and the ROBAIR Project of the Spanish Research and Development Program (DPI2008-03847).

I. Maza (✉) · A. Ollero  
Robotics, Vision and Control Group, University of Seville, 41092 Seville, Spain  
e-mail: imaza@cartuja.us.es

A. Ollero  
e-mail: aollero@cartuja.us.es

K. Kondak · M. Bernard  
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany

K. Kondak  
e-mail: kondak@cs.tu-berlin.de

M. Bernard  
e-mail: bernard@cs.tu-berlin.de

A. Ollero  
Center for Advanced Aerospace Technology (CATEC), Seville, Spain  
e-mail: aollero@catec.aero

**Keywords** Multiple UAVs · Multi-UAV load transportation · Sensor deployment · Multi-UAV distributed decision

## 1 Introduction

The progress on miniaturization technologies, together with new sensors, embedded control systems and communication, have fuelled the development of many new small and relatively low cost Unmanned Aerial Vehicles (UAVs). However, constraints such as power consumption, weight and size plays an important role in UAVs, and particularly in small size, light and low cost UAVs. Then, the cooperation of many of these vehicles is the most suitable approach for many applications. A single powerful aerial vehicle equipped with a large array of different sensors of different modalities is limited at any one time to a single view point. However, a team of aerial vehicles can simultaneously collect information from multiple locations and exploit the information derived from multiple disparate points to build models that can be used to take decisions. Team members can exchange sensor information, collaborate to track and identify targets and perform detection and monitoring activities among other tasks [19]. Thus, for example, a team of aerial vehicles can be used for exploration [16], detection, precise localization, monitoring and measuring the evolution of natural disasters, such as forest fires. Furthermore, the multi-UAV approach leads to redundant solutions offering greater fault tolerance and flexibility.

Unmanned aerial vehicles are mainly concerned with environment sensing and perception activities, with limited ability to react against environment changes. The above mentioned surveillance and detection applications lie in this category. If the environment is considered known and static, or quasi static with low dynamic behaviour when comparing with the UAVs motion, then the coordination problem can be solved before the real-time execution of the mission. Then, the multi-UAV mission planning, task allocation and trajectory generation could be done before the mission execution. However, environment perturbations (e.g. in a disaster management scenario) or a lack of environmental data require real-time coordination efforts. Furthermore, applications in non segregated aerial spaces where other non-cooperative aircrafts may operate, also require collision detection and avoidance capabilities.

Other applications, such as tracking the motion of a mobile ground target, require the real-time computation of the trajectory from environment perception, and, in general, the closing of real-time control loops by means of environment perception in the aerial robot [20]. The same happens when considering the monitoring of dynamic events such as forest fires. In this case, the application of real-time coordination and cooperation methods is required.

However, notice that, up to now, in all the above mentioned applications, the aerial robots, as many other mobile robots, are mainly considered as platforms for environment sensing. Then, the aerial robots do not modify the state of the environment and there are no physical interactions between the UAV and the environment. Furthermore, the interactions between the UAVs are essentially information exchanges, without physical couplings between them.

This paper deals with the cooperation and control of multiple aerial robots with sensing and actuation capabilities for the deployment of loads, and particularly,

sensor nodes. Furthermore, the paper considers the multi-UAV load transportation when an object is too heavy to be carried by one single UAV. The latter requires the consideration of physical interactions between the aerial robots.

On the other hand, Wireless Sensor Network (WSN) technologies have experienced an important development in the last ten years. Research work includes node mobility using robots and the navigation of mobile robots by using wireless sensor network information (see survey in [1]). Connectivity is a main issue in wireless sensor networks. Then, the application of robots for the deployment and connectivity repair of wireless sensor networks has been also proposed. In [6] the application of an UAV for deployment and connectivity repair of a wireless sensor network is proposed and demonstrated by using a single autonomous helicopter. In this paper, the deployment and connectivity repair of a wireless sensor network in a multi-UAV context is proposed and shown experimentally.

The transportation of a load by several ground robots has been an active subject of research and development for many years. The coordinated control of the motion of the vehicles should consider the involved forces. Thus, each robot could be controlled around a common compliance center attached to the transported object. Under the assumption that each robot holds the object firmly, the real trajectories of all of the robots are equal to the real trajectory of the object. Both centralized and decentralized compliant motion control algorithms have been proposed, including the consideration of non-holonomic constraints [14]. The method has been implemented in an experimental system with three tracked mobile robots with a force sensor. In [23] the decentralized control of cooperating mobile manipulators is studied with a designated lead robot being responsible for task planning. The control of each robot is decomposed (mechanically decoupled) into the control of the gross trajectory and the control of the grasp. The excessive forces due to robot positioning errors and odometry errors are accommodated by the compliant arms. In [4] the Omminate system which uses a compliant linkage platform between two differential drive mobile robots is presented. In [9] distributed coordinated control of two rovers carrying a 2.5 m long mockup of a photovoltaic tent is presented and demonstrated as an example of the CAMPOUT behavior-based control architecture.

The transportation of a single load by means of several helicopters has been also proposed in the literature. In experiments with two manned helicopters it was determined that the piloting of two coupled helicopters is a very challenging task, so at least, an automatic stabilization of the helicopters is required. The motivation for using two or more small helicopters instead of one with bigger load capacity are:

- Like in the case of real manned transport helicopters, the costs for two small helicopters are often less than for one with double load capacity.
- Whatever load capacity an existing helicopter has, there will be always a task where a higher load capacity is required. In this case the control software could allow to couple existing helicopters in order to form a system with sufficient load capacity.

Particularly, research on lifting and transportation of loads by means of two helicopters (twin lift) was presented in e.g. [18, 21]. However, this research work has been done only in simulation. In this paper real flight experiments for load transportation with one and three helicopters are presented. We describe an automatic control system which allows the usage of one or multiple small size helicopters for slung

load transportation. The number of helicopters should be configurable depending on helicopters' capabilities and the load to be transported.

The research and developments presented in this paper have been conducted in the framework of the AWARE project. The general objective of this project is the design, development and experimentation of a platform providing the middleware and the functionalities required for the cooperation among UAVs and a ground sensor-actuator wireless network, including mobile nodes. The platform will enable the operation in sites with difficult access and without communication infrastructure. Then, the project considers the self-deploying of the network by means of autonomous helicopters with the ability to transport and deploy loads (communication equipment and nodes of the ground network).

This paper is organized as follows. Section 2 presents the multi-UAV distributed architecture developed for the AWARE platform, that allows different levels of cooperation and coordination among the UAVs and between the UAVs and the environment, including both sensing and actuation. Section 3 is devoted to the transportation of a single load by means of multiple joined helicopters, and includes experiments with one and three helicopters. Section 4 presents the multi-UAV deployment of multiple loads and particularly the deployment of the sensor nodes of a wireless sensor network for connectivity repairing. The last sections are devoted to the Conclusions and References.

## 2 Architecture for Aerial Robots Cooperation in the AWARE Platform

This section addresses specifically the architectural aspects linked to the necessary coordination and cooperation issues induced by the operation of several AWARE subsystems, such as the aerial robots.

A global mission for the AWARE platform is specified by the user using the Human Machine Interface (HMI) software. Each mission  $\mathcal{M}$  will consist in a set of *tasks* (possibly ordered) that must be executed by the platform. The task allocation process to the different UAVs could be done by the user or may be autonomously performed in a distributed way. The latter might be necessary in situations where large numbers of UAVs have to interact, where direct communication with a central station is not possible, or where the local dynamics of the situation require timely reaction of the UAVs involved in it.

The tasks that will be executed by the AWARE platform involve coordination, mainly for sharing space, and cooperation, for example for the surveillance at different altitudes or from different viewpoints of the same object, or when an UAV plays the role of a radio re-transmitter from/to other UAVs and the central station. Cooperation includes coordination, but there is role sharing between the subsystems to achieve a global common task.

In the following, a general framework that addresses the operation of the AWARE multi-UAV platform is described. Let us consider a team of UAVs that plan their motions according to a set of decentralized and cooperative rules  $\mathcal{R}$ . In particular, we assume that the set  $\mathcal{R}$  defines  $k$  possible *tasks*  $T = \{\tau_1, \tau_2, \dots, \tau_k\}$  that UAVs can perform, and specifies  $n$  *logical conditions* requiring a change of task in the current plan. Let  $E = \{e^1, e^2, \dots, e^n\}$  be a set of discrete events associated with

such conditions. Each task has a set of  $m$  parameters  $\Pi = \{\pi^1, \pi^2, \dots, \pi^m\}$  defining its particular characteristics.

Aerial robotic systems composed of a physical plant and a decisional and control system implementing such kind of cooperation rules  $\mathcal{R}$  can be modeled as hybrid systems [5, 8, 15]. The general components of a simplified hybrid model  $\mathcal{H}$  are explained in the following. Let  $q_i \in \mathcal{Q}$  be a vector describing the state of the  $i$ -th UAV in the configuration space  $\mathcal{Q}$ , and let  $\tau \in T$  be the task that the UAV is currently executing. The  $i$ -th UAV's configuration  $q_i$  has a continuous dynamics

$$\dot{q}_i = f(q_i, c_i, \gamma_i), \quad (1)$$

where  $c_i \in \mathcal{C}$  are control inputs and  $\gamma_i \in \Gamma$  models the influence of the possible physical coupling with other UAVs and transported objects.

The  $i$ -th UAV's current task has a discrete dynamics  $\delta : T \times E \rightarrow T$ , i.e.

$$\tau^+ = \delta(\tau, e), \quad (2)$$

where  $e \in E$  is an event (internal or external) requiring a change of task from  $\tau$  to  $\tau^+$ , both from the set of tasks  $T$ .

Event activation is generated by

$$e = \mathcal{A}(q_i, \varepsilon, X, \bar{\mu}), \quad (3)$$

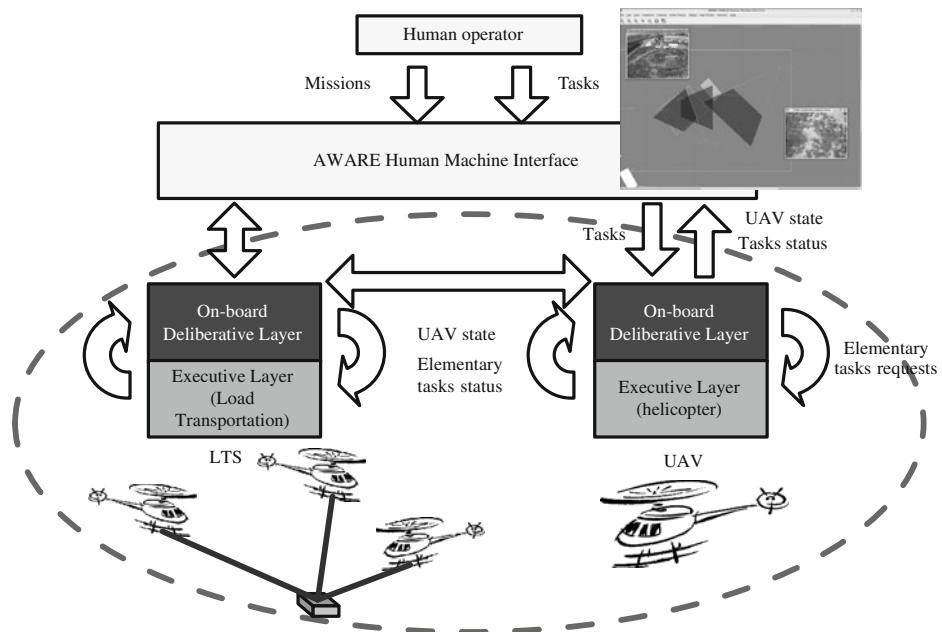
where  $\varepsilon$  represents the internal events (such as changes in the execution states of the tasks),  $X$  is a vector with information about external events in the environment and  $\bar{\mu}$  is a vector  $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_{N_m})$  containing the messages coming from  $N_m$  UAVs cooperating with the  $i$ -th UAV. Those messages are used for example in the negotiation processes involved in the intentional cooperation mechanisms and are generated in each robot by a decisional module  $\mathcal{D}$ . This module encompasses high level reasoning and planning, synchronization among different robots, negotiation protocols for task allocation and conflict resolution purposes, task management and supervision, complex task decomposition, etc.

In the following section, the particular details of the architecture adopted for the AWARE multi-UAV platform are described.

## 2.1 Architecture

The main objective in the design of the architecture was to impose few requirements to the execution capabilities of the autonomous vehicles to be integrated in the platform. Basically, those vehicles should be able to move to a given location and activate their payload when required. Then, autonomous vehicles from different manufacturers and research groups can be integrated in the AWARE architecture easily.

The global picture of the AWARE distributed UAV system is shown in Fig. 1. In each UAV, there are two main layers: the On-board Deliberative Layer (ODL) and the proprietary Executive Layer. The former deals with high-level distributed decision-making whereas the latter is in charge of the execution of the tasks. In the interface between both layers, the ODL sends task requests and receives the execution state of each task and the UAV state. For distributed decision-making purposes, interactions among the ODLs of different UAVs are required. Finally, the HMI software allows the user to specify the missions and tasks to be executed by the



**Fig. 1** Global overview of the distributed UAV system architecture

platform, and also to monitor the execution state of the tasks and the status of the different sub-systems of AWARE.

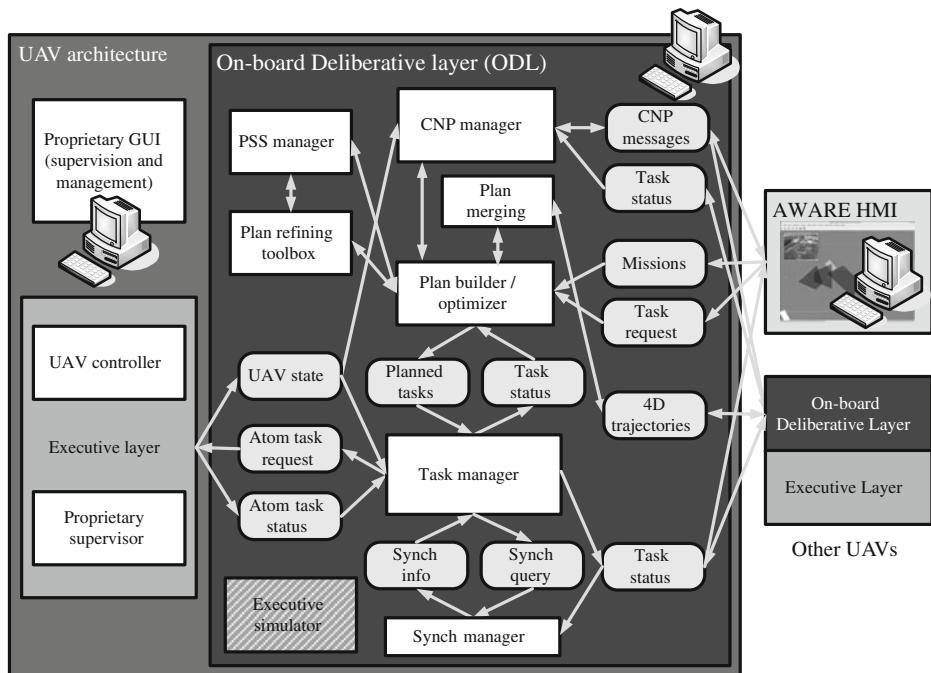
A more detailed view of the Deliberative Layer architecture is shown in Fig. 2. As it has been mentioned above, the ODL has interactions with its executive layer and with the ODL of other UAVs as well as with the HMI. The different modules shown in the ODL supports the distributed decision-making process involving cooperation and coordination. After presenting the task model in the next section, the operation of each module will be detailed.

## 2.2 Task Model

Let us consider a mission  $\mathcal{M}$  specified by the AWARE platform user. This mission is decomposed (autonomously or manually) in a set of partially ordered tasks  $\mathcal{T}$ . Those tasks can be allocated to the UAVs manually from the HMI or autonomously in a distributed way (see Section 2.5). Let us define a task with unique identifier  $k$  and type  $\lambda_k$  allocated to an UAV  $i$  as  $\tau_k^i = (\lambda_k, \neg\Omega_k, \Omega_k^+, \varepsilon_k, \Pi_k)$ , where  $\neg\Omega_k$  and  $\Omega_k^+$  are respectively the set of preconditions and postconditions of the task, and  $\varepsilon_k$  is the event associated to the task evolution (see Table 1). Finally,  $\Pi_k = \{\pi_k^1, \pi_k^2, \dots, \pi_k^m\}$  is the set of  $m$  parameters which characterizes the task. As an example, see Table 2, which shows the parameters considered in a task consisting in covering a given area for surveillance.

Regarding the type of task ( $\lambda_k$ ) at the ODL level of the architecture, the list shown in Table 3 has been considered in the AWARE platform.

On the other hand, preconditions and postconditions are event-based mechanisms that can deal with events related to the evolution of the tasks states (see Table 1), to



**Fig. 2** Detailed view of the internal On-board Deliberative Layer (ODL) architecture

the reception of messages, to the detection of a given event by the Perception System (see Section 2.8), the elapsing of a certain time period, etc. Then, the execution of a task starts when all the associated preconditions are satisfied. On the other hand, it is also possible to specify postconditions, i.e. conditions which satisfaction triggers the abortion of a task. If a task does not have any precondition or postcondition, then  $\tau_k^i = (\lambda_k, \neg \Omega_k = \emptyset, \Omega_k^+ = \emptyset, \varepsilon_k, \Pi_k)$ .

An example of a precondition or a postcondition related to the evolution of the tasks is the “end of task” event of a different task. Furthermore, thanks to

**Table 1** Possible events considered in the status evolution of a task  $\tau_k^i$

Event ( $\varepsilon_k$ )	Description
EMPTY	No task
SCHEDULED	The task is waiting to be executed
RUNNING	The task is in execution
CHECKING	The task is being checked against inconsistencies and static obstacles
MERGING	The task is in the plan merging process to avoid conflicts with the trajectories of other UAVs
ABORTING	The task is in process to be aborted. If it is finally aborted, the status will change to ABORTED, and otherwise will return to RUNNING.
ABORTED	The task has been aborted (the human operator has aborted it or the UAV was not able to accomplish the task properly)
ENDED	The task has been accomplished properly

**Table 2** Parameters of a task with type  $\lambda_k = \text{SURV}$ 

Parameters ( $\Pi_k$ )	Description
$\pi^1$ (Polygon)	The set of vertices defining the polygon of the area to be covered by the UAV
$\pi^2$ (Altitude)	Altitude (m) ellipsoid-based datum WGS84 for the flight
$\pi^3$ (Speed)	Desired speed (m/s) for the flight

the synchronization manager module (see Section 2.3), it is possible to specify preconditions between tasks of different UAVs. Finally, it should be mentioned that perception events (not related to the execution of a task) such as the detection of a fire or a fireman in a disaster scenario, could be also the precondition of a task (i.e. a tracking task).

The ODL processes the tasks received and generates simpler tasks, called elementary tasks, that are finally sent to the executive layer of the UAV. Let us define an elementary task with unique identifier  $k$  and type  $\hat{\lambda}_k$  allocated to the  $i$ -th UAV as  $\hat{\tau}_k^i = (\hat{\lambda}_k, \hat{\Pi}_k, \hat{\epsilon}_k)$  where  $\hat{\Pi}_k = \{\hat{\pi}_k^1, \hat{\pi}_k^2, \dots, \hat{\pi}_k^m\}$  is the set of  $\hat{m}$  parameters which characterizes the elementary task and  $\hat{\epsilon}_k$  is the event associated to the elementary task evolution. It should be mentioned that RUNNING and ENDED are the only events considered for the elementary tasks. On the other hand, as an example, Table 4 shows the seven parameters that are considered in an elementary task consisting in visiting a given location.

Then, the vehicles to be integrated in the AWARE platform should be able to receive elementary tasks, report their associated execution events and execute them. A small set of elementary tasks have been considered in order to allow the integration of a broader number of vehicles from different manufacturers and research groups. Basically, those vehicles should be able to move to a given location and activate their payload when required.

In the next section, the module in charge of the management of the tasks following the model presented above is described.

**Table 3** Several type of tasks ( $\lambda_k$ ) considered at the ODL level

Type of task ( $\lambda_k$ )	Description
TAKE-OFF	The UAV takes off and stabilizes at a default safe height, then switches to a secured wait mode, waiting for further instructions.
LAND	The UAV starts landing procedures, lands, and is set to a ground secured mode.
GOTO	The UAV moves from its current location to a point P (or to its vicinity).
GOTOLIST	The UAV moves from its current location to each of the points of the waypoints list, following the order of the points.
WAIT	The UAV is set to a secured waiting mode: hover or pseudo-hover, during a given period.
SURV	The UAV covers a given area defined by a polygon at a certain altitude.
DETECT	The perception system of the UAV starts to operate in detection mode, providing an alert if a given event in the environment (fire, persons, etc.) appears.
TRACK	The perception system of the UAV starts to operate in tracking mode, providing (if possible) estimations of a given event location (fire, persons, etc.). The UAV moves accordingly to follow the event.
HOME	The UAV is commanded to return home.

**Table 4** Elementary task with type  $\hat{\lambda}$  = GOTO: list of parameters

Parameters ( $\hat{\Pi}$ )	Description
$\hat{\pi}^1$ (Latitude)	Latitude ellipsoid-based datum WGS84
$\hat{\pi}^2$ (Longitude)	Longitude ellipsoid-based datum WGS84
$\hat{\pi}^3$ (Altitude)	Altitude (m) ellipsoid-based datum WGS84
$\hat{\pi}^4$ (Speed)	Desired speed (m/s) along the way to the waypoint
$\hat{\pi}^5$ (Force_heading)	1: force to the specified heading, 0: Not force
$\hat{\pi}^6$ (Heading)	Desired heading (degree) along the way (N is 0°, E is 90°, W is -90° and S is 180°)
$\hat{\pi}^7$ (Payload)	1: to activate the payload around the location of the waypoint, 0: not to activate

### 2.3 Task and Synchronization Managers

The task manager module (see Fig. 2) receives the planned tasks from the plan builder module. Those tasks can have preconditions and/or postconditions and the task model assumed is based on elementary events processing, which are expected to occur whenever the states of the tasks and the environment evolve. The starting event is the only controllable event: all other kind of events related to a task are contingent, i.e. the system can neither guarantee that such an event will occur nor when exactly it may occur.

In each task request for the task manager, the operation to be applied should be one of the following two alternatives:

- Dynamic task insertion (**INSERT** operation): this allows to request tasks insertion in the UAV's current plan, according to the relative order specified for the newly inserted task, versus the current partial order of the tasks already scheduled. It allows to insert a task with preconditions and/or postconditions. Preconditions can be specified either as mandatory or optional. If it is mandatory and the precondition happens not to be satisfiable anymore, then the task is aborted. On the contrary, if it is specified as optional, the precondition is considered as satisfied (and hence removed from the task's list of preconditions) if it is actually satisfied or if its own satisfiability becomes unsatisfiable (and in this case the task is not aborted). An example of task precondition is the “end of task” event of a different task. On the other hand, it is also possible to specify postconditions, i.e. conditions which satisfaction triggers the abortion of a task. For example, it allows to interrupt a given task when another one is achieved: during a surveillance task, once a sequence of waypoints covering an area have been visited, we might want to interrupt also the **DETECT** task being carried out by the Perception Sub-System (PSS) on-board.
- Dynamic task abortion (**ABORT** operation): this mechanism allows to dynamically request task abortion in the current plan, while the plan is being executed. If the task is already running, then the abortion of the task is an interruption. If the task is not yet running, then the abortion is a cancellation (the task is de-scheduled). The abortion triggers a propagation mechanisms, that checks which of the scheduled tasks depends on the aborted task (i.e. the tasks having a precondition expecting an event from the aborted task, like an “end of task”

event): if the dependence is a mandatory precondition, then this task is also aborted and so on. If it is an optional precondition, then the dependence is removed as if the precondition was satisfied, and the corresponding task is not aborted.

On the other hand, the task manager also interfaces with the executive layer of the UAV. It sends elementary tasks to the executive, which reports the state of both those tasks and the UAV itself.

The management of the preconditions and postconditions is carried out together with the task synchronization module. In the interface between both, there is a simple protocol based on messages. The synchronization manager is in charge of keeping the dependencies coherent among the different tasks in the current plan of the UAV, and also with the tasks of other UAVs.

When a new task request with preconditions and/or postconditions arrives to the task manager module, it sends a REPORT\_DEPS message with the dependencies to the synchronization manager, which saves it in a local database.

The synchronization manager is always checking the state of the tasks in the distributed UAV system and, if there is any change, it updates the dependencies database. For instance, if a given task changes its status to ENDED and if this event is precondition for other tasks, those preconditions are changed to satisfied. On the other hand, if all the postconditions of a task are satisfied, an INFO message is sent to the task manager module that will proceed with the corresponding task abortion.

Before requesting an elementary task to the executive layer, the task manager sends a QUERY message with the sequence number of the task to the synchronization manager. The satisfiability is checked and the answer is sent back with an INFO message. If all the preconditions are not satisfied, the task manager will ask again periodically.

## 2.4 Plan Builder / Optimizer

In the plan building stage of the AWARE platform, there are two different possibilities:

- Offline planning: previous to the mission execution, the AWARE platform user can plan a mission to tune its parameters and check its feasibility using the EUROPA framework developed at NASA's Ames Research Center.
- Online planning: it is based on a plan builder / optimizer module integrated in the ODL architecture (see Fig. 2) and programmed to solve the specific planning problems of the UAVs in the AWARE platform.

Both options are described in the next subsections.

### 2.4.1 Plan Builder / Optimizer: Offline Planning

The EUROPA framework developed at NASA's Ames Research Center is available under NASA's open source agreement (NOSA) since 2007. NOSA is an OSI-approved software license accepted as open source but not free software. EUROPA (Extensible Universal Remote Operations Planning Architecture) is a class library and tool set for building planners (and/or schedulers) within a Constraint-based Temporal Planning paradigm and it is typically embedded in a host application.

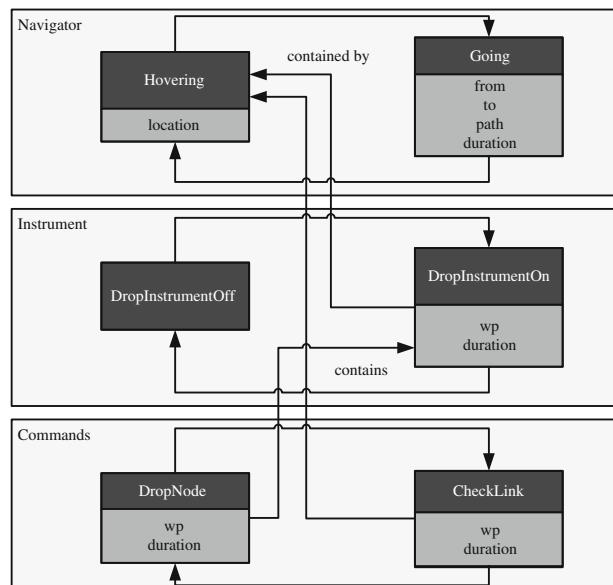
Constraint-based Temporal Planning (and Scheduling) is a paradigm of planning based on an explicit notion of time and a deep commitment to a constraint-based formulation of planning problems. This paradigm has been successfully applied in a wide range of practical planning problems and has a legacy of success in NASA applications.

As a simple application example in the AWARE project context, a deployment mission for an autonomous helicopter will be considered in the following. EUROPA is used offline before the mission execution to tune the parameters of the tasks and test its feasibility. The particular application domain model and problem definition has to be provided to the planner, which will generate a plan to solve the problem. In our example, the main entity application domain is the helicopter. The next decision is to identify the entities that will describe changes in the state of the helicopter as it moves around the environment performing a mission. Each entity is called a *timeline* in the EUROPA framework. Then, the following stage is to identify the states (called *predicates*) in which each timeline can be. Figure 3 shows the set of predicates identified on each timeline along with the variables in each state.

Analyzing the components of the helicopter produces the following breakdown of timelines (see Fig. 3) and states:

- Navigator:** controls the motion of the helicopter between locations and hovers at a location. States: The helicopter can be *hovering* at a location or *going* between locations.
- Instrument:** controls the instrument for dropping sensor nodes. States: the instrument can be *on* or *off*.
- Commands:** manages instructions from the HMI and checks the communication link after dropping a node. States: the helicopter can be instructed

**Fig. 3** Timelines and Predicates with Transitions between Predicates on each Timeline



to drop a node and has to check the communication link in order to evaluate if the sensor network connectivity is repaired.

Then, the application domain description is encoded in NDDL (an acronym for New Domain Description Language) along with its initial state. The locations of the waypoints where the sensor nodes should be dropped, the initial location and battery level of the UAV and the different paths (with their associated costs computed by the plan refining toolbox) between the locations of interest are specified.

Finally, let consider that the goal of the mission is to deploy three sensor nodes in different waypoints:

- wp1 at time 30.
- wp4 at time 60.
- wp3 at time 90.

It can be also easily encoded in NDDL in the following way:

```
goal(Commands.DropNode drop_node_1);
drop_node_1.start.specify(30);
drop_node_1.wp.specify(wp1);

goal(Commands.DropNode drop_node_2);
drop_node_2.start.specify(60);
drop_node_2.wp.specify(wp4);

goal(Commands.DropNode drop_node_0);
drop_node_0.start.specify(90);
drop_node_0.wp.specify(wp3);
```

With the model and the initial state specified, the planner can be started to compute the solution. Figure 4 shows a screenshot with the visualization of the results obtained with PlanWorks.

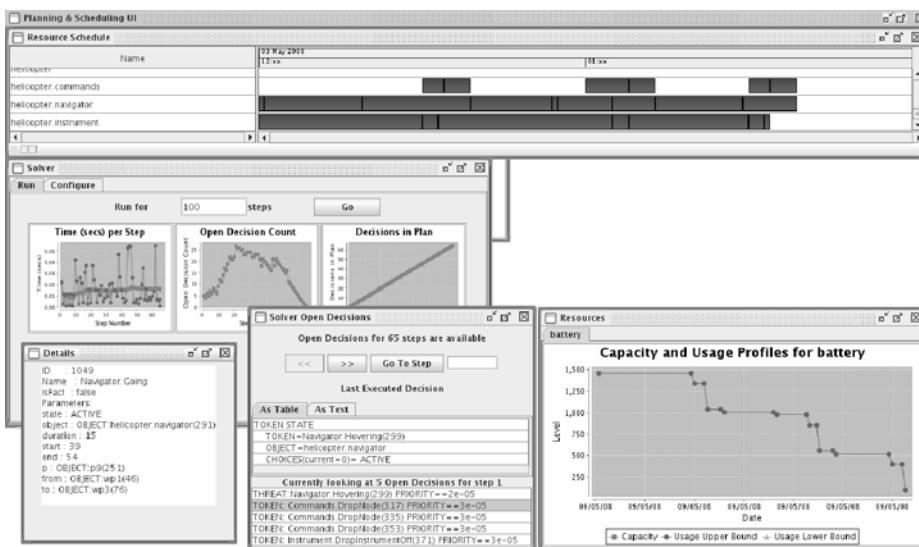
From the solution obtained, the AWARE platform user can check the feasibility of the mission and tune its parameters properly.

#### 2.4.2 Plan Builder/Optimizer: Online Planning

In general, the plan builder/optimizer module running during the mission execution generates a plan  $\mathcal{P}$  as a set of partially ordered tasks. In the AWARE platform, the main function of the online planner will consist in ordering the motion tasks allocated to the UAV. Let us consider the  $i$ -th UAV with a set of  $n_m$  motion tasks to be executed. The planner will compute the order of the tasks  $\{\tau_k^i / k = 1 \dots n_m\}$  that minimizes the execution cost:

$$C^i = \sum_{k=1}^{n_m-1} c_{k,k+1}^i \quad (4)$$

where  $c_{k,k+1}^i$  is the motion cost between the locations associated to the tasks  $\tau_k^i$  and  $\tau_{k+1}^i$ . This problem is an instance of the *Travelling Salesmen Problem*, often referred as *TSP*, which is NP-hard. The simplest exact algorithm to solve it is based on a brute



**Fig. 4** Screenshot with the visualization of the results obtained with PlanWorks

force search that tries all the ordered combinations of tasks. The running time for this approach lies within a polynomial factor of  $\mathcal{O}((n_m - 1)!)$ , so this solution is only feasible when a small number of tasks are allocated to the UAVs, which is the case in most of the AWARE platform missions.

But, if the user delegates the allocation process of tasks to the ODLs, each UAV will have to run many times the planning algorithm during the autonomous negotiation with other UAVs. Then, when the autonomous distributed allocation is launched, another algorithm with a lower computational cost is required. Each time a new task is received, the plan builder runs an algorithm that inserts the new task in all the possible and feasible locations in the current plan and chooses the order with lowest plan cost.

In the next Section, the module driving the autonomous distributed task allocation process is described.

## 2.5 CNP Manager

The CNP manager module (see Fig. 2) operation is based on the Contract Net Protocol [22] and it manages the distributed task allocation process among the different UAVs.

The *multi-robot task allocation* (MRTA) problem has become a key research topic in the field of distributed multirobot coordination in recent years. In the AWARE project, three algorithms (SIT, SET and S+T) have been developed and implemented to solve the distributed task allocation problem among the UAVs in the platform. All of them apply a market-based approach in which the UAVs consider their local plans when bidding and multiple tasks can be allocated to a single UAV during the negotiation process. The second one (SET) is based on the negotiation of subset of tasks and can be considered as a generalization of the former (SIT),

which only negotiates single tasks. Both algorithms have been tested in simulation with good results in multiple missions involving the execution of random GOTO tasks [25].

The latter algorithm (S+T) solves the MRTA problem in applications that require the cooperation among the UAVs to accomplish all the tasks. If an UAV cannot execute a task by itself, it asks for help and, if possible, another UAV will provide the required service. In the AWARE platform, tasks such as surveillance, that could involve transmitting data in real-time to a given location are considered. For those tasks, communication relay services could be required if the communication range of a single UAV is not enough. On the other hand, the parameters of the S+T algorithm can be adapted to give priority to either the execution time or the energy consumption in the mission. The potential generation of deadlocks associated to the relation between tasks and services have been studied and a distributed algorithm that prevents them have been also implemented [26].

In the following, the SIT algorithm used during the mission execution described in Section 4 is detailed.

### *2.5.1 Dynamic Single Task Negotiations with Multiple Allocations to a Single Robot (SIT)*

As our goal was to find solutions close to the global optimum (minimize the sum of all the individual costs assuming independent tasks), the approach presented in [7] was taken as a starting point. In the same manner, robots with a local plan and multiple tasks allocated to a single robot during the negotiation were considered. But in the implementation of the SIT algorithm, several differences with the work in [7] can be pointed out: revenues are not used, a different synchronization method is applied and there is an agent acting as an entry point for the tasks.

In the negotiation process, each UAV bids for a task with the cost of inserting this task in its local plan (marginal cost). Let us assume that the  $i$ -th UAV has a local plan  $\mathcal{P}_i$  consisting in a set of  $n$  ordered tasks  $\tau_1, \tau_2, \dots, \tau_n$  with cost  $C^i$  and receives a new task. If this task is inserted at the position  $j$  in the plan  $\mathcal{P}_i$ , then a new plan  $\mathcal{P}_i(\tau_j)$  with cost  $C^i(\tau_j)$  is generated. In that case, the associated marginal cost  $\mu_j$  is given by:

$$\mu_j = C^i(\tau_j) - C^i \quad (5)$$

The plan builder module of each UAV will compute the optimal insertion point of the new task in its current plan. Taking into account the local plan of each UAV in the negotiation leads to better solutions as it will be shown later.

The SIT algorithm is based on two different roles played dynamically by the UAVs: auctioneer and bidders. In each auction there is only one auctioneer which is the UAV that has the token. The auction is opened for a period of time and all the bids received within it are considered. When the auction is finished and the task allocated, the auctioneer considers to pass the token to another UAV. If that happens, the auctioneer changes its role to a bidder role and the UAV with the token becomes the new auctioneer. These basic steps of the algorithms executed by the auctioneer and the bidders are given in Algorithms 1 and 2. In Algorithm 1, the best bid collected by the auctioneer is increased by a given percentage (usually 1%) to avoid transactions that will not significantly improve the solution.

**Algorithm 1** SIT auctioneer algorithm

---

```

if there is any task to announce then
    announce task
    while timer is running do
        receive bids
    end while
    calculate best bid
    if best bid is smaller than the auctioneer bid then
        send task to best bidder
    end if
    delete task from announcement list
end if

```

---

The main difference with the basic CNP protocol is that the bid of each UAV depends on its current plan and every time the local plan changes, the negotiation continues until no bids improve the current global allocation. When the initial negotiation is over, the mission execution can start, but new tasks can be generated at any moment. Therefore, the negotiation is dynamic in the sense that new tasks are handled also during the mission execution. All the UAVs take part in the negotiation of those new tasks with the only restriction that the current tasks in execution are not re-negotiated.

**Algorithm 2** SIT bidder algorithm

---

```

a new message is received
if new message is a task announcement then
    calculate the optimal position of the task in the local plan
    calculate bid (marginal cost)
    send bid to the auctioneer
else if new message is a task award then
    insert task in the local plan in the position calculated before
    introduce task in announcement list
end if

```

---

The SIT algorithm has been tested in multi-UAV missions consisting in visiting waypoints and returning to the home location. In this case, the local plan cost for an UAV  $i$  visiting a set of  $n_w$  ordered waypoints  $P_1, P_2, \dots, P_{n_w}$  can be expressed as:

$$C^i = d(P(x_i), P_1) + \sum_{l=2}^{n_w} d(P_{l-1}, P_l) + d(P_{n_w}, P(h_i)), \quad (6)$$

where  $P(x_i)$  are coordinates corresponding to the current state of the  $i$ -th UAV,  $P(h_i)$  is its home location and  $d(A, B)$  is the Euclidean distance between the points  $A$  and  $B$ . In this particular missions, each UAV should build its own local plan visiting the waypoints in an order that minimizes the total distance travelled. This problem is equivalent to the TSP problem which is a well known *NP-hard* problem. In our

implementation, a greedy approach has been applied to solve it, inserting the new task in the position which minimizes its insertion cost.

Hundreds of simulations with different number of UAVs and waypoints have been run to compare the SIT algorithm with the global optimal solution. Additionally, another algorithm has been implemented in order to evaluate the relevance of the local plans computed by the plan builder module in the quality of the solutions. This second algorithm, that will be called *NoP* (No local Plan), uses a basic CNP protocol where UAVs only participate in the auction when they are idle. Furthermore, a brute force algorithm has been used to compute the global optimal solutions when the sum of UAVs and tasks is below a certain value.

In particular, for each given number of UAVs and waypoints, one hundred missions have been run in a virtual scenario of  $1000 \times 1000$  meters using random positions for the UAVs and the waypoints. Each mission has been simulated with the two algorithms implemented and Table 5 shows the different solutions compared with the global optimum. In each cell the first number is the arithmetic mean of the global cost for the 100 random missions, the value between brackets is its standard deviation (in meters) and the third number is the difference in percentage with the optimal solution. The global cost is given by the sum of the individual costs of the UAVs.

From the results, it should be noted that using a local plan during the auction process improves the solutions significantly. On the other hand, the SIT algorithm achieves very good results, with a maximum difference of 4.7% w.r.t. the optimal solution. Also, it is important to point out that the standard deviation values are high because missions are calculated at random, i.e., the global cost of the different random missions can differ very much among them.

## 2.6 Plan Merging Module

When considering the different plans of the UAVs, the main resource they share is the airspace. Therefore, a plan merging module (see Fig. 2) has been included in the architecture to detect potential conflicts among the different trajectories and also to follow a policy in order to avoid them. Then, this module has to interact with the plan builder module and also with other UAVs to interchange the different 4D trajectories.

**Table 5** Solutions computed with the distributed task allocation algorithms implemented and the optimal result

UAVs	Tasks	NoP	SIT	Optimum
3	3	2371, 22 (742, 4) 65, 18%	1453, 44 (369, 63) 1, 25%	1435, 4 (362, 36)
3	5	4144, 7 (923, 42) 101, 23%	2097, 83 (414, 52) 1, 74%	2061, 8 (396, 3)
3	7	5073, 2 (788, 75) 114, 73%	2473, 65 (385, 46) 4, 7%	2362, 6 (335, 81)
3	9	6070, 8 (850, 46) 129, 13%	2816, 59 (398, 16) 6, 3%	2649, 5 (332, 58)
5	3	1764, 36 (627, 87) 39, 49%	1274, 88 (365, 74) 0, 79%	1264, 78 (356, 04)
5	5	3808, 55 (921, 45) 112, 37%	1842, 96 (363, 62) 2, 76%	1793, 35 (337, 56)
5	7	5407, 59 (1238, 38) 150, 15%	2225, 67 (384, 74) 2, 96%	2161, 68 (365, 82)

The first number is the arithmetic mean of the global cost, the value between brackets is its standard deviation (both values in meters) and the third number is the difference in % with the optimal solution

A distributed collision avoidance method (see Algorithm 3) for the helicopters of the AWARE platform has been designed and implemented. The method is based on the hovering capabilities of the helicopters and guarantees that each trajectory to be followed by the UAVs is clear of other UAVs before proceeding to the execution.

This algorithm has been applied in the execution of the mission described in Section 4.

## 2.7 Plan Refining Toolbox

The plan refining toolbox (see Fig. 2) provides services to the plan builder module and can also interact with the Perception Sub-System (see Section 2.8) depending on the UAV mission.

The main services provided are:

- Task decomposition rules.
- Path planning algorithms to avoid static obstacles.
- Task and mission costs computations for the planner module.

Those services are based on the decomposition of the tasks received by the ODL into an ordered set of elementary tasks. Then, once a task  $\tau_k^i = (\lambda_k, \neg \Omega_k, \Omega_k^+, \varepsilon_k, \Pi_k)$

---

### Algorithm 3 Distributed collision avoidance algorithm running in the $i$ -th UAV

---

```

while (true) do
    a new message is received from UAV  $j$ 
    if new message is a path clearance ( $\Delta_j$ ) request with timestamp then
        if a task is running with path  $\Delta_i$  then
            if there is no conflict between paths  $\Delta_i$  and  $\Delta_j$  then
                send a path grant  $\Delta_j$  message to UAV  $j$ 
            else if then
                add the new request to the list of pending path clearance requests
            end if
        else
            if there is no conflict between current UAV  $i$  location and path  $\Delta_j$  then
                send a path grant  $\Delta_j$  message to UAV  $j$ 
            else if then
                add the new request to the list of pending path clearance requests
            end if
        end if
    else if new message is a  $\Delta_k$  path grant message AND elementary task  $\hat{\tau}_k$  is ready
    to be executed then
        if  $\Delta_k$  path grant messages have been received from all the UAVs then
            execute corresponding GOTO elementary task  $\hat{\tau}_k$ 
        else if then
             $\Delta_k$  path grant messages counter is increased
        end if
    end if
    check again the clearance state of the pending path requests previously received
end while

```

---

is received, it is processed following a set of decomposition rules  $\mathcal{D}$  and an associated set of  $n_e$  elementary ordered tasks  $\{{}^1\hat{\tau}_k^i, {}^2\hat{\tau}_k^i, \dots, {}^{n_e}\hat{\tau}_k^i\}$  is obtained. This set of tasks inherits as a whole, the preconditions  ${}^-\Omega_k$  and postconditions  $\Omega_k^+$  from the task  $\tau_k^i$ .

In the AWARE platform, there are several predefined decomposition rules  $\mathcal{D}$  for the different possible tasks. For example, for a surveillance task, consisting in covering a given area defined by a polygon at a certain altitude, an algorithm based in [16] is applied to decompose it into elementary GOTO tasks:

$$\tau_k^i = (\lambda_k = \text{SURV}, {}^-\Omega_k, \Omega_k^+, \varepsilon_k, \Pi_k) \longrightarrow \left\{ {}^j\hat{\tau}_k^i = (\hat{\lambda}_k = \text{GOTO}, \hat{\varepsilon}_k, \hat{\Pi}_k) / j = 1 \dots n_e \right\} \quad (7)$$

## 2.8 Perception Subsystem (PSS) Module

The main purpose of the Perception System (PS) of the AWARE platform is to build and update a consistent representation of the environment. A fully distributed probabilistic framework has been developed in order to achieve detection and tracking of events using the sensors provided by the AWARE platform: visual and infrared images from UAVs and ground cameras, scalar measures like temperature, humidity, CO or node signal strength from the sensor nodes of the WSN. It allows reducing the network bandwidth requirements on data transmission and dividing the processing load among different computers. As a result, the scalability of the whole system will be improved. In addition, the ability of separately process the information increases the robustness of the architecture.

Then, the whole PS system is divided in several software instances called perception sub-systems (PSS) modules, each attached to an AWARE platform element with perception capabilities. Then, there are PSS modules for the UAVs with cameras on-board (see Fig. 2), ground cameras, and the wireless sensor network (WSN). Each of them processes locally the environment information (images, sensors, ...) in order to reduce the amount of data transferred through the network. All the PSSs share their beliefs about specific events. The variety of events to be detected and tracked by the PSS is large but a common representation based on a probabilistic representation has been considered [17, 24]. The events are related with the real world by means of their position and velocity in a global coordinate frame. This information is always accompanied by the error estimation represented as an information matrix. In order to disambiguate among different events located in the same place (or close), general information about the event is also included: mean color, histogram, intensity and received signal strength indication (RSSI) when available. Finally, in order to fuse the different estimations, an Information Filter has been applied due to its properties for distributed implementation.

## 3 Multi-UAV Load Deployment

The transportation of loads by a single UAV is strongly limited by the payload constraint of the UAV. Then, when using small UAVs, this constraint may preclude the transportation and deployment of communication equipment or loads required for the application, as for example first aid supplies required for victims in search and rescue operations. The system designed in the AWARE project allows the

transportation of a single load by means of several helicopters. The number of helicopters is configurable depending on helicopters capabilities and the load to be transported. The multi-UAV cooperation in the load transportation task is characterized by the physical coupling between the members of the team. Then, the individuals are connected by physical links transmitted through the common load. The dynamic behaviour of the state variables  $q_i$  of the  $i$ -th helicopter is given by Eq. 1.

In terms of motion planning and collision avoidance, all the members of the team and the load can be considered as a single entity. Let  $\tau$  be the task being executed by this single entity, such as for example GOTO or GOTOLIST in the above section. Then, the control law can be expressed as

$$c_i = g(q_i, \bar{q}_i, \tau), \quad (8)$$

In general the control law  $u_i$  applied to each helicopter depends not only on  $q_i$  but also of the state variables  $\bar{q}_i$  of the other  $N_c$  helicopters linked to the load. However, as will be shown in this section, in some cases the problem can be decoupled.

### 3.1 Modeling

The model of a small size helicopter is a key component for behavior description of the whole system composed of one or several helicopters coupled to a load by means of ropes. Here we use the model presented in our previous work [12, 13]. The small size model helicopter shows some specific effects which are not presented or are negligible small in case of the full-size helicopter and vice versa. For that reason, it is impossible to use the models derived for full-size helicopters (see e.g. [10]) without any adaptation. As it was pointed out in our previous work the main differences between model and full size helicopters in respect of modeling and control are:

- a model helicopter has a much higher ratio of the main rotor mass to the fuselage mass
- the main rotor rotation speed of a model helicopter is higher compared to most full-size helicopters
- a very stiff main rotor without flapping hinges (for almost all commercially available helicopters)

Due to these differences, the inertial effects of the main rotor make a significant contribution to the rotational dynamics of the system and can not be neglected. Therefore, the mechanical model should be considered as composed of two rigid bodies: the main rotor and the fuselage. For the considered class of model helicopters, the dominant component is the main rotor and not the fuselage. Unfortunately, the main rotor is very often neglected in papers related to modeling and control of small-size helicopters and only one rigid body—the fuselage—is accounted for in the dynamical equations.

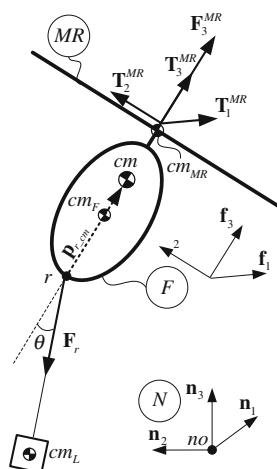
The complete model of a model helicopter is composed of two main components: the mechanical model and the model for generation of aerodynamic forces and torques. From experimental results with helicopters we concluded that the generation of aerodynamic forces and torques, at least for the considered class of helicopters, can be approximated with simple algebraic relations (corresponding time delay should be taken into account). Therefore, the dynamics of one small-

size helicopter or of a system composed of several coupled helicopters are mostly determined by its mechanical model. The lifting forces  $\mathbf{F}_3^{MR}$  and torques  $\mathbf{T}_{1,2}^{MR}$  generated by the main rotor of each helicopter, see Fig. 5, and the forces  $\mathbf{F}_2^T$  generated by the tail rotors will be considered as abstract control inputs  $\mathbf{c}$  in the above equation.

The important issue for the control of a helicopter coupled to a load is illustrated in Fig. 5. The mechanical model of the helicopter is composed of two rigid bodies: the fuselage  $F$  and the spinning main rotor  $MR$ . The load, denoted as mass point  $cm_L$ , is connected to the helicopter's fuselage by means of a rope in the point  $r$ . The motion of the whole system is considered in respect to a Newtonian frame  $N$ . The points  $cm_F$ ,  $cm_{MR}$  and  $cm$  are the center of mass (CoM) of the fuselage, of the main rotor and of the complete helicopter respectively. For a real system it is difficult to place the point  $r$  at the helicopter CoM  $cm$ , so the vector  $\mathbf{p}_{r-cm}$  connecting these two points is not zero and the rope force  $\mathbf{F}_r$  causes a non zero torque  $\mathbf{T}_r = \mathbf{F}_r \times \mathbf{p}_{r-cm}$  on the helicopter fuselage  $F$ . As we have shown in [12, 13] the rotation dynamics of the helicopter modeled as two rigid bodies is quite complicated, but is not coupled with the translation dynamics which means that equations for rotational dynamics depend only on generalized speeds describing the rotation of the helicopter. Because the lifting force  $\mathbf{F}_3^{MR}$  generated by the main rotor is approximately perpendicular to the main rotor plane or to the fuselage plane  $\mathbf{f}_1 - \mathbf{f}_2$ , the translational accelerations are always given by the absolute value of  $\mathbf{F}_3^{MR}$  and the orientation of the helicopter. Therefore, the relationship between rotation and translation dynamics for a single helicopter can be expressed as follows: rotation  $\Rightarrow$  translation.

In the case where one or several helicopters are connected by means of rope to a load, the rope force  $\mathbf{F}_r$  and, therefore, an additional torque  $\mathbf{T}_r$  acts on the fuselage of the helicopter. This torque depends on orientation of the helicopter and its translational motion in the frame  $N$  (e.g. if helicopter and load are in free fall,  $\mathbf{F}_r$  and  $\mathbf{T}_r$  are zero). Here we have a more complicated relationship between rotation and translation dynamics for each helicopter: rotation  $\Leftrightarrow$  translation. If several helicopters are connected to the load, the translational and rotational motion of one particular helicopter has direct influence on the rotational dynamics of all

**Fig. 5** Mechanical model of the helicopter connected to the load



other helicopters in the compound. Even the translation with constant acceleration, e.g. to the right in the Fig. 5, can cause oscillation of the angle  $\theta$  between the rope and the helicopter axis. Then, the coupling in Eq. 1 can be expressed as

$$\gamma_i = h(q_i, {}^p\bar{q}_i, {}^p\dot{\bar{q}}_i), \quad (9)$$

with vector  ${}^p\bar{q}_i = (q_{i1}, q_{i2}, \dots, q_{iN_c})$  containing the configurations of the  $N_c$  neighbors physically connected to the  $i$ -th helicopter.

The strong mutual coupling between rotation and translation makes the usage of an orientation controller, which was designed for a single helicopter, problematic (even if the techniques for robust control design were used). Please note that in many practical cases the absolute value of torque  $\mathbf{T}_r$  is similar or even larger than the values of torques needed to control the rotation of a single helicopter without the rope.

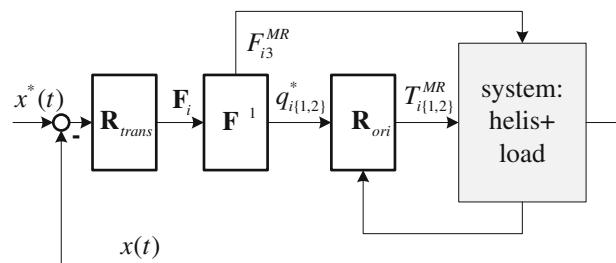
For modeling we consider the general case where  $n$  helicopters are connected to one load. To get the dynamical equations of motion we used the Kane-method, see e.g. [11], which allows to generate the equations for systems with an arbitrary number of helicopters. The coupling between equations for different helicopters is established by adding of one motion constraint (the length of the rope does not change during the time) for each helicopter in the compound. The resulting equations are used for control design as well as for simulation. For more on modeling see [3, 13]. The aerodynamics of the load and ropes are currently neglected, but will be considered in future work for flights with high velocity.

### 3.2 Controller Design

The general scheme of the proposed control algorithm for one or several helicopters coupled with the load is composed of two loops: the outer loop for translation control and inner loop to control the orientation of each helicopter, see Fig. 6.

The input of the control scheme in Fig. 6 is the desired trajectory  $\mathbf{x}^*(t)$  for helicopters or for the load. The translational motion of each helicopter is controlled in the outer loop by controller  $\mathbf{R}_{trans}$ . Using the deviations from the desired trajectories the controller  $\mathbf{R}_{trans}$  calculates for each helicopter  $i$  the forces  $\mathbf{F}_i$  which should be generated by its rotors. The helicopter  $i$  can realize the force  $\mathbf{F}_i$  by adjusting the absolute value of the main rotor lifting force  $F_{i3}^{MR}$  and adjusting the orientation of the main rotor plane or fuselage  $q_{i[1,2]}^*$ . The desired orientation of the main rotor plane is controlled in the inner loop by controller  $\mathbf{R}_{ori}$ . The values  $F_{i3}^{MR}$  and  $q_{i[1,2]}^*$  are calculated using algebraic relations in block  $\mathbf{F}^{-1}$ .

**Fig. 6** General control scheme for control of coupled helicopters



As mentioned above, the rope can not be connected to the center of mass of the helicopter and the force in the rope produces torques imposed on the helicopter fuselage. Therefore the influence from the load or from the other helicopters transmitted to the helicopter fuselage by the rope, makes the orientation control a challenging task [12, 13]. The developed approach for control of coupled helicopters is based on following three ideas:

- The design of the orientation controller (for each helicopter) accounts for the complete dynamics (translational and rotational) of the whole system, all coupled helicopters and the load. This is required due to the strong mutual coupling between translation and rotation of each helicopter and the load.
- The design of the translation controller is based on simplified model and accounts only for the translational dynamics of the whole system. In this simplified model the helicopters are modeled as mass points which can produce forces in direction perpendicular to current orientation of the main rotor plain.
- The usage of the force sensor in the ropes simplifies the design of orientation controller and makes it robust against variations of system parameters and disturbances.

The usage of the force sensor in the rope is a key issue in the proposed controller design. The force measured in the rope can be easily recalculated into resultant force and torque acting on the fuselage from the rest of the system. These two values are used in the feedback loop of orientation controller. The usage of the force sensor signal in the feedback loop has three main advantages:

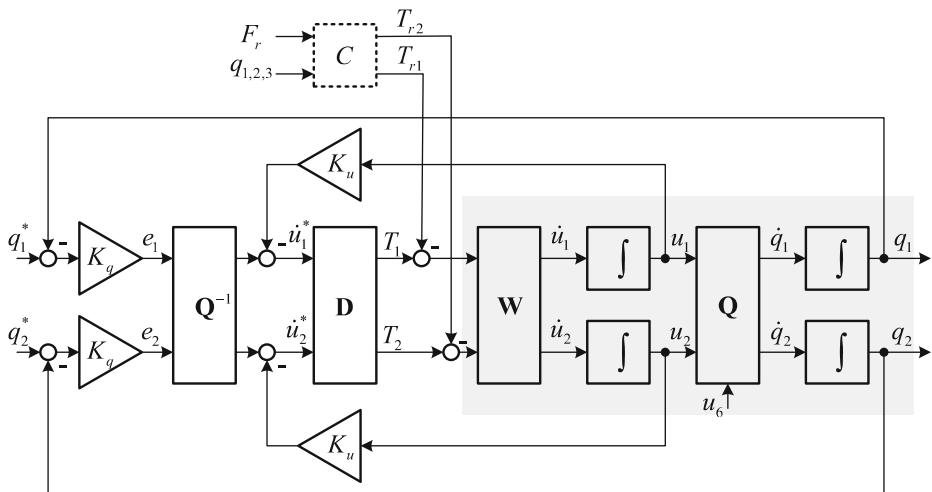
1. the closed loop system becomes very robust against variation of system parameters and disturbances
2. the orientation controller for such complicated systems becomes quite simple
3. the orientation controller does not depend on the number of helicopters connected to the load.

As shown in [13], the rotation dynamics for a single helicopter, represented by two rigid bodies for the fuselage and the main rotor, can be expressed by the following equations:

$$T_1^{MR} + K_{12}u_2 + K_{11}\dot{u}_1 = 0 \quad (10)$$

$$T_2^{MR} + K_{21}u_1 + K_{22}\dot{u}_2 = 0 \quad (11)$$

Where  $T_{1,2}^{MR}$  are the torques generated around the longitudinal and lateral axes of the fuselage,  $u_{1,2}$  are rotation speeds of the fuselage and the coefficients  $K_{xx}$  are constant parameters of the helicopter and of the main rotor speed. We assume here that the influence of the rotation around the vertical axis (rotation speed  $u_3$ ) on  $u_{1,2}$  is small and can be considered as disturbance. This assumption is true if  $u_3$  is hold on zero or on constant values using additional tail rotor controller with a time constant much smaller compared to the time constant of the orientation controller for  $u_{1,2}$ . Equations 10, 11 are coupled through  $u_{1,2}$ . This coupling leads to oscillations (for the parameters of typical small size helicopter) once the system has been stimulated. The scheme for the control of roll and pitch angles  $q_{1,2}$  for a single helicopter is shown in Fig. 7, where the gray block denotes the model of helicopter rotational dynamics and kinematics. The controller is composed of blocks  $\mathbf{Q}^{-1}$ ,  $\mathbf{D}$  and two feedback loops with gains  $K_u$ ,  $K_q$  for rotation speeds  $u_{1,2}$  and orientation angles  $q_{1,2}$ .



**Fig. 7** Scheme for the orientation control

respectively. The rotation dynamics described by Eqs. 10, 11 are represented in Fig. 7 by the block  $\mathbf{W}$ . The block  $\mathbf{D}$  of the controller is used to decouple the plant between  $T_{1,2}^{MR}$  and  $u_{1,2}$ . The decoupling can be performed by means of known techniques from linear control theory, e.g. using matrix composed of compensating transfer functions. This orientation controller shows a good performance and robustness in simulation and real flight experiments with different types of helicopters as we have shown in [12, 13].

The rotational dynamics of a helicopter coupled to the load by means of a rope are strongly influenced by the motion of the whole system. To account for this influence, block  $\mathbf{D}$  should be replaced by the inverse rotational dynamics  $\tilde{\mathbf{D}}$  not of a single helicopter, but of the whole system (considering both, the rotation and translation of each helicopter). With this new block  $\tilde{\mathbf{D}}$ , the orientation controller for helicopter coupled to the load shows equal performance as the orientation controller with block  $\mathbf{D}$  for a single helicopter without load in case for nominal values of all system parameters  $\rho_i$ :

$$c_i = g(q_i, \dot{q}_i, \rho_i), \quad (12)$$

The simulation experiments have shown that, unlike in the case of a single helicopter, the orientation controller with inversion block  $\tilde{\mathbf{D}}$  for coupled helicopter is quite sensitive to variation of the system parameters  $\rho_i$  (5% variation could be critical). To overcome this problem, we propose to use a force sensor in the rope. The force  $\mathbf{F}_r$ , measured in the rope, will be used to calculate the influence on the rotational dynamics of the coupled helicopters from the remainder of the system. This influence is expressed by means of torque  $\mathbf{T}_r = \mathbf{F}_r \times \mathbf{p}_{r-cm}$ , where  $\mathbf{p}_{r-cm}$  is the position vector connecting rope attaching point  $r$  and helicopter CoM. The resulting orientation controller

$$c_i = g(q_i, F_r), \quad (13)$$

is composed of the orientation controller for a single helicopter and the compensator block  $\mathbf{C}$ , see Fig. 7, where  $\mathbf{T}_r$  is calculated and subtracted from torques calculated

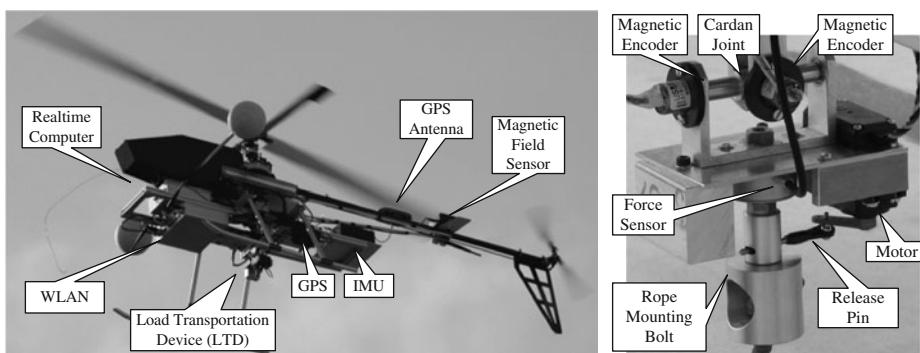
in **D**. The usage of the compensator **C** allows us to decouple the control of each helicopter from the rest of the system and to use the same controller independent of the number of helicopters coupled together via the ropes.

There are two reasons for the robustness of the proposed orientation controller: First, the actual influence of the load on the fuselage is measured through **F<sub>r</sub>**, and, therefore, the compensation becomes independent from the mass of the load and the length of the rope. Second, as long as the orientation of the helicopter is known, the calculated compensation torque is always in the correct phase.

The details of the presented control algorithms can be found in [2, 3, 12, 13].

### 3.3 System Description

In Fig. 8 one of the TUB UAVs, used for the slung load transportation experiments, is shown during flight. The UAVs are based on commercially available small size helicopters. The helicopters have a rotor diameter of 1.8 m, a main rotor speed of approximate 1300 rpm and are powered by a 1.8 KW two-stroke engine. The UAVs can carry about 1.5 kg of additional payload, whereas the weight of the UAV itself is 12.5 kg. The different components necessary to achieve autonomous flight capabilities are mounted to the helicopters, using a frame composed of strut profiles. Through the use of these profiles, the location of hardware components can be altered and new hardware can be installed easily. This allows quick reconfiguration of the UAVs for different applications, easy replacement of defective hardware and alteration of the position of different components to adjust the UAVs center of gravity. The components necessary for autonomous operation are shown in Fig. 8: A GPS, an IMU, a control computer and a communication link. Due to the strong magnetic field of the engine a magnetic field sensor is mounted on the tail. All UAVs are equipped with Load Transportation Devices (LTD), which are specially designed for the transportation of slung loads using one or more UAVs (see Fig. 8). The LTD is composed of a two axis cardan joint with two magnetic encoders attached to each axis. After the joint a force sensor is mounted. After the force sensor a release mechanism for the rope is attached, which is composed of a bolt, inserted into a tube. The bolt is fixed in the tube through a pin, which can be pulled out by a small motor, to release the load. The release mechanism can be used for emergency decoupling of



**Fig. 8** UAV component diagram

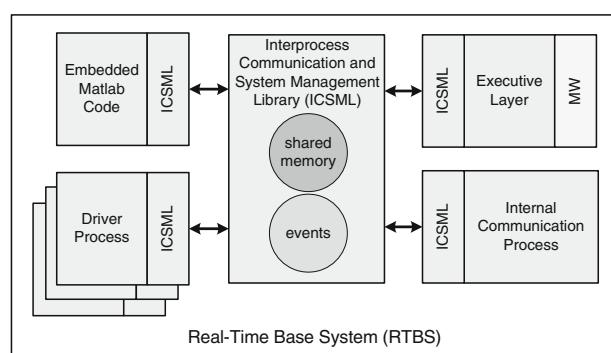
the UAV from the load (or the other coupled UAVs), but also to release the load after successful transportation. The magnetic encoders allow to measure the rope orientation relative to the UAV fuselage. With this information and the measured force in the rope, it becomes possible to calculate the torque and force imposed on the UAV through the load (and/or the other coupled UAVs), which are used in the feedback loop of the compensator block **C**, see Section 3.2.

In Fig. 9 the concept of the software system running on the UAV board computers is shown. The Real-Time Base System is composed of multiple separate modules, which are connected by the Interprocess Communication and System Management Library (ICSL). The ICML offers, among other features, convenient interprocess communication methods, using a shared memory design and a event system for interprocess notifications (e.g. on the change of shared memory variables). The driver modules are used to communicate with the peripheral devices of the UAV, such as a compass, IMU or GPS. The device specific protocol is encapsulated in the driver process and the sensor data are presented to the remaining system through ICML as generalized objects. The controller of the system was designed and tested in simulation using Matlab/Simulink. The RTBS provides a generic wrapper module, to embed C-Code generated by Matlab/Simulink. This allows fast implementation of new control algorithms and an error free transition from Simulink models to real-time controller code. The internal communication process is used for development and testing purposes and for experiments without the AWARE system. As explained in Section 2 the executive layer is the proprietary interface to the AWARE On-Board Deliberative Layer (ODL). The executive layer module provides the basic functionalities needed by the ODL. This includes the commanding of the UAVs, reporting of the command execution progress and the UAV state. Additionally the executive layer module provides a certain level of hardware abstraction: The relative configuration of the UAVs during the load transportation is part of the control loop and shouldn't be altered by the ODL. Therefore, during the load transportation the coupled UAVs are presented to the ODL as a single entity. This way the commanding interfaces for coupled and uncoupled UAVs are equal.

### 3.4 Experimental Results

The load transportation system has been used several times in different experiments, as stand alone system as well as a part of AWARE platform. In Fig. 10 two load

**Fig. 9** Real-Time Base System (RTBS)

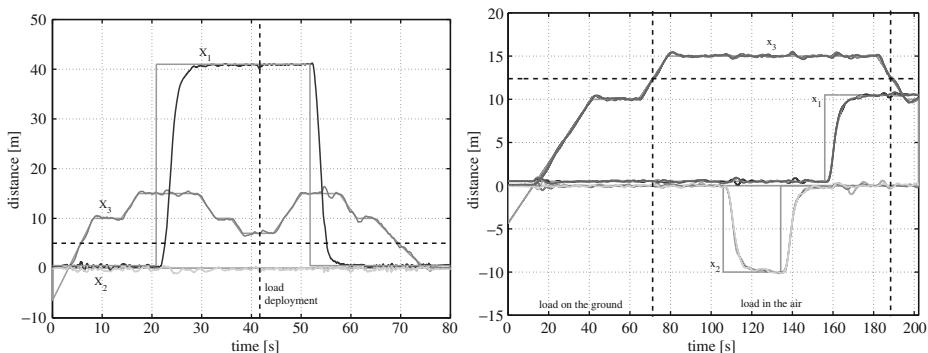


**Fig. 10** Single and multiple UAV slung load transportation



transportation tasks are shown. The picture on the left with one helicopter transporting the load was taken in Utrera (Spain), April 2008. The first successful experiment was conducted by the authors in Berlin, November 2007 (<http://www.youtube.com/watch?v=-J3gmzn0bSa4>). The picture on the right with three helicopters transporting the load was taken on our test ground near Berlin, December 2007 (<http://www.youtube.com/watch?v=tl6DYWNe9ac>). For these flight experiments three identical helicopters as described in Section 3.3 were used. The helicopters are equipped with a multi-UAV modular autopilot system developed at TUB. The rope is attached to the helicopter by means of the Load Deployment Device (LDD) which is mounted between the landing skids.

In the first experiment one helicopter transported a load of 1 kg using a rope of 5 m length. Figure 11 (left) shows part of the system state during this experiment. All coordinates are given in an ground fixed frame with the origin at the helicopter take-off position. Coordinate  $x_3$  represents the helicopter altitude. Until the load was dropped the coordinates  $x_{1,2}$  show the position of the load and after the load



**Fig. 11** Motion of the load (1 kg) during transportation using one helicopter (left) and motion of three helicopters transporting one load of 4 kg (right)

dropping, the position of the helicopter itself. The dropping is indicated by a vertical slashed line. The gray line denotes the desired position given to the controller. We consider the performance of the controller as quite good, despite the stormy weather conditions on that day. Even with steady wind of 30 km/h and wind gusts up to 40 km/h the controller was able to stabilize the helicopter and damp upcoming oscillation of the load.

In the second experiment a load of 4 kg was transported by means of three helicopters. In this experiment ropes with a length of 13 m were used. The helicopters were arranged as a equilateral triangle on the ground, with a distance of 8 m between the helicopters. In Fig. 11 (right) the coordinates of all three helicopters during the whole flight are shown. The coordinates of the helicopters are shown in different ground fixed frames, which have the same orientation, but different origins (take-off position of each helicopter), therefore there are no offsets between the helicopter trajectories. The load was lifted when the helicopters reached approximate 12.4 meters. The weight of the load was not considered in the controller and therefore a small disturbance in the  $x_3$  trajectories can be observed at the moment the load was lifted from the ground as well as during strong acceleration in  $x_{1,2}$ -direction. A position error of each helicopter in hovering was approx.  $\pm 0.3$  m During the whole flight the triangular formation of the helicopters with a precision of about  $\pm 0.3$  meters and the load was moved very smoothly. To our knowledge these were the very first successful flight experiments to transport a load using multiple autonomous helicopters.

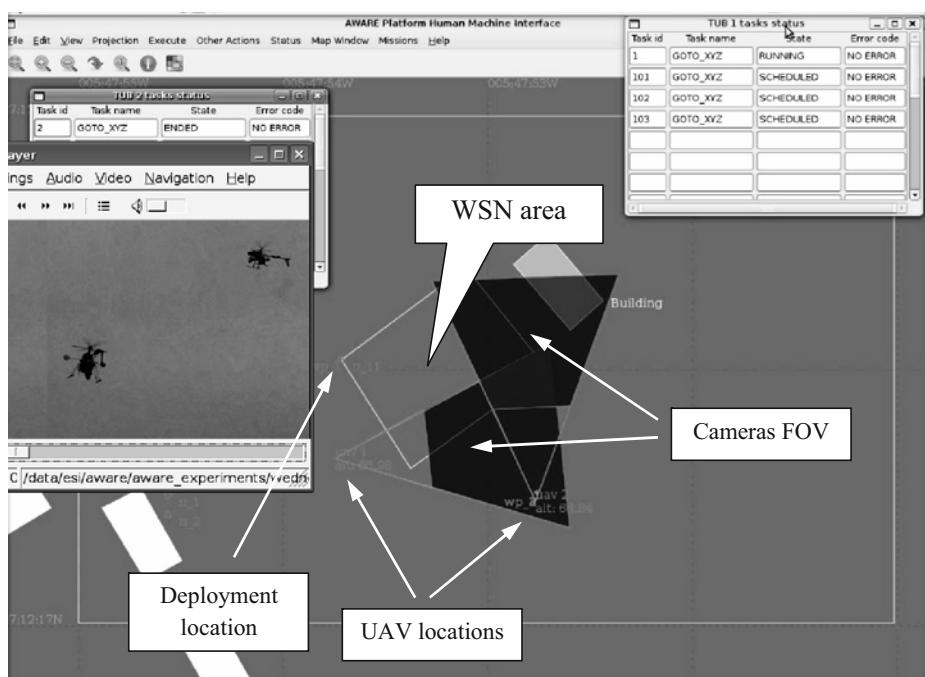
#### 4 Multi-UAV Deployment of Multiple Loads: Experimental Results

In this section, in order to illustrate the architecture of the AWARE platform, the execution of a real mission consisting in the deployment of sensor nodes and the monitoring of the scene by the helicopter team is described. In this case, the helicopters are not physically coupled but at least one helicopter carries a device that is able to deploy several sensor nodes.

This mission was performed in 2008 during the second year of the AWARE project (see [www.aware-project.net/videos/videos.shtml](http://www.aware-project.net/videos/videos.shtml)). The objective was to deploy a sensor node from an UAV in a given location to repair the WSN network connectivity, whereas another UAV supervised the operation with the on-board camera and also monitored the area for the deployment and a nearby building. During the mission, a third UAV took-off to monitor the whole operation. It should be mentioned that all the UAVs of the AWARE platform are autonomous helicopters.

The following systems were involved in the execution:

- HMI (Human Machine Interface) station: allowed to specify the mission (waypoints with their corresponding headings, speeds, altitudes, etc). After that, there was a mission briefing with the operators of the UAVs using a simulation of the execution in the HMI to tune several parameters of the mission using EUROPA. Once an agreement was met, the mission was ready to be executed. During the execution, the HMI allowed to supervise the execution state of the different tasks allocated to the UAVs. Figure 12 shows a screenshot of the HMI application during the experiment.



**Fig. 12** The HMI screen during the experiment: visualization of the locations and status of the TUB helicopters with their allocated elementary tasks

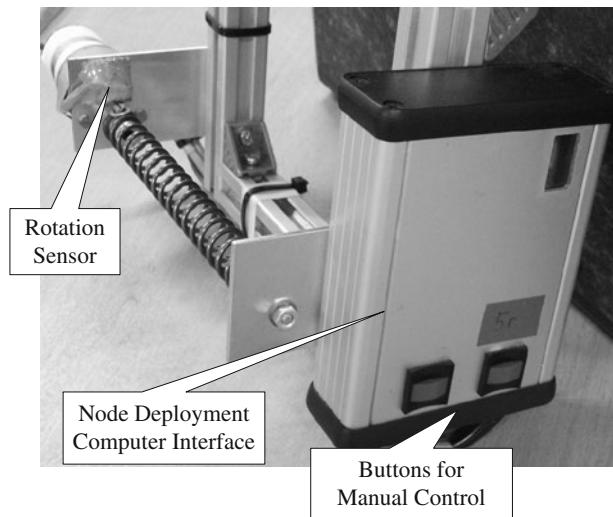
- (On-board Deliberative Layer) software of the UAVs: performing the function described in Section 2.
- UAV supervisor and executive software components: allowed to supervise the elementary tasks sent by the ODL in order to have another check point of their consistency. Finally, the executive software on-board the helicopter was in charge of the final execution of the elementary tasks.
- WSN gateway: In this experiment, the gateway was in charge of autonomously repairing the connectivity of the network once a node was placed between two isolated networks. The system was able to automatically re-compute the data routing and to send all the information from the nodes to the HMI.
- Middleware: allowed to communicate the tasks status, telemetry and images, and also to send high level commands from the HMI to the ODL and commands from the ODL to the UAV executive layer.

#### 4.1 Mission Description

The mission was specified by the AWARE platform user using the HMI software. It was a node deployment mission to repair the WSN connectivity involving three UAVs:

- UAV 1: equipped with a fixed camera aligned with the fuselage of the helicopter and pointing downwards 45°.

**Fig. 13** Detail of the device on-board the helicopter for the node deployment operation



- UAV 2: it has a device on-board for node deployment (see Fig. 13) and also a camera.
  - UAV 3: equipped with a camera mounted inside a mobile gimbal to film the whole mission from different point of views, transmitting images to the HMI through the middleware.
- The AWARE platform user specified the following tasks to be executed (see Table 6):
- $\tau_1$ : deploy a node in the waypoint wp\_1 with GPS geographical coordinates ( $-5.79816709, 37.20502859$ ) after  $\tau_2$  is completed. The goal was to repair the connectivity with the node with identifier 28 deploying the node 11 in wp\_1.
  - $\tau_2$ : visit waypoint wp\_2 ( $-5.79790241, 37.20488147$ ) to monitor a nearby building.
  - $\tau_3$ : During the node deployment, the corresponding WSN area should be also monitored with the camera from wp\_2.
  - $\tau_4$ : After node deployment, the building should be monitored again from wp\_2.

In the next section, the role of the different modules in the ODL architecture is described.

#### 4.2 ODL Modules During the Mission

As the user did not allocate those tasks manually, the distributed task allocation process started from the HMI software. The negotiation involved the CNP manager modules of UAV 1 and UAV 2, and due to the different devices on-board each UAV,

**Table 6** Tasks specified for the mission

$\tau_k$	$\lambda_k$	${}^-\Omega_k$	$\Omega_k^+$	$\Pi_k$
$\tau_1$	DEPLOY	END( $\tau_2$ )	$\emptyset$	$\Pi_1$
$\tau_2$	GOTO	$\emptyset$	$\emptyset$	$\Pi_2$
$\tau_3$	GOTO	START( ${}^{101}\hat{\tau}_1$ )	$\emptyset$	$\Pi_3$
$\tau_4$	GOTO	END( ${}^{103}\hat{\tau}_1$ )	$\emptyset$	$\Pi_4$

The values of the parameters ( $\Pi_k$ ) are detailed in Table 7

**Table 7** Values for the tasks parameters ( $\Pi_k$ )

Parameters ( $\Pi_k$ )	$\Pi_1$	$\Pi_2$	$\Pi_3$	$\Pi_4$
$\pi^1$	−5.79816709	−5.79790241	−5.79790241	−5.79790241
$\pi^2$	37.20502859	37.20488147	37.20488147	37.20488147
$\pi^3$	67.0	67.0	67.0	67.0
$\pi^4$	1.1	1.1	1.1	1.1
$\pi^5$	1	1	1	1
$\pi^6$	90	0	−45	0
$\pi^7$	1	0	0	0

task  $\tau_1$  was allocated to UAV 2 whereas the rest of tasks were allocated to UAV 1 (bids with infinite cost for task  $\tau_1$ ).

In this mission, the plan builder role was trivial: for both UAVs a take-off was required before executing the allocated tasks and the ordering of the tasks was fixed by the preconditions.

Then, the plan refining module of UAV 2 decomposed  $\tau_1$  as follows:

$$\tau_1 \longrightarrow \{ {}^1\hat{\tau}_1, {}^{101}\hat{\tau}_1, {}^{102}\hat{\tau}_1, {}^{103}\hat{\tau}_1 \} \quad (14)$$

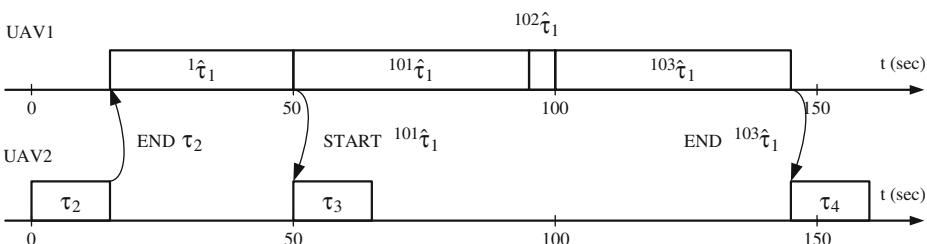
with:

- ${}^1\hat{\tau}_1$ : visit wp\_1.
- ${}^{101}\hat{\tau}_1$ : descend to an altitude of 3 meters above the ground.
- ${}^{102}\hat{\tau}_1$ : activate the device for node deployment.
- ${}^{103}\hat{\tau}_1$ : ascend back to the initial altitude.

whereas in UAV 1, it computed the headings required for monitoring the WSN area and the node deployment operation with the fixed camera from wp\_2.

During the mission, the interface with the executive layer was the task manager. For example, once the executive layer changed the status of  ${}^1\hat{\tau}_1$  from RUNNING to ENDED, the task manager sent the next task ( ${}^{101}\hat{\tau}_1$ ) for its execution. Moreover, the dependencies between tasks of different UAVs were also handled by the task manager with the assistance of the synchronization module. On the other hand, the plan merging modules running the algorithm shown in Section 2.6 did not detect any conflict between the planned 4D trajectories of the UAVs and no change in the plan was inserted.

The resulting evolution of the tasks during the mission is shown in Fig. 14, where the different preconditions have been represented by arrows.



**Fig. 14** Tasks executed by each UAV. The arrows represent the different preconditions summarized in Table 6



**Fig. 15** Coordinated flights during the sensor node deployment mission (see <http://www.aware-project.net/videos/videos.shtml>)

Finally, it should be mentioned that this experiment represents the first test of the AWARE platform integrating the three helicopters and the wireless sensor network in the same mission. Regarding the WSN, the connectivity with node 28 was achieved a few seconds after node 11 was deployed. Figure 15 shows several photographs taken during the experiment.

## 5 Conclusions

The cooperation of multiple autonomous aerial vehicles is a very suitable approach for many applications, including detection, precise localization, monitoring and tracking in emergency scenarios. In these applications the UAVs do not modify the state of the environment and there are no physical interactions between the UAV and the environment. Furthermore, the interactions between the UAVs, are essentially information exchanges without physical couplings between them. This paper demonstrates the possibilities of the cooperation and control of multiple aerial robots with sensing and actuation capabilities allowing load deployment (in particular sensor nodes deployment).

Furthermore, the paper presents the multi-UAV load transportation, which requires the consideration of physical interactions between the aerial robots. The paper

has presented a multi-UAV architecture developed in the AWARE project that allows different level of interaction among the UAVs and between the UAVs and the environment, including both sensing and actuation. The interaction between all the systems in this architecture can be represented by means of an hybrid system formulation coping with both discrete events associated to the tasks to achieve a given mission and the continuous dynamic interactions between physically coupled UAVs.

Particularly, the paper has presented results obtained in the AWARE project demonstrating the lifting and transportation of a slung load by means of one helicopter and also by three coupled helicopters, which has been the first demonstration of this challenging application. Finally, the paper also presents a multi-UAV mission consisting of the deployment of the sensor nodes of a WSN.

The proposed methods open many different new opportunities in missions involving the cooperation of multiple UAVs for applications such as search and rescue and interventions in disaster management and civil security. The transportation of loads by means of the UAVs can be also considered as a first step toward the cargo transportation by means of UAVs.

**Acknowledgements** The authors would like to thank the AWARE Project reviewers Fernando Lobo Pereira and Ørnulf Jan Rødseth for their very constructive advice and orientation in the project.

## References

1. Banâtre, M., Marron, P., Ollero, A., Wolisz, A.: Cooperating Embedded Systems and Wireless Sensor Networks. Wiley, New York (2008)
2. Bernard, M., Kondak, K.: Generic slung load transportation system using small size helicopters. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3258–3264 (2009). doi:10.1109/ROBOT.2009.5152382
3. Bernard, M., Kondak, K., Hommel, G.: Load transportation system based on autonomous small size helicopters. In: 23rd Int. Unmanned Air Vehicle Systems Conf. (2008)
4. Borenstein, J.: The Omnimate: a guidewire- and beacon-free AGV for highly reconfigurable applications. Int. J. Prod. Res. **38**(9), 1993–2010 (2000)
5. Chaimowicz, L., Kumar, V., Campos, M.F.M.: A paradigm for dynamic coordination of multiple robots. Auton. Robots **17**(1), 7–21 (2004)
6. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3602–3608 (2004)
7. Dias, M.B., Stenz, A.: Opportunistic optimization for market-based multirobot control. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2714–2720, Lausanne (2002)
8. Fierro, R., Das, A., Spletzer, J., Esposito, J., Kumar, V., Ostrowski, J.P., Pappas, G., Taylor, C.J., Hur, Y., Alur, R., Lee, I., Grudic, G., Southall, B.: A framework and architecture for multi-robot coordination. Int. J. Rob. Res. **21**(10–11), 977–995 (2002)
9. Huntsberger, T.L., Trebi-Ollennu, A., Aghazarian, H., Schenker, P.S., Pirjanian, P., Nayar, H.D.: Distributed control of multi-robot systems engaged in tightly coupled tasks. Auton. Robots **17**(1), 79–92 (2004)
10. Johnson, W.: Helicopter Theory. Dover, New York (1980)
11. Kane, T., Levinson, D.: Dynamics: Theory and Applications. McGraw-Hill, East Roseville (1985)
12. Kondak, K., Bernard, M., Losse, N., Hommel, G.: Elaborated modeling and control for autonomous small size helicopters. In: ISR/ROBOTIK 2006 Joint Conference on Robotics (2006)
13. Kondak, K., Bernard, M., Meyer, N., Hommel, G.: Autonomously flying VTOL-robots: Modeling and control. In: IEEE Int. Conf. on Robotics and Automation, pp. 2375–2380 (2007)

14. Kosuge, K., Sato, M.: Transportation of a single object by multiple decentralized-controlled non-holonomic mobile robots. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, vol. 3, pp. 1681–1686 (1999)
15. Li, H., Karray, F., Basir, O., Song, I.: A framework for coordinated control of multiagent systems and its applications. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans* **38**(3), 534–548 (2008)
16. Maza, I., Ollero, A.: Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In: Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems, pp. 211–220, Toulouse (2004)
17. Merino, L.: A cooperative perception system for multiple unmanned aerial vehicles. Application to the cooperative detection, localization and monitoring of forest fires. Ph.D. thesis, University of Seville (2007)
18. Mittal, M., Prasad, J.V.R., Schrage, D.P.: Nonlinear adaptive control of a twin lift helicopter system. *IEEE Control Syst. Mag.* **11**(3), 39–45 (1991)
19. Ollero, A., Maza, I.: Multiple Heterogeneous Unmanned Aerial Vehicles. Springer Tracts on Advanced Robotics. Springer, New York (2007)
20. Ollero, A., Merino, L.: Control and perception techniques for aerial robotics. *Annu Rev Control* **28**(2), 167–178 (2004)
21. Reynolds, H.K., Rodriguez, A.A.:  $H_\infty$  control of a twin lift helicopter system. In: Proceedings of the 31st IEEE Conference on Decision and Control, pp. 2442–2447 (1992)
22. Smith, G.: The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. Comput.* **29**(12) (1980)
23. Sugar, T., Kumar, V.: Decentralized control of cooperating mobile manipulators. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 2916–2921 (1998)
24. Sukkarieh, S., Nettleton, E., Kim, J.H., Ridley, M., Goktogan, A., Durrant-Whyte, H.: The ANSER project: data fusion across multiple uninhabited air vehicles. *Int. J. Rob. Res.* **22**(7–8), 505–539 (2003). doi:10.1177/02783649030227005
25. Viguria, A., Maza, I., Ollero, A.: SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3339–3344. Rome (2007)
26. Viguria, A., Maza, I., Ollero, A.: S+T: an algorithm for distributed multirobot task allocation based on services for improving robot cooperation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3163–3168. Pasadena (2008)

# Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter

Sara Erhard · Karl E. Wenzel · Andreas Zell

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 2 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** An unmanned aerial vehicle (UAV) needs to orient itself in its operating environment to fly autonomously. Localisation methods based on visual data are independent of erroneous GPS measurements or imprecise inertial sensors. In our approach, a quadrocopter first establishes an image database of the environment. Afterwards, the quadrocopter is able to locate itself by comparing a current image taken of the environment with earlier images in the database. Therefore, characteristic image features are extracted which can be compared efficiently. We analyse three feature extraction methods and five feature similarity measures. The evaluation is based on two datasets recorded under real conditions. The computations are performed on a Nokia N95 mobile phone, which is mounted on the quadrocopter. This lightweight, yet powerful device offers an integrated camera and serves as central processing unit. The mobile phone proved to be a good choice for visual localisation on a quadrocopter.

**Keywords** Computer vision · Unmanned aerial vehicles (UAV) · Visual localisation · Mobile devices · Onboard computation

## 1 Introduction

This paper introduces a visual self-localisation system for unmanned aerial vehicles (UAV) which is efficient enough to run at 1.7 Hz on a Nokia N95 mobile phone.

---

S. Erhard (✉) · K. E. Wenzel · A. Zell  
Department of Computer Science, University of Tübingen,  
Sand 1, 72076 Tübingen, Germany  
e-mail: sara.erhard@uni-tuebingen.de

K. E. Wenzel  
e-mail: karl.e.wenzel@uni-tuebingen.de

A. Zell  
e-mail: andreas.zell@uni-tuebingen.de

Our UAV is planned to operate autonomously from external control: It is designed to navigate without a pilot and is to be independent of a base station's processing power or memory. Further, we focus on vision-based localisation without GPS or inertial sensors. The environment is not artificially changed by markers.

We use a model quadrocopter as UAV. The system is limited to a payload of only 300 g. Thus, we were looking for a lightweight device with a camera and enough processing power and memory. We decided to use a mobile phone. It weighs only 120 g and meets the above criteria.

Our approach is based on *content-based image retrieval* (CBIR). That is, we build an image database from the environment during one exploration flight. Afterwards, the quadrocopter is able to locate itself by comparing the current image to the training database. Therefore, the actual localisation is only based on visual data. However, at this stage, we use GPS as ground truth at the exploration flight.

To compare images, we extract features which identify and characterize an image. The main concern of appearance-based localisation methods is to find unique image attributes that represent the view efficiently and in a simple data structure [27]. Finding features containing these characteristics is a difficult task due to the complex nature of the environment of an outdoor robot. Shapes are unstructured and irregular, colors are permanently changing. Illumination alters depending on time of day and weather conditions. People or objects like cars can change their position. In our analysis we lay our focus on the influence of illumination changes. Further, the camera is mounted on a quadrocopter, which implies that its pose can vary in six degrees of freedom (DoF). Therefore, the features should not be influenced by image rotations and translations.

One distinguishes between global methods, which use the whole image to calculate features, and local methods, which use several points of interest. The detection of salient points is computationally expensive. Multiple local features have to be stored for each image. A global feature establishes a compact representation of one image. Besides simplicity, it requires less memory in comparison with local features. Its main disadvantage is its sensitivity to occlusion. In order to achieve robustness while still being efficient, we use local image grids to handle the occlusion problem. We analyse three global methods under real conditions. To compare these image features, one needs distance measures to determine how similar two features are. Thus, we analyse five different comparison measures.

We combine the two aspects of visually locating and mobile computing to an UAV system which we call "Flyphone". The focus of this paper lies mainly on analysing different feature extraction algorithms for localisation purposes.

The remainder of the paper is organized as follows. In Section 2, we introduce the related work. Section 3 describes our robot system, consisting of a quadrocopter and a mobile phone. We present algorithms, which extract image features, and similarity measures to compare these features in Section 4. Section 5 analyses the presented extraction and comparison methods with our airborne system. Finally, conclusions are drawn in Section 6.

## 2 Related Work

One contribution of this work is to bring together UAVs and mobile phones as lightweight, but computationally powerful processing devices. Thus, related work can

be categorised into two main areas: Self-localisation of robots, especially UAVs, and image processing on hand-held mobile phones.

Autonomous aerial vehicles have become an active area of research in recent years. Some approaches give a solution to the problem of self-localisation by realizing *Simultaneous Localisation And Mapping* (SLAM) with visual data and additional sensors [11, 18, 19, 24, 25]. It aims at simultaneously building and updating an environment map. The problem of determining the position and orientation of a robot by analysing the associated camera images is known as *visual odometry* in the robot vision community. Accurate results have been obtained by using stereo or omnidirectional cameras [6, 14, 20]. Methods like optical flow analysis or local feature tracking are used for instance on Mars missions [14]. Not only for localisation, but also for stabilisation or control purposes various approaches use artificial markers or known targets [2, 16, 17]. By contrast, we analyse images by extracting features of the environment without artificial markers or targets. This is also done in work with groundrobots [1, 9, 10, 21, 23, 29, 33]. Instead of an artificially modified environment, we need some prior knowledge with means of raw image data. This knowledge is obtained by preceding exploration flights, where images of the environment are recorded. Splitting the process in two phases has also been done by Valgren et al. [29], Lamen et al. [12], Ulrich and Nourbakhsh [28] and Zhou et al. [34]. Valgren's work is similar to ours in the sense that it uses image features to localise a robot in a changing, but explored environment. Our approach uses global features instead of local ones. It deals with the characteristics of flying robots and uses solely the built-in camera of a mobile phone instead of an omni-directional camera.

Some UAV systems perform complex vision-based applications not onboard, but on a ground station [7], because they demand high computational power. We decided not to use a ground station, but to perform the entire image processing onboard a mobile phone. Although, the technical limitations of mobile devices make it difficult to implement image processing on them, there are approaches in context of user interaction and information services. Wang et al. [31] detect the motion of a mobile phone by analysing image sequences with motion estimation. While Castle et al. [5] compute visual SLAM for wearable cameras on a 3.2 GHz Pentium 4 CPU, there are image processing methods which run on a Nokia N95 mobile phone. Wagner et al. [30] implemented the *Scale Invariant Feature Transform* (SIFT) [13] on a N95 to find known textured targets. The mobile information system by Takacs et al. [26] uses CBIR. Here, users take pictures of a building and the system finds similar images in a database by comparing *Speeded Up Robust Features* (SURF) [3]. The system further provides the user with information about the building. In place of local image features, we study the applicability of global features for vision-only localisation.

### 3 System Components

The main components of our robot system are the quadrocopter and a mobile phone (Fig. 1). A quadrocopter is a fixed-wing aircraft which is lifted and propelled by four rotors. It is controllable by variation of the rotor's rotation speed. The lightweight quadrocopter is a commercial Asctec X3D-BL model quadrocopter, which is in detail described in [8]. It is equipped with a GPS sensor which reaches an accuracy of about 2.5 m. The quadrocopter lifts a maximum payload of 300 g. Thus, we need a lightweight device which supports a camera, computing power, memory, and which

**Fig. 1** The quadrocopter Asctec X3D-BL with a Nokia N95 mobile phone



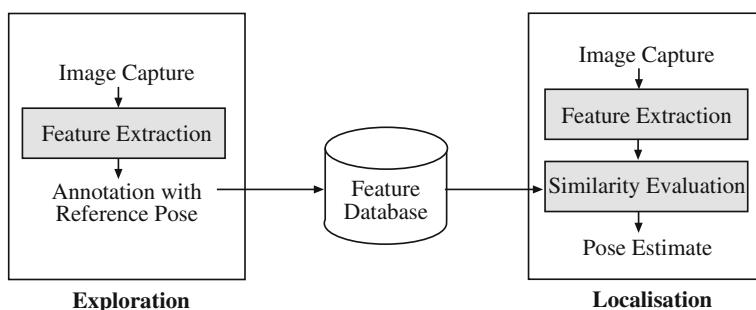
is able to connect to the quadrocopter. Because of these requirements we assembled a mobile phone, which supports all these functionalities, under the quadrocopter.

We use the N95 by Nokia with the operating system Symbian OS. Phones are embedded systems with limitations both in the computational facilities (low throughput, no floating point support) and memory bandwidth (limited storage, slow memory, tiny caches). The N95 is based on an Arm11 dual processor at 332 MHz. It contains 160 MB internal memory and an 8 GB memory card. Furthermore, the mobile device includes a 5-megapixel camera, a GPS sensor and data transfer techniques such as WLAN, infrared and Bluetooth. To apply these technologies, the mobile phone is programmable in Symbian C++, which offers a fast method to read the camera viewfinder images (sized  $320 \times 240$  pixels, QVGA) in 3 ms.

The images, taken during flight missions, are stored in an image database for later monitoring. The image features are stored in a relational database system. Symbian OS supports a database API in the database language SQL.

#### 4 Localisation Process

The localisation consists of two steps, an *exploration phase* and the actual *localisation phase* (Fig. 2). During the exploration phase, aerial images are taken. We extract characteristic features from these images and store them in a database. During the localisation phase, the quadrocopter navigates in an already known area. The mobile phone takes pictures and extracts their features. We compare these features to those



**Fig. 2** The process consists of an exploration phase and the actual localisation phase. The *highlighted* steps are under investigation in this paper

from the database. Similar features and therewith similar looking images are supposed to originate from the same location.

#### 4.1 Exploration Phase

The goal of the exploration phase is to map unknown areas in which the robot should be localised later. On exploration missions, the camera takes pictures of size  $320 \times 240$  pixels and annotates them with their recorded positions. We used GPS as ground truth, because the values did not significantly deviate from the real position, as it can be seen in Fig. 6. At this stage, the mobile application requests the GPS data from the quadrocopter via infrared connection. We decided not to use the mobile phone's GPS sensor, because it is restricted by a fee required signing process. The GPS information is stored in the *Exif*<sup>1</sup>-header of each captured *JPEG*-image. For the future, we plan to turn our mapping stage into a SLAM system in order to eliminate the depending on GPS. In this stage, however, we used GPS as ground truth in order to benchmark our approach.

During flight, the mobile application extracts features which describe image characteristics. Compared to robots on the ground, aerial systems differ in their movement in six DoF ( $\theta, \phi, \psi, x, y, z$ ). Image features have to be stable to changes of illumination and they have to cope with image transformations in six DoF. The camera performs large rotation and translation motions during flight, which can be seen in Fig. 3. It shows that images taken at the same location do not have to be similar. We examine to what extent global image features are capable of encoding image similarities. We expect that the localisation accuracy will suffer from intense image transformations, but that global image features should already lead to robust localisation. This limitation is compensated by the relative simplicity of the approach. Further, we expect that the features are fast to compute and need little memory, which enables the mobile computation.

In the following, we introduce the three global feature extraction algorithms.

##### 4.1.1 Grid Greyscale Histogram

A straightforward approach to characterize an image as a whole is to build a greyscale histogram. Ulrich and Nourbakhsh [28] established visual self-localisation for topological place recognition using color histograms. We extended the approach on two points. First, the illumination of the environment affects greyscale histograms. To obtain robustness to illumination changes, the grey value scale is divided into eight parts. Thus, one histogram bin represents 32 grey values. And second, partial occlusion has less impact on the histogram if the image is divided into subimages. By this means, changes of a subimage do not influence the whole histogram, but only the part of the histogram which represents the affected subimage. Therefore, we split the image into  $4 \times 4$  subimages. For each subimage, we then compute a separate histogram. Translations or rotations of the image cause pixels to fall into another subimage. Pixels in the centre of a subimage obtain a larger weight than those near the subimage border. The weighting is realized by a Gaussian function placed in the centre of each subimage.

<sup>1</sup>Exchangeable Image File Format



**Fig. 3 a–d** Images of a building under usual transformations

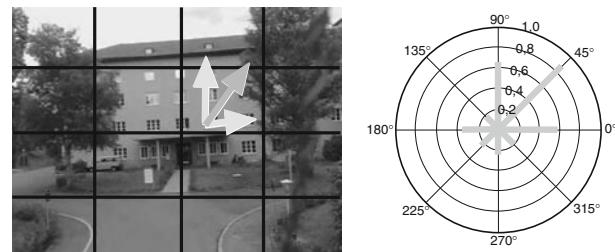
To get the final  $1 \times 128$  feature vector, the histograms of the subimages are concatenated. The vector is normalized according to the chosen norm (Section 4.2).

We expect the algorithm to be sensitive to changing lighting conditions, but at the same time, it is fast to compute.

#### 4.1.2 Weighted Grid Orientation Histogram

Bradley et al. [4] presented the *Weighted Grid Orientation Histogram* (WGOH) approach in 2005. It is inspired by the popular SIFT by Lowe [13]. The feature uses weighted histograms of image gradient orientations. In contrast to the local SIFT

**Fig. 4** Creation of the WGOH feature vector by determining the gradient orientation in each pixel and building an orientation histogram



feature, histograms are not computed around interest points, but globally for each pixel. We chose WGOH, because it was reported to obtain good results under different illumination conditions [4]. It benefits from the robustness of the SIFT idea and the simplicity of the global approach.

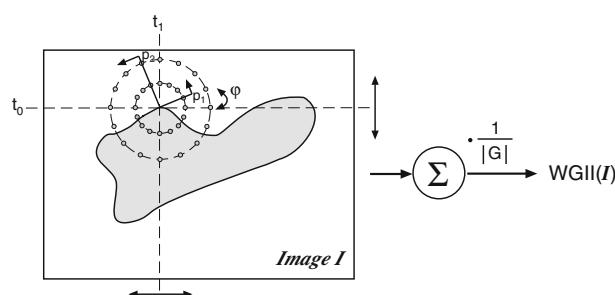
The algorithm works as follows (Fig. 4): The image is divided again into  $4 \times 4$  subimages. For each pixel the orientation of the gradient is calculated. A histogram of gradient orientations is built for each subimage. For this purpose, the orientations are divided into eight intervals. The gradient orientations are weighted by the magnitude of the gradient. Additionally, a Gaussian function is used for weighting each magnitude. The gradient histograms are concatenated to a  $1 \times 128$  vector, which is normalized subsequently.

#### 4.1.3 Weighted Grid Integral Invariants

The *Weighted Grid Integral Invariants* (WGII) approach by Weiss et al. [32] is based on Integral Invariants, which were first introduced by Manay et al. [15]. A detailed description of Integral Invariants can be found in [22]. Weiss et al. achieved good results under illumination changes. The idea is to apply as many transformations as possible to an image and to integrate the outcomes. Integration makes the features independent of the sequence of the transformations and leads to invariance against these transformations.

Integral Invariants proceed as follows (Fig. 5): A kernel function  $f$  involves the local neighborhood of each pixel. The function subtracts the intensities of two pixels lying on circles with different radii around the considered pixel. The two observed pixel positions are rotated 10 times. The kernel function is evaluated for each rotation  $g$ . Then, we integrate over these evaluations to achieve independence

**Fig. 5** The creation of the WGII feature vector by rotating a pair of neighbor pixels and averaging over the difference of their intensities



of the order of transformations. The Integral Invariant feature of the image  $\mathbf{I}$  of size  $N_0 \times N_1$  is calculated by

$$F(\mathbf{I}) = \frac{1}{RN_0N_1} \sum_{t_0=0}^{N_0-1} \sum_{t_1=0}^{N_1-1} \sum_{r=0}^{R-1} f\left(g\left(t_0, t_1, 2\pi \frac{r}{R}\right) \mathbf{I}\right), \quad (1)$$

where  $R$  is the number of rotations. Integral Invariants are applied to each pixel and weighted by a Gaussian function. The image is split in  $4 \times 4$  subimages. To calculate the Integral Invariants, we use two pairs of pixels  $(2; 0)^T, (0; 3)^T$  and  $(5; 0)^T, (0; 10)^T$  as kernels. The final feature is a histogram of the Integral Invariants. We use an 8-bin histogram per subimage. These histograms of sixteen subimages and both kernels are concatenated to obtain the final  $1 \times 256$  feature vector.

#### 4.2 Localisation Phase

During the localisation phase, the quadrocopter flies in an already visited area. The mobile application takes pictures and compares them with those in the database. To compare two pictures, it is not only important to find significant features, but to define a similarity measure to detect similar features. In the following, we define five measures which compare histograms, represented by vectors.

In [22], Siggelkow distinguished two concepts to compare feature histograms: a *bin-by-bin measure* and a *cross-bin comparison measure*. In the work at hand, we use bin-by-bin measures, which compare corresponding bins. The reason for this is that bin-by-bin measures have linear complexity in the number of bins, while cross-bin comparison measures have higher complexity. The notation of histograms corresponds to notation of vectors  $h = (h_0, h_1, \dots, h_{M-1})^T$ .

First, we remember the well-known Minkowski distance. It is defined by

$$d_{L_p}(h^{(0)}, h^{(1)}) = \left( \sum_{m=0}^{M-1} |h_m^{(0)} - h_m^{(1)}|^p \right)^{\frac{1}{p}}. \quad (2)$$

We consider three choices of  $p$ : the  *$L_1$ -norm* or *Manhattan norm* distance with  $p = 1$ , the *Euclidian distance* with  $p = 2$  and the *infinity norm* distance with  $p \rightarrow \infty$ :

$$\begin{aligned} d_{L_\infty}(h^{(0)}, h^{(1)}) &= \lim_{p \rightarrow \infty} \left( \sum_{m=0}^{M-1} |h_m^{(0)} - h_m^{(1)}|^p \right)^{\frac{1}{p}} \\ &= \max_{m=0, \dots, M-1} (|h_m^{(0)} - h_m^{(1)}|). \end{aligned} \quad (3)$$

The infinity norm performs bin-by-bin comparisons and considers the largest difference as the measurement value. If a single pair of bins is very different and all other bins are similar to each other, the pair of feature histograms is not regarded as a match.

Furthermore, Siggelkow introduced another distance with intuitive motivation. It is called the *intersection measure*  $d_{\cap}$  and calculates the common part of both histograms:

$$d_{\cap}(h^{(0)}, h^{(1)}) = \sum_{m=0}^{M-1} \min(h_m^{(0)}, h_m^{(1)}). \quad (4)$$

Bradley et al. [4] use the following measure to compare two normalized WGOH feature vectors:

$$d_{\text{Bradley}}(h^{(0)}, h^{(1)}) = 1 - (h^{(0)})^T h^{(1)}. \quad (5)$$

It is based on the fact, that the scalar product of two identical vectors is 1. The reverse does not hold: If a scalar product of two vectors is 1, the vectors do not have to be identical.

The intersection measure and Bradley's measure are not induced by norms. Thus, we normalized the feature vectors depending to the norm, which is the most suitable to the measures. We chose the  $L_1$ -norm for the intersection measure and the Euclidian norm for Bradley's measure, because experimentally, these combinations yielded the best results.

In the following we examine to what extent the measures influence the image feature approaches.

## 5 Experimental Results

In this section, we analyse the presented localisation system on the basis of two datasets and different configurations of the feature algorithms and similarity measures.

### 5.1 Datasets

The monocular camera of our aerial system faces forwards. Thus, a quadrocopter which rotates or varies its altitude, has many different views at the same latitude and longitude position. Besides, a large amount of data has to be recorded. To organize the exploration phase effectively and to minimize the database, we restrict our setting: According to the quadrocopter's task, the altitude and orientation are constrained. We collect data in an exemplary task, which is a flight from a start to an end point for a given route. The view of the camera pointed in course direction. The quadrocopter's altitude varied between 5 and 10 m, which is the range of the desired altitude. The images of the dataset were taken at different illumination conditions.

The dataset consists of 1,275 images in total, taken during a traversal of flying eight rounds in a courtyard. During an exploration mission, we took 348 training images of the courtyard with diffuse illumination (dataset T), representing one round. Dataset T was chosen to serve as our reference dataset, because it was a slow-moving exploration flight with twice the number of images as the test flights (see Table 1). Dataset C consists of 558 images taken at four rounds at cloudy weather. Dataset S contains 369 images, representing three rounds, which were taken on a sunny day. The images show hard shadows (Fig. 3a, left). The distance between images varies between the datasets. Figure 6 shows one round of dataset C and S, respectively.

**Table 1** Characteristics of the datasets

Dataset	Illumination	Number of images	Purpose
T	Cloudy	348	Exploration
C1	Cloudy	88	
C2	Cloudy	125	
C3	Cloudy	142	
C4	Cloudy	203	
S1	Sunny	94	
S2	Sunny	118	
S3	Sunny	157	
		558	Localisation
		369	Localisation

## 5.2 Results

In the following analysis, we evaluate the matching results of the visual localisation algorithms and the corresponding measures. GPS data were added to all datasets, serving as ground truth positions.

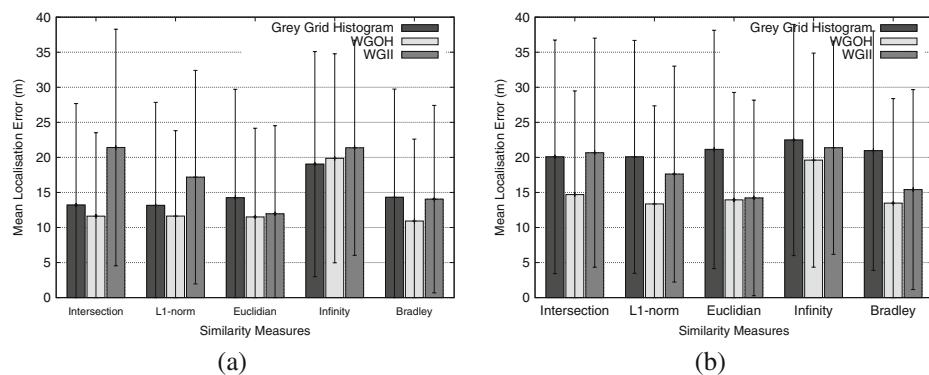
### 5.2.1 Comparison of Similarity Measures

In this section we analyse the matching accuracy for the different similarity measures. We measured the 2D position distance between the query image and the matched image for each feature algorithm and for each similarity measure.

The results of the different measures under varying illumination are shown in Fig. 7. The mean localisation error lies between 10.9 and 21.4 m. Considering this error, the appropriate choice of image feature and distance measure is important. With regard to the intense image transformations and the size of the test area of



**Fig. 6** The localisation area: The route is about 180 m in length. The captured images are placed in the aerial picture by their geotags. The two rounds can be clearly distinguished from each other (dataset C1 (white); dataset S1 (black)). GPS values are consistent and not falsified by erroneous signals. The distances between images vary because of gusts of wind



**Fig. 7** Comparison of the similarity measures combined with the different features. **a** The mean absolute localisation error and standard deviations under comparable illumination conditions during exploration and localisation (dataset C), while **b** is evaluated under different conditions (dataset S)

about 180 m in length, we consider the localisation error as good. An image taken close to a query image can look very different. However, an image which is taken from a 20 m distance can picture the same motif (Fig. 8).

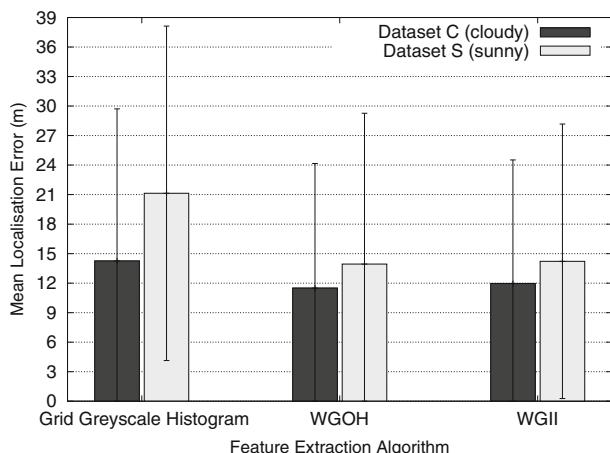
Figure 7a shows the results under similar illumination of training and test images. It illustrates the importance of the correct choice of distance metric for the particular feature extraction algorithm. The localisation error with the WGII approach varies about 10 m, depending on the chosen measure. In contrast, WGOH is only marginal influenced by the chosen measure. The measurements with the Euclidian norm and the Bradley measure (Eq. 5) perform with similar good results on all features. Each feature requires a different measure to yield the best performance. WGOH convinces in conjunction with Bradley's norm resulting in a localisation error of 10.92 m. WGII achieves best results (11.5 m) with the Euclidian norm. The grid greyscale histogram works best with the intersection measure (13.2 m) and the  $L_1$ -norm (13.15 m). The results of the infinity norm are worst. Here, only histograms are matched whose bins are without exception similar to its correspondent bins. This approach is not robust enough for our aerial images.

The experiments with dataset S under different illumination conditions are shown in Fig. 7b. The localisation error increases by 3 m on average. This difference is not caused by the measures, but by the feature extraction algorithms which are analysed in the following.



**Fig. 8** Combination (*right*) of a query (*left*) and a matched image (*middle*) which have a distance of 20 m to each other and which are congruent to nearly a third

**Fig. 9** Comparison of the different feature algorithms under the Euclidian norm and varying illumination conditions



### 5.2.2 Comparison of Feature Extraction Algorithms

Having found the most appropriate similarity measure for each approach, we concentrate on comparing the feature extraction algorithms.

As shown in Fig. 7, WGOH achieves the best results, followed by WGII. WGOH represents an image by its gradient orientations, while WGII examines the differences of neighbouring pixel intensities. Both approaches can cope with intense image transformations like rotation and translation to a certain extent. WGOH divides gradient orientations into eight intervals, which means that the feature is rotation-invariant up to 45°. WGII rotates its kernels ten times. Thus, it is invariant to 36°-rotations. Both features divide the image into  $4 \times 4$  subimages. Grid building helps being robust to occlusions.

Concerning the good results of the Euclidian norm, we restricted the following analysis of the influence of illumination changes by using the Euclidian norm (Fig. 9). WGOH yields worse results under heterogenous illumination during training and testing phase of 2.44 m. The localisation error of WGII increases by 2.26 m, and the one of the greyscale histogram by 6.88 m. WGOH makes use of gradients. Images taken at sunshine show deep shadows with gradients large in magnitude which exert influence on the WGOH feature. WGII is least affected by illumination changes. It compares only the difference between neighbouring pixel intensities. Varying illumination has only little effect on the difference in intensity. For instance, plain-coloured areas likely remain plain-coloured despite illumination variations. The grid greyscale histogram shows surprisingly good results under similar lighting conditions.

**Table 2** Computing time [s]

		WGOH	WGII	Histogram
a	Feature calculation	0.49	4.39	0.16
b	Database creation	0.66	0.66	0.66
c	Single feature comparison to a database of 348 entries	0.15	1.17	0.15
a + b	Exploration	1.15	5.05	0.82
a + c	Localisation	0.64	5.56	0.31

In conjunction with the intersection and  $L_1$ -norm, the greyscale histogram works better than WGII. But it suffers under different illumination conditions, because it is solely based on greyscale values.

### 5.2.3 Computation Times

Table 2 shows the computation time required by the different algorithms for the Euclidian measure. The greyscale histogram is the fastest approach, followed by WGOH and finally WGII. During localisation phase, the histogram needs 0.31 s and WGOH needs 0.64 s to process an image. WGII requires about 5.5 s to compute and match features, and hence more than eight times as long as WGOH. This can be improved by calculating Integral Invariants in every forth randomized pixel location in the image. In this case, the mean localisation error worsens from 11.96 to 12.83 m, but the time consumption of the feature extraction reduces from 4.4 to 1.3 s.

## 6 Conclusions

In this research, we visually localised a small UAV with techniques applied onboard in a large (180 m) outdoor environment. We used only a mobile phone as visual sensor and onboard vision processing computer. We tested the system under image transformations and illumination changes with different algorithm and similarity measure configurations. The feature extraction algorithm WGOH and the feature comparison measure Euclidian norm worked best. The computing time in the exploration phase is 1.15 s, the localisation phase takes 0.64 s. The mean localisation error is 10.92 m. This is comparable to Valgren's [29] results in large outdoor environments. Valgren et al. defined a threshold at a distance of 10 m between match and query image to classify a match as correct.

We see great potential in using our system in real-world applications. Once an environment has been explored with accurate GPS position values, the system does not depend on GPS anymore. If GPS is unavailable or noisy, we can switch to the visual localisation system. We will concentrate on a SLAM system to eliminate the GPS-dependent exploration phase.

In future work we focus on the acceleration of the localisation process. Our prototype implementation can be improved by integer optimized algorithms and SIMD instructions.

Further work includes expanding the experiments to a larger environment. In addition, a particle filter will be used to improve the localisation error.

## References

1. Artac, M., Leonardis, A.: Outdoor mobile robot localisation using global and local features. In: Proceedings of the 9th Computer Vision Winter Workshop (CVWW), pp. 175–184. Piran, Slovenia (2004)
2. Bath, W., Paxman, J.: UAV localisation and control through computer vision. In: Proceedings of the Australasian Conference on Robotics and Automation. Sydney, Australia (2004)
3. Bay, H., Ess, A., Tuytelaars, T., Gool, L.: Surf: speeded up robust features. In: Computer Vision and Image Understanding (CVIU), vol. 110, pp. 346–359 (2008)

4. Bradley, D., Patel, R., Vandapel, N., Thayer, S.: Real-time image-based topological localization in large outdoor environments. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 3670–3677. Edmonton, Alberta, Canada (2005)
5. Castle, R., Gawley, D., Klein, G., Murray, D.: Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In: Proceedings of the International Conference on Robotics and Automation (ICRA), pp. 4102–4107. Rome, Italy (2007)
6. Corke, P., Strelow, D., Singh, S.: Omnidirectional visual odometry for a planetary rover. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), vol. 4, pp. 4007–4012. Sendai, Japan (2004)
7. De Wagter, C., Proctor, A., Johnson, E.: Vision-only aircraft flight control. In: The 22nd Digital Avionics Systems Conference (DASC), vol. 2, pp. 8.B.2–81–11 (2003)
8. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.M., Hirzinger, G., Rus, D.: Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 361–366. Roma, Italy (2007)
9. Hofmeister, M., Liebsch, M., Zell, A.: Visual self-localization for small mobile robots with weighted gradient orientation histograms. In: 40th International Symposium on Robotics (ISR), pp. 87–91. Barcelona, Spain (2009)
10. Jogan, M., Leonardis, A., Wildenauer, H., Bischof, H.: Mobile robot localization under varying illumination. In: 16th International Conference on Pattern Recognition (ICPR), vol. II, pp. 741–744. Los Alamitos, CA, USA (2002)
11. Kim, J., Sukkarieh, S.: Real-time implementation of airborne inertial-SLAM. In: Robotics and Autonomous Systems archive, vol. 55, pp. 62–71. Amsterdam, The Netherlands (2007)
12. Lamon, P., Nourbakhsh, I., Jensen, B., Siegwart, R.: Deriving and matching image fingerprint sequences for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1609–1614. Seoul, Korea (2001)
13. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision (ICCV), pp. 1150–1157. Corfu (1999)
14. Maimone, M., Cheng, Y., Matthies, L.: Two years of visual odometry on the mars exploration rovers: field reports. Journal of Field Robotics **24**(3), 169–186 (2007)
15. Manay, S., Hong, B.-W., Yezzi, A.J., Soatto, S.: Integral invariant signatures. In: Proceedings of the 8th European Conference on Computer Vision (ECCV). LNCS, vol. 3024, pp. 87–99. Prague, Czech Republic (2004)
16. Mondragon, I., Campoy, P., Correa, J., Mejias, L.: Visual model feature tracking for UAV control. In: IEEE International Symposium on Intelligent Signal Processing (WISP), pp. 1–6. Alcala de Henares (2007)
17. Musial, M., Brandenburg, U., Hommel, G.: Cooperative autonomous mission planning and execution for the flying robot MARVIN. In: Intelligent Autonomous Systems, vol. 6, pp. 636–643. Amsterdam, The Netherlands (2000)
18. Pinies, P., Lupton, T., Sukkarieh, S., Tardos, J.D.: Inertial aiding of inverse depth SLAM using a monocular camera. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA), pp. 2797–2802. Rome, Italy (2007)
19. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Vision-based autonomous landing of an unmanned aerial vehicle. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2799–2804. Washington, DC, USA (2002)
20. Scaramuzza, D., Siegwart, R.: Appearance guided monocular omnidirectional visual odometry for outdoor ground vehicles. In: IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1015–1026 (2008, appearance – guided)
21. Scaramuzza, D., Siegwart, R.: Monocular omnidirectional visual odometry for outdoor ground vehicles. In: Lecture Notes in Computer Science, chap. Computer Vision Systems, vol. 5008/2008, pp. 206–215. Springer, New York (2008)
22. Sigalkow, S.: Feature histograms for content-based image retrieval. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg, Fakultät für Angewandte Wissenschaften, Freiburg, Germany (2002)
23. Sim, R., Dudek, G.: Comparing image-based localization methods. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1560–1562. Apaculco, Mexico (2003)
24. Sinopoli, B., Micheli, M., Donato, G., Koo, T.J.: Vision based navigation for an unmanned aerial vehicle. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 1757–1765. Seoul, Korea (2001)

25. Steder, B., Rottmann, A., Grisetti, G., Stachniss, C., Burgard, W.: Autonomous navigation for small flying vehicles. In: Workshop on Micro Aerial Vehicles at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). San Diego, CA, USA (2007)
26. Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., Girod, B.: Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In: Proceedings of the 1st ACM international conference on Multimedia information retrieval (MIR), pp. 427–434. New York, NY, USA (2008)
27. Tamimi, H.: Vision-based features for mobile robot localization. Ph.D. thesis, Eberhard-Karls-Universität Tübingen, Tübingen, Germany (2006)
28. Ulrich, I., Nourbakhsh, I.: Appearance-based place recognition for topological localization. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 1023–1029. San Francisco, CA, USA (2000)
29. Valgren, C., Lilienthal, A.: SIFT, SURF and seasons: long-term outdoor localization using local features. In: Proceedings of the European Conference on Mobile Robots (ECMR), pp. 253–258. Freiburg, Germany (2007)
30. Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., Schmalstieg, D.: Pose tracking from natural features on mobile phones. In: 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), pp. 125–134. Cambridge, UK (2008)
31. Wang, J., Zhai, S., Canny, J.: Camera phone based motion sensing: interaction techniques, applications and performance study. In: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST), pp. 101–110. New York, NY, USA (2006)
32. Weiss, C., Masselli, A., Tamimi, H., Zell, A.: Fast outdoor robot localization using integral invariants. In: Proceedings of the 5th International Conference on Computer Vision Systems (ICVS). Bielefeld, Germany (2007)
33. Williams, B., Klein, G., Reid, I.: Real-time SLAM relocation. In: IEEE 11th International Conference on Computer Vision (ICCV), pp. 1–8. Rio de Janeiro, Brazil (2007)
34. Zhou, C., Wei, Y., Tan, T.: Mobile robot self-localization based on global visual appearance-based features. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1271–1276. Taipei, Taiwan (2003)

# A Rotary-wing Unmanned Air Vehicle for Aquatic Weed Surveillance and Management

Ali Haydar Göktoğan · Salah Sukkarieh ·  
Mitch Bryson · Jeremy Randle · Todd Lupton ·  
Calvin Hung

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 19 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** This paper addresses the novel application of an autonomous rotary-wing unmanned air vehicle (RUAV) as a cost-effective tool for the surveillance and management of aquatic weeds. A conservative estimate of the annual loss of agricultural revenue to the Australian economy due to weeds is in the order of A\$4 billion, hence the reason why weed control is of national significance. The presented system locates and identifies weeds in inaccessible locations. The RUAV is equipped with low-cost sensor suites and various weed detection algorithms. In order to provide the weed control operators with the capability of autonomous or remote control spraying and treatment of the aquatic weeds the RUAV is also fitted with a spray mechanism. The system has been demonstrated over inaccessible weed infested aquatic habitats.

**Keywords** Rotary-wing unmanned aerial vehicle (RUAV) · Civilian UAV applications · Ecological research · Weed surveillance · Weed management

---

A. H. Göktoğan (✉) · S. Sukkarieh · M. Bryson · J. Randle · T. Lupton · C. Hung  
ARC Centre of Excellence for Autonomous Systems, Australian Centre for Field Robotics,  
School of Aerospace, Mechanical, and Mechatronic Engineering, The University of Sydney,  
Sydney, 2006, New South Wales, Australia  
e-mail: a.goktogan@acfr.usyd.edu.au

S. Sukkarieh  
e-mail: s.sukkarieh@acfr.usyd.edu.au

M. Bryson  
e-mail: m.bryson@acfr.usyd.edu.au

J. Randle  
e-mail: j.randle@acfr.usyd.edu.au

T. Lupton  
e-mail: t.lupton@acfr.usyd.edu.au

C. Hung  
e-mail: c.hung@acfr.usyd.edu.au

## 1 Introduction

Weeds are one of the most serious threats to the Australian ecological environment, to rangelands and primary production industries. Weeds hinder the growth of the native species, cause land degradation, and reduce farm and forest productivity. A conservative estimate of the annual cost of weeds to the Australian economy as loss in agricultural revenue is in the order of A\$4 billion [1]. Apart from the economic impact weeds also have an adverse affect on the surrounding fauna, flora, biodiversity, native landscape, irrigation, and any supporting tourism. Hence the reason why weed control is of national significance.

An unmanned aerial system (UAS) has been developed as a cost-effective tool for the surveillance and management of weeds. The UAS consists of a rotary-wing unmanned aerial vehicle (RUAV) equipped with a navigation subsystem, a control subsystem, an image acquisition subsystem, a spray mechanism, a ground station, a communication subsystem, and a surveillance data processing subsystem with aquatic weed detection algorithms.

The paper is organised as follows: Section 2 examines related works. Section 3 briefly explains what a weed is in our context, and which weeds we are going to address in our work. Section 4 explains the sequence of the operational processes of a typical aquatic weed surveillance and management mission. Section 5 presents the overall system architecture and explains the primary subsystems. Section 6 provides examples from our field trials. Section 7 presents the weed detection, classification and mapping algorithm. Section 8 addresses the georeferencing process. Finally, Section 9 draws the conclusions and addresses future work.

## 2 Related Works

Although unmanned aerial vehicles (UAVs) have been used for military applications since the First World War [2], their civilian applications [3–5], particularly their use in agriculture [6], rangeland and ecological research [7] is relatively new. The high cost of the UAV platforms, safety concerns, and the need for specialised operating personnel are some of the major reasons behind the delayed introduction of UAVs to ecological research. The increasing availability of more reliable and low-cost UAV platforms offers new possibilities.

Aerial photography is the most common UAV application in rangeland and ecological research. The UAVs used in these application varies from low-cost model aircrafts [8] to custom built, state-of-the-art aerial platforms [6, 9].

As a low-cost solution, a commercial-off-the-shelf (COTS), prebuilt airframe of a radio-controlled (RC) model aircraft has been demonstrated in [8] for aerial rangeland photography. This platform was selected for its slow flight speed and inherent stability. The model was fitted with a COTS flight stabiliser, a barometric altitude-hold module from the RC hobby sector, a single-lens reflex (SLR) camera, and a global positioning system (GPS) receiver for geolocation of the aerial photos. Although the model aircraft was lacking an autopilot and on-board navigation unit for accurate attitude determination it was demonstrated that even low-tech solutions can be useful for some applications.

At the other extreme, [6, 9] addresses precision agriculture using a solar-powered, Pathfinder-Plus UAV developed by NASA and AeroVironment. This high-tech, high-altitude flying UAV is equipped with multispectral and hyperspectral imaging systems and was demonstrated over the Kauai Coffee Plantation.

The above mentioned works focused on the surveillance of large rangelands. The work presented in this paper differs in that the RUAV is not only used for aerial imagery of weed infested aquatic habitats, but also for spot spraying the designated locations. Furthermore, the presented work incorporates an automated weed detection and classification algorithm as well.

Another distinguishing factor is that the RUAV has multiple modes of operations: manual control mode, semi-autonomous, and autonomous flight modes. Although the current aviation safety rules in Australia limit the RUAV operations to be performed only by qualified UAV operators, the flexibility in proving various operational modes increases the future possibility of operating the RUAV without specialised UAV personnel.

Remote sensing and satellite imagery have been used in environmental research for a long time. Typical satellite imagery available to the civilian domain has approximately 15–45 m resolution per pixel. These images are useful for a high level overview of very large landscapes.

Remote sensing data from human piloted aircrafts can provide better than 1 m spatial resolution imagery. This is useful for providing ground truth for the satellite data as well as investigation of relatively large rangelands. Although, it is possible to acquire even higher resolution imagery from human piloted aircrafts by using higher resolution cameras and flying at lower altitude, these options introduce their own draw backs and in particular safety concerns as human piloted aircrafts cannot safely operate in narrow corridors around natural obstacles such as valleys, trees or bushes, even if ultralight aircrafts [10] are used.

Hence, when centimetre or better resolution is desired from aerial imagery, UAVs can provide an effective *close sensing* solution without putting human operators into risk.

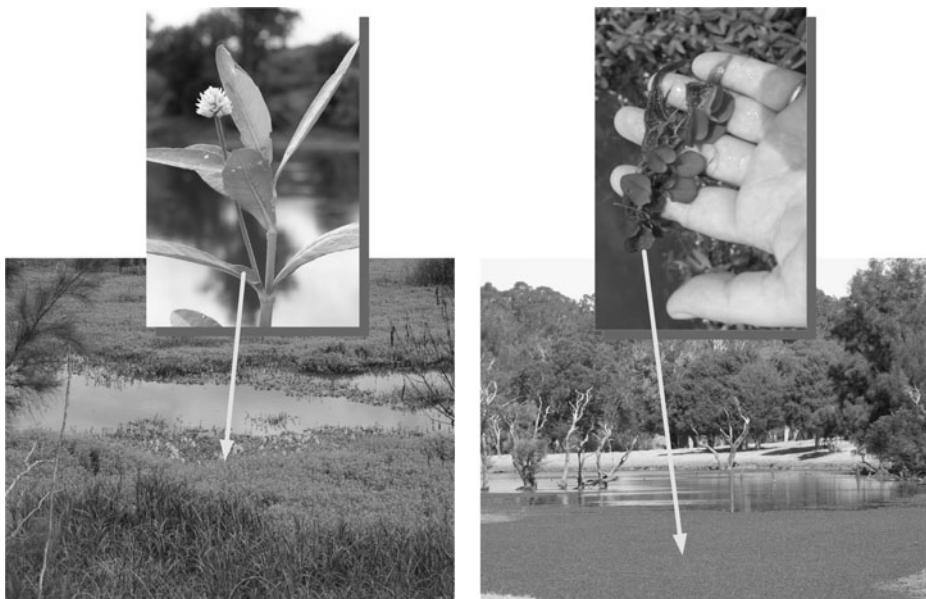
### 3 What is a Weed?

Weeds are rapidly spreading invasive plants which have an adverse effect on the environment and can have serious economic impact. Although some native plants can also become weeds if the right conditions arise, the overwhelming majority of weeds in Australia are those which are introduced into Australia from other continents, mainly due to human activity.

In our work we have focused on alligator weed and salvinia. Both of them are considered as “Weeds of National Significance”, because of their devastating impact to the environment and economy. Figure 1 shows examples from alligator weed and salvinia infested aquatic habitats.

#### 3.1 Alligator Weed

Alligator weed (*alternanthera philoxeroides*) [11] is a native plant of the Parana River system in north-eastern Argentina. It was first reported in Australia in the 1940s and



**Fig. 1** “Weeds of National Significance”; (left image) an irrigation channel blocked by Alligator Weed (*alternanthera philoxeroides*), (right image) a pond severely covered by *Salvinia (salvinia molesta)*

believed to have been introduced in shipping ballast. Alligator weed continues to threaten to spread along the east coast of Australia.

Alligator weed is extremely invasive, and has high potential for rapid spread to large areas. Furthermore it can also invade both aquatic and land habitats. Control of alligator weed is very hard and has a very high economic and environmental impact.

### 3.2 Salvinia

*Salvinia (salvinia molesta)* [11] originated from south-east Brazil as an aquarium plant and was first recorded in Australia in 1952. *Salvinia* can cause devastating and severely affect riparian ecosystems, water quality, wildlife and the surrounding primary industries.

Given the right conditions *salvinia* exhibits an extraordinary growth rate; it can double its dry weight in two and a half days. With such a growth rate it does not take too long to block waterways, irrigation channels, smother dams and reservoir surfaces. The sun light cannot penetrate the *salvinia* covered layer. This eventually causes the underwater plants and the fauna to die. Holm and East-West Center [12] describes *salvinia* as one of the world’s worst weed.

## 4 Aquatic Weed Surveillance and the Management Process Flow

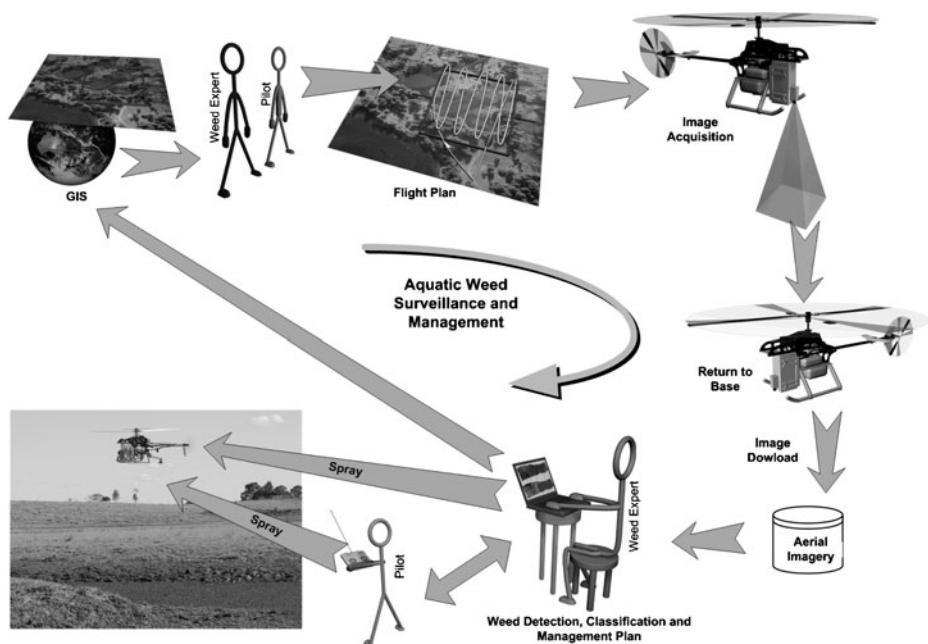
We performed our early RUAV based aquatic weed surveillance and management demonstrations over a weed infested aquatic habitat near the Richmond area, north-

west of Sydney in 2008. These areas are often closed-off by dense bushes and known for the spread of alligator weed and salvinia. Snakes, spiders and other dangerous animals add extra difficulties for human access for surveillance and management of weeds. In practical terms these habitats are inaccessible neither from the ground or through water ways and so consequently weeds flourish.

As shown in Fig. 2, a typical weed surveillance mission starts with some a-priori estimate of the weed location given by human weed experts. These initial estimates are often based on a geographical information system (GIS), environmental conditions, weed sightings, and/or data from satellite imagery, aerial imagery obtained from manned flights, but above all, it is based on the tacit knowledge of the subject matter experts with years of experience in the field.

Based on the draft situation model and environmental constraints, such as topography of the region, obstacles, wind directions and so forth, flight plans are prepared on the ground and uploaded to the RUAU autopilot for autonomous surveillance. Alternatively, a human pilot remotely controls the RUAU to visit the suspected regions for data collection.

The collected data is then uploaded to a ground based computer for weed detection, classification, and mapping process. Human experts do play a major role and supervise the computer algorithms for the detection and classification of the weeds. A number of different machine learning techniques are being studied. The current version of the weed detection and classification system is based on Support Vector Machines (SVM) which are used to create weed infestation maps that is fed back to the GIS and weed authorities for future planning.

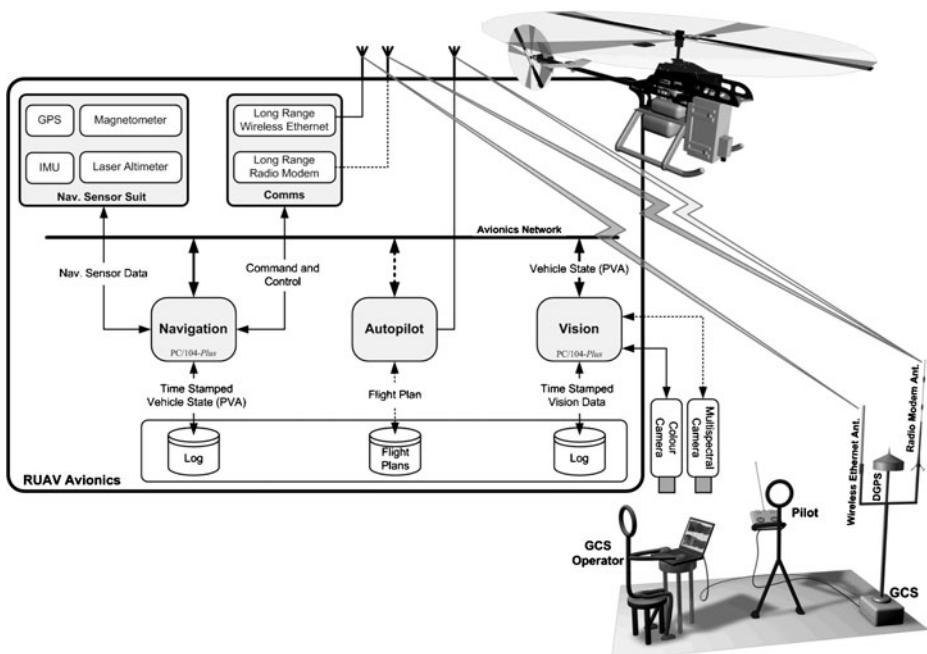


**Fig. 2** Aquatic weed surveillance and management process flow

Once the weed infested regions are mapped, human experts' are then used to define potential points for the RUAV to spot spraying. Current civil aviation rules and regulations imposed by the Australian Civil Aviation and Safety Authority (CASA) have tight restrictions on the spraying of chemicals from aircrafts. The relevant acts provide details about the endorsement of the aircraft and the human pilot for the spraying operations. Obviously these rules cannot be applied directly to unmanned operations. Therefore in order to demonstrate the proof-of-concept of using the RUAV for weed control in our experiments without infringing the laws we used water soluble non-toxic dye instead of real herbicides.

## 5 Architecture

This section describes the hardware and software architecture of the RUAV. As shown in Fig. 3 the RUAV is fitted with a navigation computer, a vision sensor computer, and an autopilot module. The navigation sensor computers are both in PC/104-Plus form factors, and the autopilot module is built around an embedded microcontroller board. They are linked together with an onboard network and they are also able to communicate to the ground control station (GCS) via the communication subsystem.



**Fig. 3** Representation UAS and the avionics architecture of the RUAV used in the aquatic weed surveillance and management demonstrations

## 5.1 RUAV Avionics

### 5.1.1 Navigation Subsystem

In order to accurately localise and georeference the weeds and other feature of interests (FoI) in the aerial imagery an accurate estimate of the RUAV position and attitude are needed. In our system architecture the navigation computer provides this estimate at rate of up to 20 Hz.

As illustrated in Fig. 3 the navigation subsystem is connected to navigation sensor suite. It consists of a Novatel OEM4-GSL-RT20 [13] global positioning system (GPS) receiver, an inertial measurement unit (IMU), a Honeywell HMR2300 [14] magnetometer which provides high accuracy magnetic heading information, and a downward pointed MDL LaserACE IM35 [15] laser range finder for determining height above ground.

The navigation computer reads the sensor data and calculates the navigation solution. Both the raw sensor data and the navigation solution is logged on an onboard disk. All the logged data is accurately time stamped for further analysis. The navigation solution is also published onto the avionics network for other subsystems.

The navigation data is also available to the ground control station (GCS). The GCS uses the navigation solution to update the GCS screen at real-time to indicate the position and attitude of the RUAV to maximise the situation awareness of the GCS operator and the pilot.

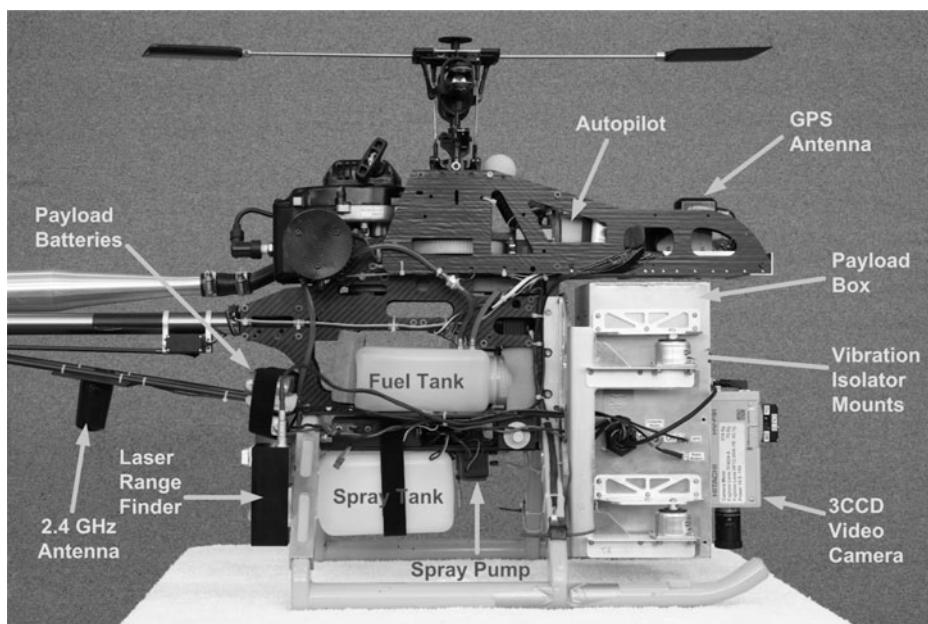
### 5.1.2 Image Acquisition Subsystem

Early in the project many experiments were conducted to test various ideas on the type of sensors and the detection algorithms. This information was needed to determine the physical size and weight of the surveillance system which in turn governed how it would be mounted on the RUAV. A number of colour cameras and multi-spectral cameras with infrared (IR) and near-infrared (NIR) bands were investigated. Finally, the Hitachi HV-F31 progressive scan 3 CCD [16] camera was selected.

The image acquisition subsystem consists of a PC/104-*Plus* computer and a digital video camera interfaced to each other via IEEE 1394 (FireWire). The Hitachi HV-F31 has  $1028 \times 764$  pixel spatial resolution and provides separate Red, Green, and Blue frames.

A Fujinon Lens TF4 DA-8 with 4 mm focal length is attached to the camera. This configuration gives a field of view of approximately 60° in the horizontal and 40° in vertical directions. The image acquisition computer is mounted inside the payload box and the camera is mounted to the forward faced surface looking down. Figures 3 and 4 illustrate the position of the camera and its relative orientation with respect to the RUAV body.

The image acquisition subsystem is configured to capture at 3 frames per second. The captured images are saved onto the onboard disk with a high resolution time stamp. The time stamp information is later used to match the navigation solution record for geo-referencing of the image frame as well as geo-referencing the detected weed in the frame.



**Fig. 4** Close-up view of the RUAU frame with the payload box and onboard electronics

The changing of the camera unit with another IEEE 1394 compatible camera is a relatively straight forward operation in terms of software configuration and/or modification. However, physical size, weight and the electrical power consumption of the new camera has to be examined carefully.

### 5.1.3 Autopilot Subsystem

The objective of the autopilot subsystem (flight control subsystem) is to provide a stabilised platform that could then be commanded to move to defined locations. We used a COTS, European made autopilot unit. This unit had its limitations and problems which only became apparent after testing.

With this flight control system (Fig. 4) the platform could be commanded to move along straight line paths to geo-referenced way points. This way the operations were conducted to suit its limitations. This impacted on what could be achieved in terms of flight regimes. For example, we were confident in flying straight channels, or over large open spaces, but less confident in following a tight meandering river. Even though we were not aiming to conduct experiments which involved difficult spraying manoeuvres or to follow tight curves, it was something that we hoped to test. This did not affect the surveillance system or its results.

### 5.1.4 Communication Subsystem

The current payload box is designed to accommodate mainly two different type of COTS communication units; a 2.4 GHz spread spectrum, long-range wireless

Ethernet transceiver or a 900 *MHz* long range radio modem. Depending on the bandwidth requirements of the data down-link either of these communication units can be used.

Figure 4 shows a configuration of the RUAV carrying a 2.4 *GHz* antenna. In our current configuration all interaction between the GCS and the onboard navigation and vision computers occurs over the 2.4 *GHz* wireless Ethernet link. The autopilot module illustrated in Fig. 3 uses its own in-built transceiver to communicate with the GCS terminal.

## 5.2 The RUAV Platform

The first prototype of the RUAV was based on the G18 Industrial Helicopter which was designed and manufactured in Australia. A range of major modifications have been made to increase the reliability, endurance, and payload capacity. Using an Australian manufacturer allowed us to easily liaise with the manufacturer to modify the helicopter for our specific requirements.

An on board telemetry system was also added to enable us to monitor parameters which were important to the health of the system. These included temperatures, voltages and rotor RPM.

The basic specifications of the RUAV platform are:

Overall Length: 2.0 m  
Overall Height: 0.63 m  
Main Rotor Diameter: 1.8 m  
Max Take Off Weight: 15 kg  
Engine: 26cc *Zenoah* (modified)  
Speed: 0 to 100 km/h  
Endurance: 20 min to 2 h (payload dependant)

The front payload area was modified to accommodate our navigation, autonomous flight control system and associated vibration isolation mounts. Additional mounting points were also installed for the GPS antenna, the magnetometer and the communications antenna.

A new payload mounting system was added to the front of the helicopter to allow the installation of our camera system, its associated imaging processing equipment and the radio communications system. This required a complex vibration isolation system to reduce the vibration to an acceptable level for the camera system and prevent motion blur of the images. The isolators had to be symmetrically located around the centre of gravity of the payload to ensure the rolling and pitching motions were not introduced by the vibration. This added significant weight to the helicopter. Further development of this mount would reduce its weight and increase the duration of the RUAV.

Additional mounting points were installed in the undercarriage legs to allow a spray boom to be attached to each side of the helicopter. A controlled droplet applicator (CDA) was attached to the end of each boom. The boom design was modified to reduce vibration caused by the rotor down draft. Ball joints were installed

and the boom hinges to allow for movement of the spray booms when in contact with the ground during takeoff and landing.

A mounting tray was designed and attached to the underside of the fuel tank. This contained the spray liquid tank, spray pump, control system and power supply. The spray tank was located under the centre of gravity of the helicopter so it did not effect the balance as the spray liquid was used. For demonstration purposes a dye was added to water to enable visualisation of the spray patterns and area covered.

The modified platform proved to be very robust. The spray mechanism can be seen in Fig. 5 during a flight test of the RUAV. Figure 4 shows a close-up view of the main body and major components of the RUAV.

The actual configuration of the helicopter depended on the mission required for each area of research. Initially the navigation and autonomous flight control system, the imaging system and the spray system were all tested individually. The imaging system and spray system were then combined with the flight control system for autonomous operations.

For demonstration purposes the helicopter was flown with the imaging system and spray system simultaneously. This configuration had an endurance of approximately 30 min. A commercial system would most likely be reconfigured to have two helicopters, one for weed detection and another for spraying.

This would increase the endurance of the detection helicopter to approximately 2 h with the current imaging system and more with a purpose designed system. The

**Fig. 5** The RUAV during hovering over a weed patch at the University of Sydney's flight test facility at Marulan ( $34^{\circ}35'40''S$ ,  $150^{\circ}3'19''E$ ) New South Wales, Australia



spraying helicopter could then carry in excess of four litres of spray and would only be required to operate if weeds were detected.

## 6 Field Trials

The pre-mission testing of the system was done at the University of Sydney's flight test facility at Marulan ( $34^{\circ}35'40''S$ ,  $150^{\circ}3'19''E$ ) New South Wales, Australia. Multiple flight tests were conducted to determine the stability of the platform as well as ensuring that all electronics could handle the vibration environment of the system. As expected each flight test taught us something new about the system and its capability eventually leading to the working system. Figure 5 shows the RUAV in flight at Marulan, carrying a gimbal unit and the sprayer mechanism.

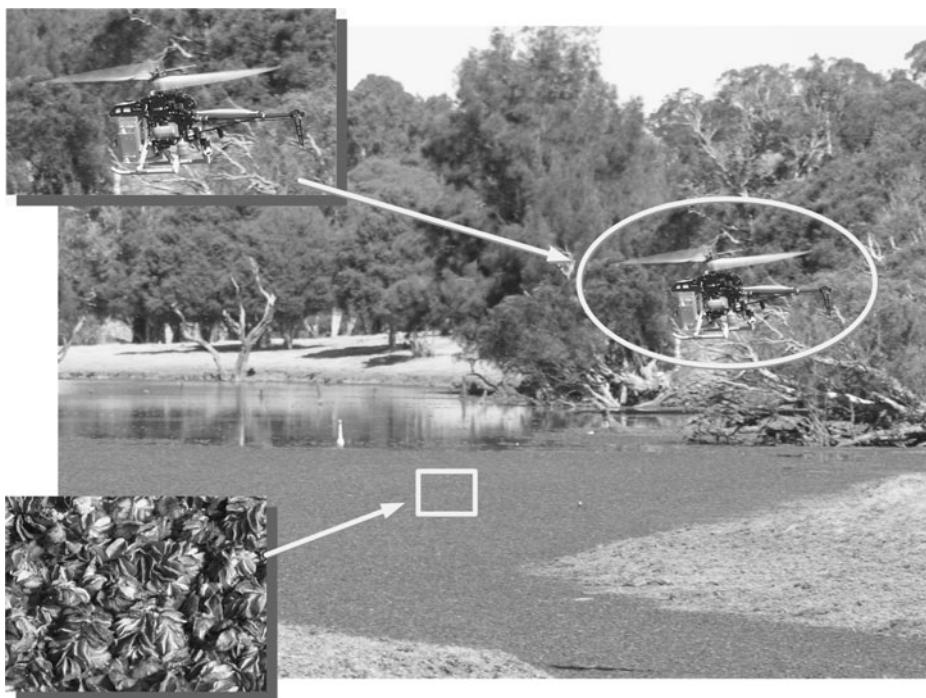
We performed our first set of weed detection trials at the Killarney Chain of Ponds in Pitt Town. Figure 6 shows a typical panoramic view from the ponds. This area is known for its spread of Alligator weed and Salvinia.

The trial was conducted on April 2008 on one of the farms that we had access to where only Alligator Weed was available. Figure 7 shows an instance from these tests. The system worked as planned, being able to fly around the aquatic site, collect imagery, communicate to the ground station, and spray water soluble dye at designated locations to simulate application of herbicide.

A set of experiments were also conducted in August 2008 at around the same region where the weeds were already sprayed by a local contractor. The test conducted with the detection focussed on classifying "sprayed Salvinia". Thus, the only theoretical difference to a normal outbreak would be the colour and a slight change in texture. The results for Salvinia look promising although more work is required when the weed is in its active period. It was also a successful trial, demonstrating that the system could fly along the creek, detect and spray. The project demonstration video can be seen on [17].



**Fig. 6** A stitched panoramic view of an aquatic weed infestation of a water way at the Killarney Chain of Ponds in Pitt Town north-west of Sydney, Australia



**Fig. 7** The RUAU flying (*upper inset*) over the *Salvinia* infested habitat (*lower inset*)

## 7 Weed Detection, Classification and Mapping

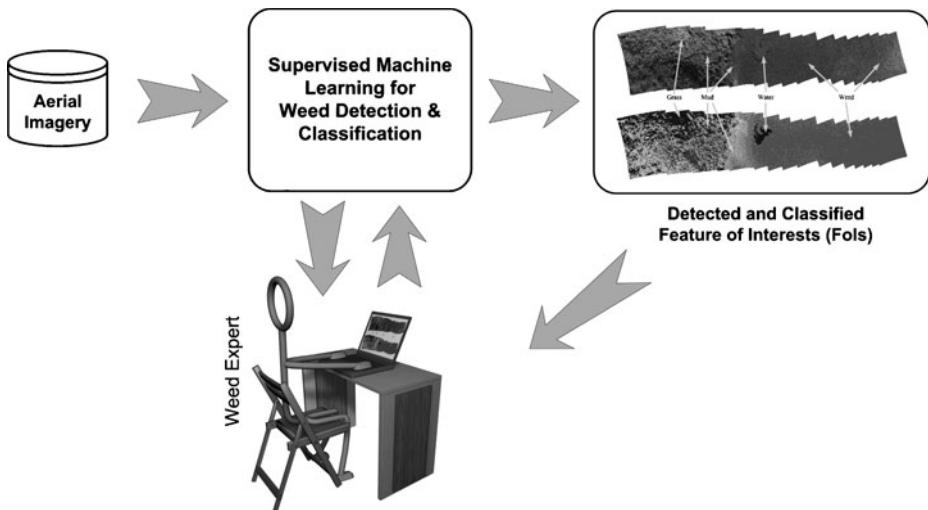
As was highlighted in Section 4, a weed experts' input into the weed detection, classification and mapping process is an important requirement for our work. In order to utilise the tacit knowledge of weed experts we concentrated on using supervised machine learning techniques.

Figure 8 illustrates the role of the human weed expert in supervision of the weed detection and classification process. The human weed expert uses the aerial imagery of the weed infested habitats and interactively highlights a number of small regions as features of interest (FoI). The supervised machine learning algorithm takes these samples and processes the rest of the imagery to detect the same or similar FoIs. It reports its findings back to human expert as probability distributions.

### 7.1 Support Vector Machine

We have investigated and applied a number of machine learning techniques. Our initial experiments with the Support Vector Machine (SVM) provided promising results. There is also literature in which SVM was used in ecological research [18].

The aim of SVM is to construct a hyperplane  $H$  used to separate two classes of data points with maximum margin [19–21]. The margin is maximised by using two parallel hyperplanes  $H_1$  and  $H_2$  pressing against the most outlying data points from



**Fig. 8** The process of supervised machine learning to weed detection and classification

each class during the training process, these points are therefore called the support vectors.

For data points to be in class one, they must sit above the hyperplane  $H_1$ , and for points to be in class two they must lie below hyperplane  $H_2$ . These are the constraints of the problem (Fig. 9).

$\mathbf{x} \cdot \mathbf{w} + b \geq +1$ ,  $y_i = +1$  and  $\mathbf{x} \cdot \mathbf{w} + b \leq -1$ ,  $y_i = -1$ . These constraints can be combined into one single constraint

$$y_i(\mathbf{x} \cdot \mathbf{w} + b) - 1 \geq 0, \forall i \quad (1)$$

Also, maximising the margin  $\frac{2}{\|\mathbf{w}\|}$  is equivalent to minimising  $\|\mathbf{w}\|$ . This is an optimization problem and can be solved using the Lagrangian Formulation.

$$L_P(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^l \alpha_i \quad (2)$$

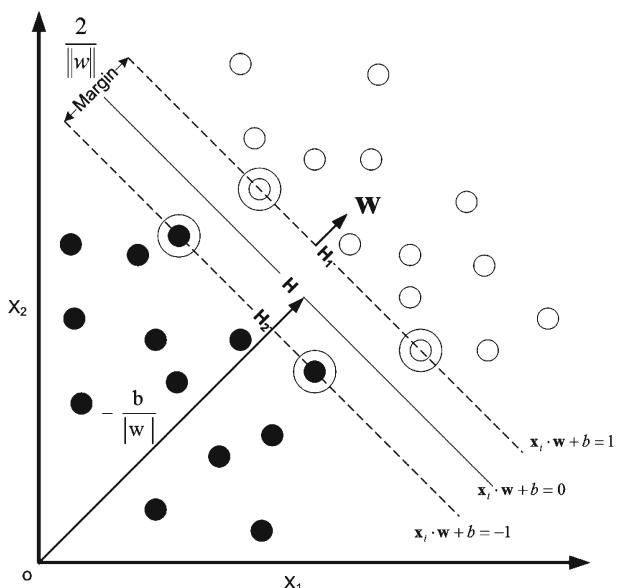
$$L_D(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^l \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (3)$$

Equation 2 is the Lagrangian in Primal Form, whereas Eq. 3 is the Lagrangian in Dual Form. The value of  $\alpha$  is solved in Lagrangian Dual form, then the hyperplane parameters  $w$  and  $b$  can be solved in the primal form during the training process. With all the available parameters, the classifier is defined as:

$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^l y_i \alpha_i (\mathbf{x}_i \cdot \mathbf{w} + b) \right) \quad (4)$$

However, this classifier can be only used on the linearly separable problems. To solve more complicated problems, kernels can be used to transform the original

**Fig. 9** The principles of support vector machines (SVM)



data from input space to higher dimensional feature space where the data points are separable. The most widely used kernels are shown in the following table (Table 1):

$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^l y_i \alpha_i K(\mathbf{x}_i^s, \mathbf{x}) + b \right) \quad (5)$$

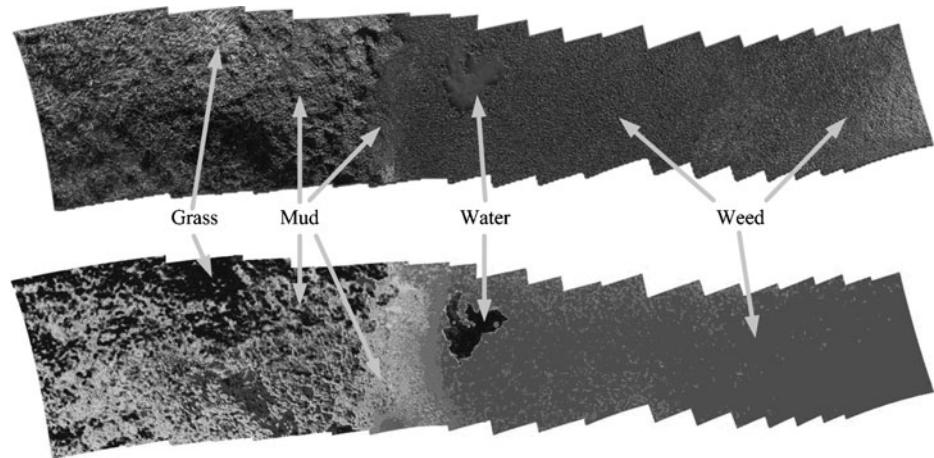
The classifier with the kernel is given in Eq. 5.

The advantages of SVM are its ability to use kernels to solve non-linear problems. Also there are no local minima, the optimisation solution is always global and unique. The disadvantage of SVM is that there is no systematic method of choosing the kernel parameter.

The detection algorithm is based on maximum margin classification. A large data set comprising of  $n \times n$  pixels of imagery taken from the camera is collected and the set separated manually into what is and what is not a weed. Each  $n \times n$  pixel is in itself marked as an image of  $p$  dimensions. In particular these dimensions take into consideration colour, shape and texture. The classification algorithm then tries to determine a hyperplane which separates the images into the two sets; this hyperplane becomes the detection model. Thus any new data that comes in is passed through this model and will fall on either side of the hyperplane depending on whether it is or is

**Table 1** Commonly used kernel and the corresponding classifier [20]

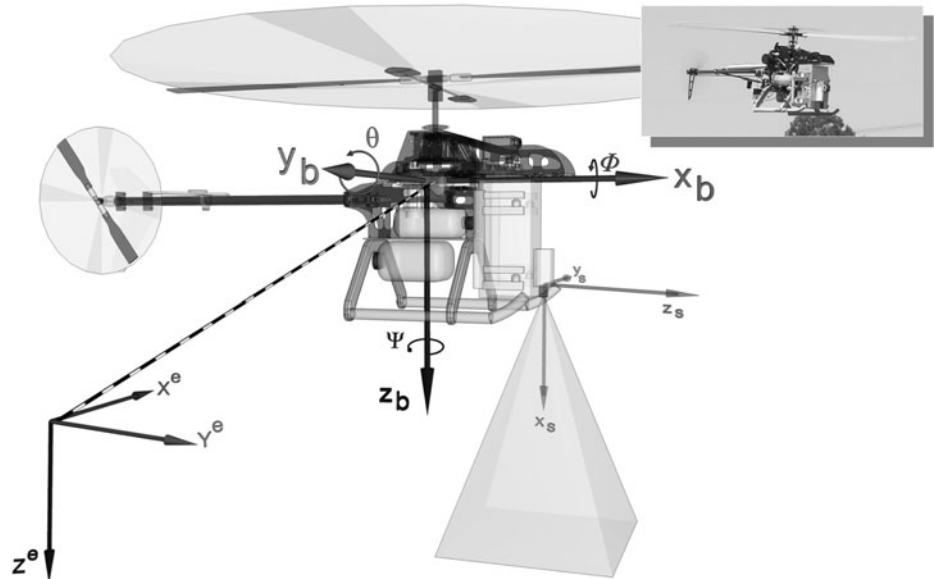
	Kernel	Classifier
$K(\mathbf{x}_i^s, \mathbf{x}) = (\mathbf{x}_i^s \cdot \mathbf{x} + 1)^p$	Polynomial kernel	Polynomial learning machine
$K(\mathbf{x}_i^s, \mathbf{x}) = \exp\left(-\frac{\ \mathbf{x}_i^s - \mathbf{x}\ ^2}{2\sigma^2}\right)$	Gaussian kernel	Radial basis function network
$K(\mathbf{x}_i^s, \mathbf{x}) = \tanh(\kappa \mathbf{x}_i^s \cdot \mathbf{x} - \delta)$	Tangent hyperbolic kernel	Two layer neural network



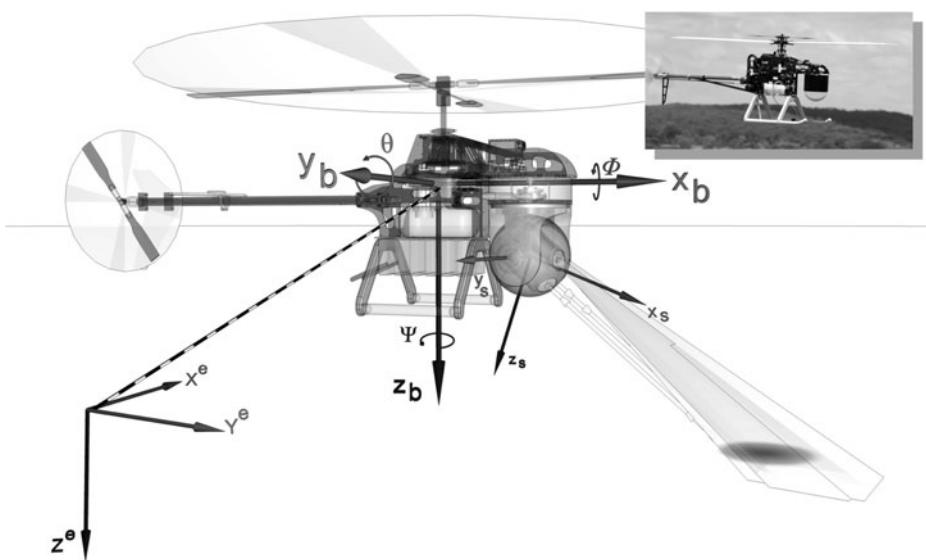
**Fig. 10** Using SVM to detect weed infested regions. The *image at the top* is a stitched mosaic of airborne images. The *image in the bottom* is the probability distribution of the weeds

not a weed. Depending on how far the image fits away from the hyperplane will determine the probability that the image is within that set.

The upper section of Fig. 10 shows the weeds, water, grass and mud covered regions in a stitched mosaic of airborne images captured by the RUAV. The lower part of the same figure indicates the probability of being weed or not weed. This



**Fig. 11** The body-fixed frame reference, and the sensor-fixed frame reference of the fixed camera system. The *inset* shows the RUAV in flight carrying down-looking camera



**Fig. 12** The body-fixed frame of reference, and the sensor-fixed frame reference of the gimbaled vision sensor. The *inset* shows the RUAU in flight, carrying a gimbal mechanism

binary grouping simplifies the spraying decision. Red colored regions indicates the high confidence in the detection of weed. The algorithm proved to be very robust and reliable. The *LIBSVM* [22] library, an integrated implementation of the SVM is utilised in the classification and detection processes.

## 8 Georeferencing

Once the FoIs and the weed infested regions are located in the image frames, the next process is to determine their global coordinates so that RUAU spray mission plans can be prepared.

Geo-referencing of airborne imagery is a well studied field. The geo-referencing and rectification of images acquired with remotely operated RC model airplane is briefly explained in [23]. Similarly in [24] shows the process of geo-referencing of airborne imagery from multiple cameras.

Figures 11 and 12 illustrate the coordinate transformations on the body-fixed camera and on the gimbaled camera systems used for the airborne *close sensing* weed imagery. The detailed formulation of geo-referencing process is given in [25].

## 9 Conclusion and Future Works

This paper presented a novel ecological research application of a UAS. A remotely controlled RUAU with optional live video and telemetry feed back to the human operators was shown to be effective to reach inaccessible, weed infested aquatic

habitats. For beyond line-of-sight operations a RUAU with autopilot and online decision making system is required.

We have investigated and applied a number of machine learning techniques. Our initial experiments with the Support Vector Machine (SVM) provided promising results. However we will continue to work in machine learning domain and improve our system further.

The presented system well performed in detection, classification and mapping of alligator weed and salvinia. Both of these weeds are considered as “Weeds of National Significance”. The tacit knowledge and expertise of the human weed experts are successfully incorporated into the decision making processes [17].

Possibilities of utilisation of the multi-spectral and hyper-spectral sensors will be investigated in future work.

Using autonomous RUAUs for the aquatic weed surveillance and management has shown to produce promising positive outcomes for the natural environment.

**Acknowledgements** This work, under the project title of “Cost Effective Surveillance of Emerging Aquatic Weeds Using Robotic Aircraft” was supported by Land & Water Australia (LWA) as a part of the “Defeating the Weed Menace” (DWM) program, and partly supported by the Australian Research Council (ARC) Centre of Excellence programme. Authors express their appreciation to Judy Lambert, DWM program coordinator, Andrew Petroeshevsky, National Aquatic Weeds Coordinator, NSW Department of Primary Industries, Grafton Agricultural Research & Advisory Station, Luke Joseph, Farm & Dam Control Pty Ltd, and SunWater for their invaluable advice. This project would not be possible without the support of ACFR’s Aerospace group members, particularly without the state-of-the-art engineering support from the team of Steve Keep, and Muhammad Esa Attia.

## References

1. Australian Government: Department of Agriculture, Fisheries and Forestry, “Contours”. ISSN:1447-9087 (2007)
2. Cook, K.L.B.: The silent force multiplier: the history and role of UAVs in warfare. In: IEEE Aerospace Conference (2007)
3. Dossier: Civilian applications: the challenges facing the UAV industry. Air Space Eur. **1**(5–6), 63–66 (1999)
4. Cox, T.H., Nagy, C.J., Skoog, M.A., Somers, I.A.: Civil UAV capability assessment (draft version). NASA Capability Assessment Report (2004)
5. Task-Force: UAV, a concept for European regulations for civil unmanned aerial vehicles (UAVs). The Joint Aviation Authorities(JAA)/AuroControl Initiative on UAVs (2004)
6. Herwitz, S.R., Johnson, L.F., Arvesen, J.C., Higgins, R.G., Leung, J.G., Dunagan, S.E.: Precision agriculture as a commercial application for solar-powered unmanned air vehicles. In: AIAA’s 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles, AIAA 2002-3404, 7 pp. (2002)
7. Rango, A., Laliberte, A., Steele, C., Herrick, J.E., Bestelmeyer, B., Schmugge, T., Roanhorse, A., Jenkins, V.: Research article: using unmanned aerial vehicles for Rangelands: current applications and future potentials. Environ. Pract. **8**, 159–168. ISSN:1466-0466 (2006)
8. Hardin, P.J., Jackson, M.J.: An unmanned aerial vehicle for Rangeland photography. J. Range-land Ecol. Manage. **58**(4), 439–442 (2005)
9. Herwitz, S.R., Johnson, L.F., Dunagan, S.E., Higgins, R.G., Sullivan, D.V., Zheng, J., Lobitz, B.M., Leung, J.G., Gallmeyer, B.A., Aoyagi, M., Slye, R.E., Brass, J.A.: Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. J. Comput. Electron. Agric. **44**(1), 49–61 (2004)
10. Grigg, G.C., Pople, A.R., Beard, L.A.: Application of an ultralight aircraft to aerial surveys of kangaroos on grazing properties. J. Wildl. Res. **24**, 359–372 (1997)

11. Sainty, G.R., Jacobs, S.W.L.: Waterplants in Australia, 4th edn, 416 pp. Sainty and Associates Pty. Ltd., Sydney. ISBN:0-9581055-1-0 (2003)
12. Holm, L.G., East-West Center: The world's worst weeds: distribution and biology, vol. xii, 609 pp. Honolulu: Published for the East-West Center by the University Press of Hawaii. ISBN:0824802950 (1977)
13. Novatel OEM4-GSL-RT20: <http://www.novatel.com/Documents/Papers/oem4g2.pdf>. Retrieved 26 Feb 2008
14. Honeywell: HMR2300 smart digital magnetometer. <http://www.ssec.honeywell.com/magnetic/datasheets/hmr2300.pdf>. Retrieved 26 Feb 2008
15. MDL Laser ACEIM35: The laser, IM OEM laser modules. [http://www.mdl.co.uk/laser\\_modules/laserace-im/index.html](http://www.mdl.co.uk/laser_modules/laserace-im/index.html). Retrieved 26 Feb 2008
16. Hitachi Kokusai Electric Inc.: HV-F31 progressive scan 3CCD camera. [http://www.hitachidenshi.com/supportingdocs/.../microscope\\_medical/HVF31.pdf](http://www.hitachidenshi.com/supportingdocs/.../microscope_medical/HVF31.pdf). Retrieved 26 Feb 2008
17. Sukkarieh, S., Bryson, M., Göktogan, A.H., Attia, M.E., Keep, S., Lunton, T., Randle, J.: "Cost effective surveillance of emerging aquatic weeds using robotic aircraft" project demonstration video. <http://www.acfr.usyd.edu.au/research/projects/aerospace/FundRes/Weed-Helicopter-2008.wmv>. Retrieved 1 March 2009
18. Trebar, M., Steele, N.: Application of distributed SVM architectures in classifying forest data cover types. *Comput. Electron. Agric.* **63**(2), 119–130 (2008)
19. Cortes, C., Vapnik, V.: Support-vector networks. *J. Mach. Learn.* **20**(3), 273–297 (1995)
20. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. *J. Data Min. Knowl. Discov.* **2**(2), 121–167 (1998)
21. Vapnik, V.: The nature of statistical learning theory, 2nd edn., 314 pp. Springer-Verlag New York, Inc. ISBN:10:0387987800 (1999)
22. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. Retrieved 26 Feb 2008
23. Ladd, G., Nagchaudhuri, A., Earl, T.J., Mitra, M., Bland, G.: Rectification, georeferencing, and mosaicing of images acquired with remotely operated aerial platforms. In: Proceedings of the American Society for Photogrammetry and Remote Sensing Annual Conference ASPRS 2006, 10 pp, 1–5 May 2006
24. Mostafa, M.M.R., Schwarz, K.P.: Digital image georeferencing from a multiple camera system by GPS/INS. *ISPRS J. Photogramm. Remote Sens.* **56**, 1–12 (2001)
25. Göktogan, A.H.: Gimbaled multimodal sensor design for unmanned aerial vehicles. In: Proc. II Aeronautics and Space Conf. (UHUK'08), pp. 16., ISBN:978-975-561-341-3. İstanbul, Türkiye, 15–17 October 2008

# Development and Evaluation of a Chase View for UAV Operations in Cluttered Environments

James T. Hing · Keith W. Sevcik · Paul Y. Oh

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 19 August 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** Civilian applications for UAVs will bring these vehicles into low flying areas cluttered with obstacles such as building, trees, power lines, and more importantly civilians. The high accident rate of UAVs means that civilian use will come at a huge risk unless we design systems and protocols that can prevent UAV accidents, better train operators and augment pilot performance. This paper presents two methods for generating a chase view to the pilot for UAV operations in cluttered environments. The chase view gives the operator a virtual view from behind the UAV during flight. This is done by generating a virtual representation of the vehicle and surrounding environment while integrating it with the real-time onboard camera images. Method I presents a real-time mapping approach toward generating the surrounding environment and Method II uses a prior model of the operating environment. Experimental results are presented from tests where subjects flew in a H0 scale environment using a 6 DOF gantry system. Results showed that the chase view improved UAV operator performance over using the traditional onboard camera view.

**Keywords** UAV safety · UAV accidents · UAV training

---

J. T. Hing · K. W. Sevcik · P. Y. Oh (✉)  
Department of Mechanical Engineering and Mechanics, Drexel University,  
3141 Chestnut Street, Philadelphia, PA, USA  
e-mail: paul@coe.drexel.edu

J. T. Hing  
e-mail: jth23@drexel.edu

K. W. Sevcik  
e-mail: kws23@drexel.edu

## 1 Introduction

Safe and efficient remote operation of a UAV in a cluttered environment requires that the pilot have a good sense of the state of the vehicle and the surrounding environment that the vehicle is operating in. Even in autonomous systems, it is still very important for the operator to know the state of the vehicle and its surroundings, especially when dealing with system faults. This awareness of the state of the vehicle and its surroundings is called situational awareness(SA). The accepted definition of SA comes from [1] where it is broken down into levels. Level 1 SA is the perception of the elements in the environment within a volume of time and space. Level 2 SA is the comprehension of their meaning and Level 3 SA is the projection of their status in the near future. Situational awareness is effected by many factors. Current remote and autonomous systems are limited in what information is relayed from the vehicle back to the UAV pilot/operator. The operator's physical separation from the vehicle eliminates all motion feedback where as manned aircraft pilots utilize this motion to help in vehicle control. A UAV pilot's field of view is restricted due to limitations of the lens and positioning of the onboard camera. The limited field of view makes it difficult for the pilot to know the location of the extremities of the vehicle, which the authors believe to be critical knowledge when operating in a cluttered environment. The onboard camera view also requires a constant mental re-mapping of the environment by the pilot due to changing camera angles, which can also lead to vertigo. A pilot's understanding of the state of the vehicle, surrounding environment and mission status is solely reliant on the overwhelming visual representation of this information, many times leading to mental exhaustion. These limitations combined with a high workload lead to a lower situational awareness thereby increasing the chance for a mishap or accident.

Civilian applications for unmanned aerial vehicles (UAVs) will introduce these vehicles into cluttered near earth environments [2]. These are low flying areas typically cluttered with obstacles such as buildings, trees and power lines. More importantly, these areas are also populated with civilians. With the current accident rate of UAVs being significantly higher than that of commercial airliners [3], the idea of UAVs being operated in this type of environment is alarming. However, the potential of these vehicles to greatly benefit civilians demands that we evaluate what is necessary to improve the safety and operations of these vehicles in these environments.

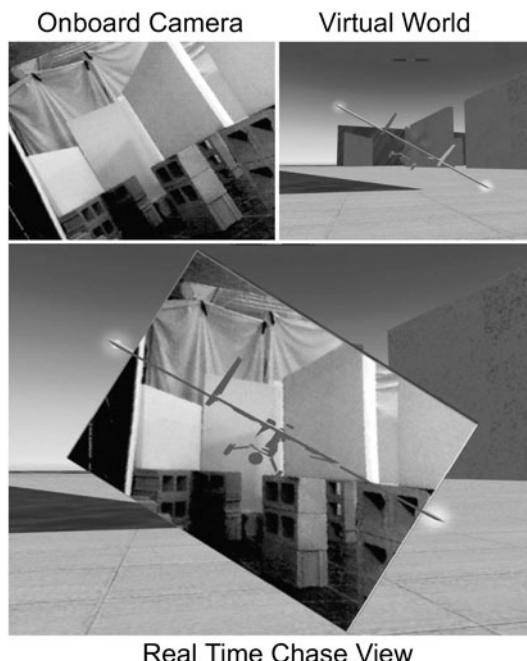
The current state of the art UAVs are designed and operated to successfully complete tasks that commonly take place in higher altitude areas with very few obstacles to navigate around [4]. Most UAVs during a majority of these mission are operated under some sort of autopilot control. However, operation in cluttered environments requires fast and accurate obstacle avoidance algorithms, fast object recognition, and quick adaptation to changing conditions. Limitations of current UAV systems, specifically the current limits of artificial intelligence, can be overcome in these environments by keeping the UAV under the full control of a human pilot. However, control of the vehicle by the pilot is not enough, especially for operations in cluttered environments, as supported by the limited SA of current UAV pilots listed earlier.

In this paper, we investigate an approach to improving SA that utilizes sensor packages common on most UAV systems. The approach uses an onboard camera and

an inertial measurement unit to generate a mixed-reality chase view to the operator as seen in Fig. 1. There are two methods that we are developing to generate the mixed-reality chase viewpoint. The mixed-reality notion comes from the fact that the surrounding environment displayed to the pilot (outside of the onboard camera field of view) is a virtual representation. In method I, the surrounding environment is created by real-time mapping of features extracted from the onboard camera view. In method II, the surround environment is created using a prior model of the environment being operated in. A prior model could be constructed using geospatial digital terrain elevation data (DTED), satellite imagery, or prior manned or unmanned forward observer reconnaissance missions. For the chase view, the onboard camera images are still relayed to the pilot but are rotated, keeping the horizon level, and keeping the perspective consistent with the displayed chase viewpoint. This view allows the pilot to see the entire aerial vehicle pose and surrounding environment as if they were following a fixed distance behind the vehicle. The benefits of this viewpoint are an increased awareness of the extremities of the vehicle, better understanding of its global position in the environment, mapping of the environment, and a stable horizon (which helps to reduce the chance for vertigo as well).

The main contribution of this paper are the results from studying the differences when piloting a UAV in an cluttered environment while using a chase viewpoint versus using an onboard camera viewpoint. The results are obtained from experiments conducted at H0 (1:87) scale using a 6 degree of freedom (DOF) gantry that emulates an aircraft's translations and rotations. Also presented is the continuing work developing this approach for real world testing. The rest of this paper is

**Fig. 1** The chase viewpoint during UAV operation in a cluttered environment consists of the real-time onboard camera image integrated into a virtual representation of the surrounding environment and aircraft pose



organized as follows: Section 2 gives some background on the previous work conducted in the area of improving situation awareness for UAV pilots. Section 3 presents the methods behind generating a chase view for UAV operation. Section 4 presents the experimental setup for evaluating UAV pilot skills in cluttered environments with different viewpoints. Section 5 presents the results from the study and Section 6 concludes the paper with a discussion and future work.

## 2 Previous Work

SA for operators of robotic ground and aerial vehicles has been investigated by a few researchers such as in [5]. In [5] it was reported that robots being operated at a post World Trade Center site were being operated with some sort of operator error 18.9% of the time due to poor interfaces and lack of functional presence. Research such as this and others have lead to proposals on ways to improve UAV pilot situational awareness. In prior work, the authors of this paper investigated the use of motion platform technology to relay motion cues to a UAV pilot [6]. While the method showed potential, the current motion platform technology is very large and expensive for a majority of the potential civilian UAV operators. Other researchers have investigated new designs for heads up displays [7], adding tactile and haptic feedback to the control stick [8, 9] and larger video displays [10]. Synthetic vision, in recent years, has been studied and shown to improve situational awareness for remotely piloted vehicles [11]. Synthetic vision displays to the operator a far distance exocentric view of the UAV with a virtual representation of the terrain based on a database of elevation maps. The method requires prior knowledge of the terrain/elevation and does not include obstacles other than the natural terrain data. Synthetic vision is mostly used to depict the planned trajectory from a 3D perspective for support in guidance and control. Also related to what we are proposing is the work conducted by [12, 13] where real world current data is used for generating virtual views.

Sugimoto et al. [12] and Nielsen et. al. [13] developed methods for viewing remotely operated ground vehicles from outside the vehicle; “Time Follower’s Vision” for [12] and tethered position in [13]. Both methods produced a viewpoint that allowed an entire visualization of the vehicle pose and the environment directly surrounding the vehicle itself. Both works presented studies showing that their methods improved remote operation of the vehicle in both speed of operation and accuracy of vehicle positioning. In [12] however, they did not use any sort of mapping of the environment, leaving this up to the memory of the operator. In [13] they generated a 2-D map of the environment as the vehicle drove around, using a laser range finder and SLAM. This map was relayed to the operator in the tethered view showing walls as slightly raised obstacles.

A major challenge faced when adapting concepts created for ground vehicles, as in [12, 13], to air vehicles is that air vehicles are capable of much more movement than ground vehicles. In cluttered and urban environments, UAVs can undergo large three dimensional translations and rotations. Another challenge is that, obstacles can not be represented by infinitely high walls (often used in 2D ground vehicle maps) as UAVs can fly around, above, and in the case of overpasses, below obstacles. UAVs, especially those flown in urban environments, will be small so they can maneuver

between obstacles with relative ease. The small size limits the payload capacity of the vehicle. Laser range sensors, like those used in [13], can be too heavy to add to a typical UAV sensor suite that already includes an IMU, GPS and an onboard camera.

Closely related to the work presented by the authors of this paper is the work conducted by [14, 15]. In [14], during high altitude search tests, they used real-time video data from the UAV and augmented it with pre-loaded map data (satellite imagery). The onboard camera view is rotated to match the preloaded terrain map and a silhouette of the UAV is displayed on the map showing its heading. The preloaded terrain map does not change its orientation. This would have some negative effects in piloting the vehicle in terms of remapping of the controls, for example when the UAV is flying south on the map, pulling left on the joystick makes the UAV go to the right. However, the purpose of the research was not to improve operator control (the UAV flew at a high altitude autonomously through waypoints), but to improve the situational awareness of the observer conducting the search task. Their results showed that the augmented image helped the observers comprehension of the 3D spatial relationship between the UAV and points on the earth. In [15] they investigated the effects of displaying a simplified “wing-view” of the UAV to the operator that shows roll and altitude of the aircraft. The motivation was that UAV operators are standing on the ground and a display with a moving horizon may confuse users not trained as pilots. This display helped with the operators in the understanding of the instantaneous relationship between the UAV and the world.

The authors of this paper believe that the concepts investigated in [12–15] could be adapted and developed such that they can be applied to UAV operation in cluttered environments. Utilizing the IMU and onboard camera, the authors of this paper show two methods for generating a chase view point for UAV pilots.

### 3 Methods Towards Generating Chase Viewpoint

UAVs operating in urban and cluttered environments will most likely be limited to smaller back-packable and hand launchable vehicles that enable quick maneuvering and access to small spaces. With limited payload, choosing an optimal sensor suite is extremely important. The ultimate goal is to gather all data about the state of the vehicle and information from the surrounding environment using as few sensors as possible.

There are two methods presented in this paper for generating a chase viewpoint. Method I utilizes an onboard camera and GPS to generate a 3D map of the environment. Method II utilizes the onboard GPS of the aircraft and prior knowledge of the operating environment to generate a surrounding 3D map. The advantage of Method I is that a map is created based on a very recent interaction with the environment and can be used without prior knowledge of the operating area. It can also be adapted to work in areas without GPS availability by finding vehicle state information from structure from motion methods. Method I however comes at a cost of computation power, which limits the speed at which the UAV is allowed to fly in the environment. Method II allows for much faster flight as the environment is already mapped. Should the environment change, the pilot will be forced to mentally remap the surrounding environment during the flight using the onboard camera view.

### 3.1 Method I

A chase viewpoint requires three dimensional measurements of the surrounding environment and accurate knowledge of the state of the vehicle. Researchers are currently working on methods to gather this information from only one onboard camera [16, 17] using Structure from Motion (SFM) methods. The added benefits of this is that UAVs can be smaller, and the vehicle is capable of map building in areas with no GPS signal. As these methods are currently computationally expensive, the authors of this paper chose to use information from an onboard IMU, GPS, and camera for the initial work toward developing the chase viewpoint. The technique for Method I is presented in the following sub sections.

#### 3.1.1 Feature Detection and Tracking

Creating a map of the surrounding environment from the onboard camera view requires that three-dimensional information be extracted from multiple two-dimensional camera images. Features in each image must be found and tracked from frame to frame. Based on recommendations from [18], the authors use a  $7 \times 7$  feature detection window and calculate the spatial gradient matrix,  $H$ , as the window scrolls through the camera image.

$$H = \Sigma \begin{vmatrix} (\delta I / \delta x)^2 & (\delta I / \delta x)(\delta I / \delta y) \\ (\delta I / \delta x)(\delta I / \delta y) & (\delta I / \delta y)^2 \end{vmatrix} \quad (1)$$

where  $I(x,y)$  is the gray level intensity and the summation is through the feature window. If the eigenvalues of  $H$  are greater than a chosen threshold then that particular area of the image is chosen as a feature point to track. Features are chosen such that they are the strongest features in the image, don't overlap, and only a set number of features desired by the user are kept.

Tracking of the feature points is conducted using a pyramidal implementation of the Lucas Kanade feature tracker (KLT)[19]. The pyramidal implementation allows for much larger movement between two images. Currently the authors are using a 3 level pyramid which can track pixel movement 8 times larger than the standard Lucas Kanade tracker. In a traditional pyramidal KLT, feature points are chosen in the highest level of the pyramid. This however did not produce desired results. As such, the authors proceeded with the following: First, features are detected on the highest resolution image which is currently at  $640 \times 480$  (onboard camera resolution). A  $5 \times 5$  gaussian blur is used before each re-sampling of the image all the way to the third level which is  $80 \times 60$  resolution. The centroids of the chosen features are mapped to the location on the third level. For frame J to K, the previous and current onboard camera image respectively, the following calculations take place over 10 iterations with iteration i starting at 1:

First an image difference  $\delta I(x, y)$  is calculated:

$$\delta I_i(x, y) = J^L(x, y) - K^L \left( x + g_x^L + v_x^{i-1}, y + g_y^L + v_y^{i-1} \right) \quad (2)$$

where for level 3 ( $L = 3$ ), the initial guess  $g_x, g_y$  is zero and the iteration guess  $v^0 = (0, 0)$ . Then the image mismatch vector  $b_i$  is calculated for the feature window:

$$b_i = \Sigma \begin{vmatrix} \delta I_i(x, y) I_x(x, y) \\ \delta I_i(x, y) I_y(x, y) \end{vmatrix} \quad (3)$$

The optic flow  $\eta^i$  is then calculated:

$$\eta^i = H^{-1}b_i \quad (4)$$

And the guess for the next iteration becomes:

$$v^{i+1} = v^{i-1} + \eta^i \quad (5)$$

After the iterations are complete the final optic flow  $d^L$  for the level is:

$$d^L = v^{10} \quad (6)$$

The guess for the next lower pyramidal level  $g_x, g_y$  becomes:

$$g_x, g_y = 2(g^{L-1} + d^L) \quad (7)$$

And the process repeats until the final level ( $L^0$ ), the original image, is reached. The final optic flow vector  $d$  is:

$$d = g^0 + d^0 \quad (8)$$

And the location of the tracked feature on image K is:

$$K(x, y) = J(x, y) + d \quad (9)$$

The tracking (50 features) is at sub pixel resolution and is currently running at 10 FPS on a 2.33 GHz dual core machine.

### 3.1.2 Reconstruction and Mapping

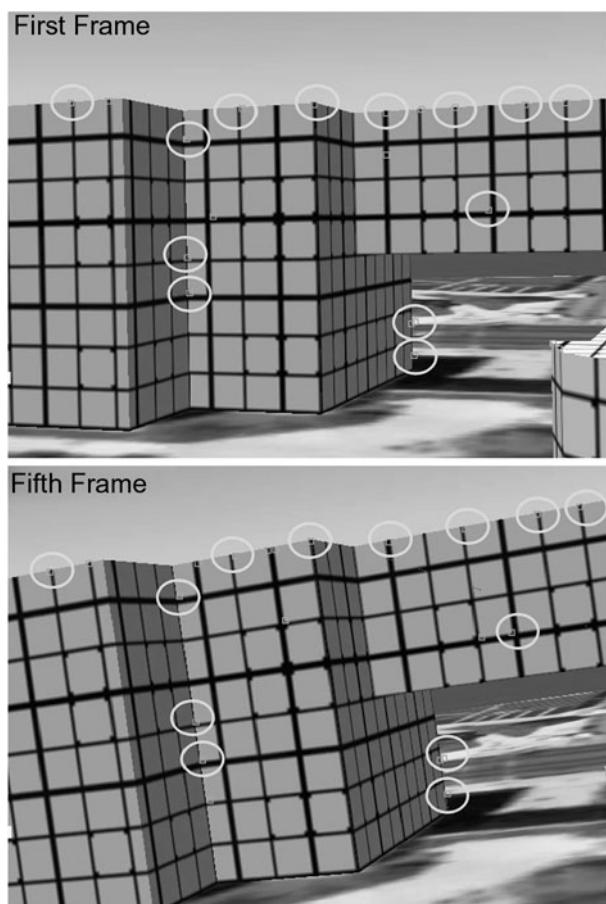
For the initial development, we are utilizing a simulated environment that we modeled in the flight simulation package X-Plane from Laminar Research as seen in Fig. 2. Since the authors chose to use an IMU and GPS along with the camera,



**Fig. 2** Environment (Drexel University Campus) created in X-Plane for testing feature tracking and reconstruction. Initial textures were of grid patterns for easier development during the initial stages

structure from motion methods are not needed and the 3-Dimensional locations of the feature points can be found through triangulation. The extrinsic parameters for the camera are extracted from GPS and IMU measurements in the X-Plane simulation. The intrinsic parameters of the camera are calculated prior to the tests using multiple images of a known grid pattern. Calibration tests found the focal length for the camera in the X-Plane environment to be 320.469 mm. Each feature point is stored in its initial frame and then tracked. If the feature point is successfully tracked for 5 frames, as seen in Fig. 3, it is used in the reconstruction algorithm. The 5 frame difference was chosen to allow a greater distance between the two camera images before reconstruction is run. The global frame of reference is chosen such that the axes lie on the latitude (Y), longitude (X) and altitude directions (Z) of the simulated environment. The origin of the axes are located in the simulated world where the vehicle is initially spawned. The distance to the aircraft camera from the global reference frame is calculated from GPS and IMU values. Locations of feature

**Fig. 3** Feature tracking across multiple frames. Features detected are surrounded by a small box. The tracked features used in reconstruction are highlighted by circles. The frames contain a rotated view (aircraft is rolling) side of a building at Drexel. The texture of the walls were created with a grid pattern to ease the use of feature detection/tracking for initial development



points in the camera image plane are transformed to the global reference frame using the following rotation and translation matrices:

$$\begin{aligned}
 R_{1,1} &= \cos(\alpha)\cos(\gamma) - \sin(\alpha)\sin(\beta)\sin(\gamma) \\
 R_{2,1} &= \sin(\alpha)\cos(\gamma) + \cos(\alpha)\sin(\beta)\sin(\gamma) \\
 R_{3,1} &= -\cos(\beta)\sin(\gamma) \\
 \\ 
 R_{1,2} &= -\sin(\alpha)\cos(\beta) \\
 R_{2,2} &= \cos(\alpha)\cos(\beta) \\
 R_{3,2} &= \sin(\beta) \\
 \\ 
 R_{1,3} &= \cos(\alpha)\sin(\gamma) + \sin(\alpha)\sin(\beta)\cos(\gamma) \\
 R_{2,3} &= \sin(\alpha)\sin(\gamma) - \cos(\alpha)\sin(\beta)\cos(\gamma) \\
 R_{3,3} &= \cos(\beta)\cos(\gamma)
 \end{aligned} \tag{10}$$

$$T = \begin{vmatrix} F_d\cos(\beta)\sin(\alpha) + \text{Lon.} - \text{Lon.of Origin} \\ F_d\cos(\beta)\cos(\alpha) + \text{Lat.} - \text{Lat.of Origin} \\ F_d\sin(\beta) + \text{Alt.} - \text{Alt.of Origin} \end{vmatrix} \tag{11}$$

where  $\alpha$  is the camera heading angle,  $\beta$  is the camera pitch angle,  $\gamma$  is the camera roll angle, and  $F_d$  is the camera focal length.

Reconstruction proceeds as follows:

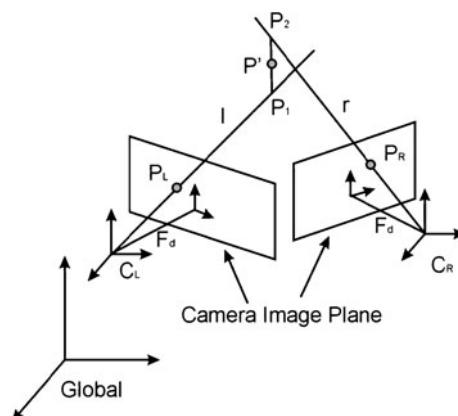
Following Fig. 4, the line running through the camera frame, C, and the feature point, P, on the image plane to the feature point in the global frame is:

$$l = C_L + a(P_L - C_L) \tag{12}$$

$$r = C_R + b(P_R - C_R) \tag{13}$$

where a and b are values between 0 and 1 representing the length of vectors l and r respectively between C and P.

**Fig. 4** Camera reconstruction geometry. Due to noise in the measurements, rays passing through the feature in the first and second camera image plane may not intersect. The midpoint of the closest point between the two rays is taken as the feature measurement



Ideally the two lines would intersect at the global location of the feature point,  $P$ , but due to noise in the measurements, they may not intersect. Therefore, it is determined that the feature point lies in the midpoint,  $P'$ , between the line segment that is perpendicular to both of the rays.

$$P_1 = C_L + a_o(P_L - C_L) \quad (14)$$

$$P_2 = C_R + b_o(P_R - C_R) \quad (15)$$

$$P' = P_1 + 1/2(P_2 - P_1) \quad (16)$$

where  $a_o$  and  $b_o$  represent the values of  $a$  and  $b$  where the line  $P'$  crosses the  $l$  and  $r$  vectors respectively.

The orthogonal vector,  $w$ , to both lines,  $l$  and  $r$ , is:

$$w = (P_L - C_L) \times (P_R - C_R) \quad (17)$$

Therefore, the line going through  $P_1$  to  $P_2$  is:

$$P_2 = P_1 + c_o w \quad (18)$$

The unknowns  $a_o$ ,  $b_o$ ,  $c_o$  are found by solving the following equation:

$$a_o(P_L - C_L) - b_o(P_R - C_R) + c_o w = C_R - C_L \quad (19)$$

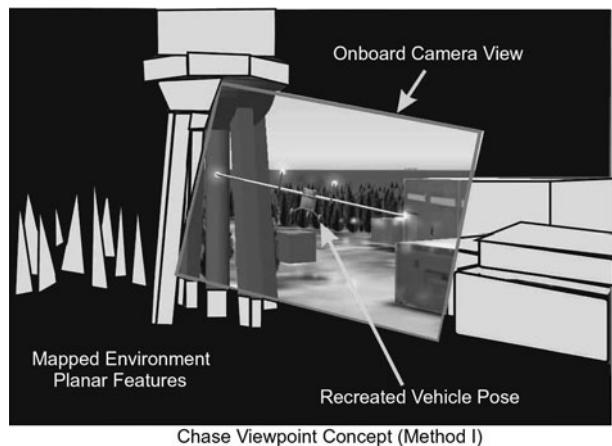
Currently the method is run without any filtering of the data so the results are somewhat noisy as seen in Fig. 5. The method up to this point runs at approximately 6 frames per second on a 2.33 GHz dual core Windows laptop. However, the programming code has not been optimized and the desired operation speed is to be no less than 10 frames per second. The following steps presented for Method I have not been completed as of yet due to current computation costs but are mentioned to build further understanding of the final goal.

Adapting a method similar to that presented by [20] we will create the three dimensional map of the environment from a single camera viewpoint. This map will then be used in the chase view perspective of the UAV pilot. What the authors of [20] do differently from a number of single camera map making algorithms is that they merge feature points into planar regions for use in SLAM. The benefits of this is that it dramatically reduces the number of stored feature points needed to create a map.

**Fig. 5** Top down view of raw (non-filtered) reconstruction of feature points with flight environment overlaid over the data. Most data points far away from building edges are points reconstructed from features detected on the ground



**Fig. 6** Conceptual graphic showing the chase viewpoint during UAV operation in a cluttered environment



Much of urban terrain contains rectangular buildings. Therefore, many detected features can be turned into planar regions that represent building walls and rooftops. Once the mapping is completed, the chase viewpoint can be generated by integrating the UAV onboard camera view with the UAV perspective of the generated map. Also, a virtual view of the UAV will be displayed in the map, generated using IMU data and its relative size in the displayed map, along with the real time onboard camera footage which will be rotated (and warped if necessary) to match the chase view perspective. This concept can be seen in Fig. 6. This method of generating the chase view allows for a current map of the environment to be relayed to the operator at the expense of high computation requirements and limited flight speed.

### 3.2 Method II

As stated earlier, Method II requires much less computation during the flight as the operating environment is modeled prior to the flight. Again, one can easily generate such models from DTED data, satellite imagery and forward-observer reconnaissance. In a few applications, the environment will stay relatively static which makes Method II valid. For this paper, X-Plane flight simulation software is used to model the UAV operation environment during flight tests. Aircraft position in the modeled environment is updated by GPS from onboard the UAV. The onboard camera view is rotated based on the roll angle received from the onboard IMU and surrounded by the simulated environment.

## 4 Experiment Setup

To validate efforts toward generating a chase viewpoint for UAV pilots, experiments were setup to test pilot skills operating in a cluttered environment using the current onboard camera viewpoint and a generated chase view point. The ideal scenario is to have a chase-view of the actual environment built from the real sensor data. Method I is the work we have done toward that goal. However, results are noisy and the update rate is slow. To evaluate the utility of a chase view, we conducted

tests using an onboard camera fused with a pre-built 3D model. For the experiment results presented later in this paper, Method II is used.

#### 4.1 Hardware

Field testing at the current stage of the project is risky and requires a long process of approvals to operate a UAV in restricted airspace. Tests using only a flight simulator would help validate the design notion but it is difficult to simulate the mechanical systems/sensors used in real world tests and environmental conditions. Such examples would be the camera visuals, actuator responses, lighting effects, fog and rain. Because of these reasons, the authors took advantage of the Systems Integrated Sensors Test Rig (SISTR) facility at Drexel University to conduct flight experiments in a scaled environment with actual UAV system hardware. SISTR was constructed with support from the U.S. National Science Foundation for the design and testing of UAVs and UAV sensor suites. SISTR is a 3 degree of freedom (DOF) gantry system with a workspace of  $18' \times 14' \times 6'$  [21]. To match the size of a reasonable real world UAV test environment, SISTR's workspace represented an H0 scale (1:87) environment as seen in Fig. 7. The flight environment consisted of narrow corridors that can be representative of corridors between large buildings in an urban environment.

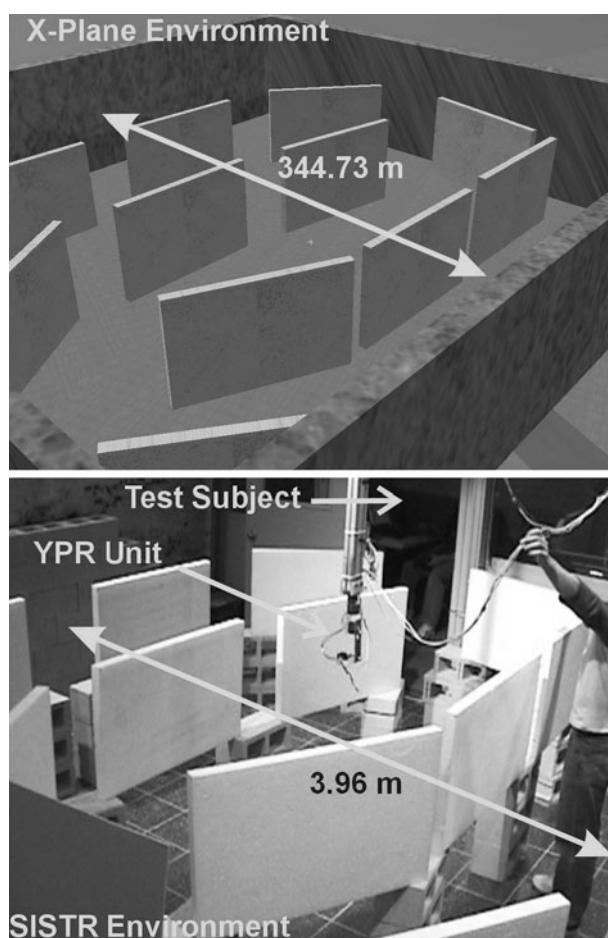
SISTR's end effector is used to represent the location of the aircraft inside of the scaled environment. Aircraft dynamics during the experiments are handled by a flight simulation package and the H0 scaled translational position of the aircraft is relayed from the flight simulator to SISTR's controller via User Datagram Protocol (UDP) at a rate of 20HZ.

The aircraft's angular positions are commanded by the experiment subject (pilot) via a joystick. The resulting angular position of the aircraft, generated by the flight simulator, is relayed to a 3 DOF yaw, pitch and roll (YPR) unit attached to SISTR's end effector as seen in Fig. 8. The YPR unit was specifically designed such that it represented the Euler angles of the aircraft; yaw is applied first, then pitch, then roll. It was also designed to have a small footprint due to operation in a scaled environment. A  $640 \times 480$  resolution camera with 35 degree field of view, seen in Fig. 8, was attached to the YPR unit. The images from the camera represented the onboard camera view from the aircraft and were relayed to the experiment subject (pilot) at a rate of 30 frames per second during onboard camera tests and 10 FPS during chase view tests.

#### 4.2 Software

Aircraft dynamics and the virtual environment are generated using a commercial flight simulator software known as X-Plane (also mentioned earlier in Method I). X-Plane incorporates very accurate aerodynamic models into the program based on blade element theory and allows for real time data to be sent into and out of the program [22]. X-Plane has been used in the UAV research community, mostly as a visualization and validation tool for autonomous flight controllers [23]. In [23] they give a very detailed explanation of the inner workings of X-Plane and detail the data exchange through UDP. During the experiment, flight commands are input into X-Plane by the subject via a joystick and X-Plane generates and sends the translational

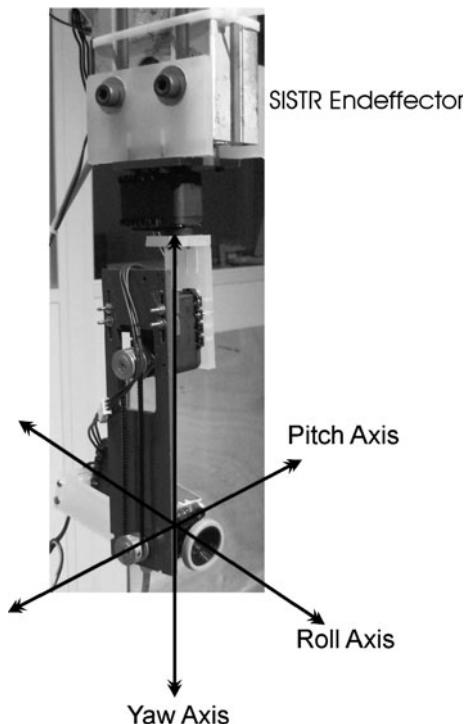
**Fig. 7** Comparison showing the real world scale flight environment with the H0 scale (1:87) SISTR environment. The *boards* create narrow corridors representative of flight between large buildings in an urban environment



and angular positions through UDP to the SISTR controller. X-Plane is also used during the chase view experiments to render the surrounding virtual view of the aircraft environment based on a prior environment model. The H0 scale environment in SISTR was built to match the full scale corridor environment we created in X-Plane. The optics of the onboard camera are accounted for by adjusting the aspect ratio in X-Plane so that the virtual environment matches up with the onboard camera view.

A UAV model was created that represents a real world UAV, known as the Mako, currently in military operation. The Mako, as seen in Fig. 9, is a military drone developed by Navmar Applied Sciences Corporation. It is 130 lbs and has a wingspan of 12.8 ft. It is very similar in size and function to the more popularly known Raven and Pioneer UAVs. This UAV platform was ideal as NAVMAR is about 10 miles of Drexel, thus it could be validated, with quick feedback on the model and our notional concept, by veteran Mako pilots. For safety reasons, the simulated version of the Mako was modified so it was lighter weight with less horsepower effectively

**Fig. 8** Yaw, pitch and roll unit used to recreate the angular position of the aircraft inside SISTR. The unit is designed based on the Euler angles of the aircraft. Yaw is applied first, then pitch, then roll



decreasing it's cruise speed to 30 miles per hour in the simulation which corresponds to 6 in./s in SISTR motion at H0 scale.

#### 4.3 User Interface

The user interface was created using Visual C#. The program handled the visual presentation to the user and also the communication between X-Plane and SISTR. The program collected translational and angular position data from X-Plane, converted it to H0 scale and then transmitted it through UDP to SISTR at 20 Hz. During onboard camera tests, only the onboard camera view was shown to the pilots during flights through the environment as seen in Fig. 1(onboard camera). During the chase view tests, the program displayed to the pilot 3 items:

1. Rotated onboard camera view so the horizon stays level

**Fig. 9** The MAKO UAV developed by NAVMAR was modeled in X-Plane and used for experiment flights



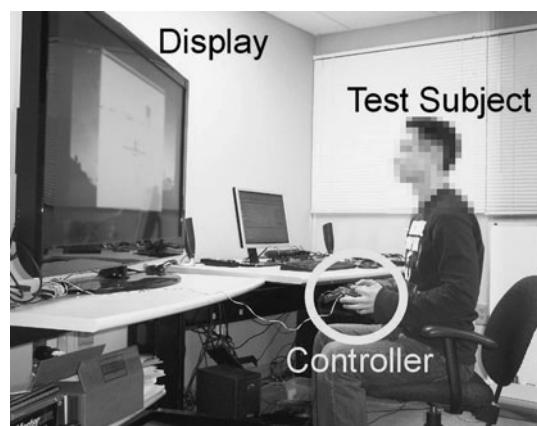
2. Virtual view of the surrounding environment based on aircraft location and prior model of the environment
3. Virtual representation of the aircraft pose to scale with the onboard camera view and surrounding environment

These items, seen in Fig. 1 are relayed in real time to the pilot. The onboard camera image is rotated by applying a simple rotation about the center of the image based on the roll angle of the aircraft.

#### 4.4 Procedure

Prior to the tests, subjects were given time to fly the Mako in an open environment in X-Plane under both simulated onboard camera view and chase view. This allowed them to become familiar with the controls and to get a feel for the response and size of the aircraft. When the subjects felt comfortable with the controls, the experiments began. As seen in Fig. 10, the subjects were placed in a room, separated from the experiment environment, with a 52" monitor from which to view the user interface. Subjects underwent multiple tests where they flew the aircraft from an onboard camera view or a chase view. During onboard camera tests, the subjects were shown only the raw view from the camera and asked to fly through the corridors of the environment while keeping a safe distance from the walls and keeping the aircraft as stable as possible. During the chase view tests, the subjects were shown the chase view and asked to fly through the corridors with the same emphasis on safe distance and stability. During each test, aircraft translational and rotational positions were recorded. If the subject crashed into the corridor walls, they were asked to continue their flight through the corridors so data collection could continue. The walls of the SISTR environment were designed to easily collapse under contact. The walls in the X-Plane environment were designed to allow the plane to pass through. After each test, subjects were given a survey on their thoughts of the different modes during the flight tests.

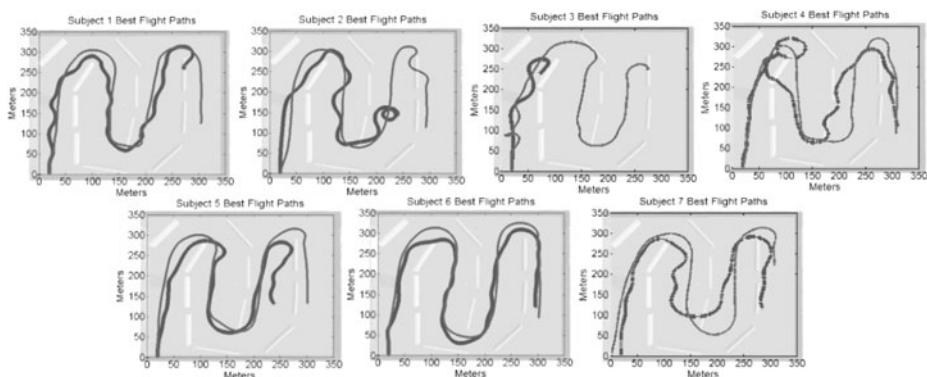
**Fig. 10** View of the operator station for the experiments. The operator sits inside of a room separated from SISTR. The 52" television displays the different views for each test. Control of the aircraft (SISTR endeffector) is provided by inputs from the operator using the controller



## 5 Results and Discussion

Seven subjects were used for initial validation of the chase view concept. Each subject varied in flight simulator experience from no experience to 5 years worth of recreational use. Shown in Fig. 11 are the best flight paths, out of all the tests, achieved for each subject using a chase view and using an onboard camera view. While using the onboard camera view, subjects showed much more of an oscillatory movement than while using the chase view. This can be attributed to two issues. During the onboard camera view tests, due to the smaller field of view, subjects would continue to turn to bring the walls into view so to establish their position in the environment. The second issue was a slight lag in the response of the camera servos and gantry to the subject's commanded desired positions similar to lag commonly experienced in current UAV field operations. This caused some subjects to overcompensate in the controls which led to increased oscillations in the flight.

Since the goal was to keep the aircraft as stable as possible, angular accelerations were recorded to quantify how well the subjects were able to do this. Table 1 shows the average angular acceleration (low angular acceleration is representative of stable flight) of each subject during trials using the chase view and trials using the onboard camera view. Due to the lack of sample size, a statistical significance was not calculated but the trend in the data leads to the conclusion that the chase view decreases the angular accelerations commanded during the flight. This is significant as quick turns under normal UAV operations can induce high stresses on the vehicle leading to accelerated wear and tear, which in turn can lead to catastrophic failure. Interesting to note, some subjects such as subject 2 and 3, did not show a dramatic decrease in the angular accelerations when switching from onboard camera view to a chase view. This shows that the view did not help the subject decrease the amount of movement they were commanding to the vehicle. However, if you look at the flight paths, the chase view did accomplish a better path through the environment. In the case of subject 3, chase view was the only way that they were able to get through the majority of the course. Subject 2 and 3 both had little to no prior flight simulator



**Fig. 11** The best flight path results for each of the 7 subjects using chase view and using the onboard camera view. The flight environment is overlayed on top of the graph. The *thin line* is the flight path using chase view, the *thick line* is the flight path when using the onboard camera view

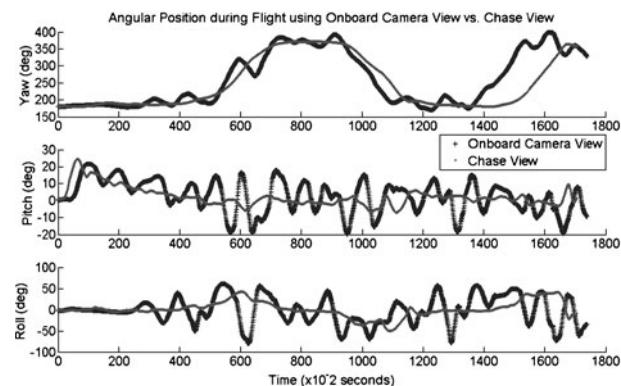
**Table 1** Average angular acceleration during chase view and onboard camera view tests

Subject	Yaw axis (deg/s <sup>2</sup> )	Roll axis (deg/s <sup>2</sup> )	Pitch axis (deg/s <sup>2</sup> )	Magnitude (deg/s <sup>2</sup> )
1 Onboard	30.00	84.56	29.65	94.52
1 Chase	22.88	36.72	20.38	47.97
2 Onboard	17.57	34.16	23.63	45.10
2 Chase	15.12	37.70	12.80	42.75
3 Onboard	29.08	56.03	36.67	73.17
3 Chase	32.06	45.87	28.95	63.36
4 Onboard	46.22	69.23	32.47	89.83
4 Chase	31.44	50.06	14.18	60.82
5 Onboard	22.95	48.95	18.86	57.29
5 Chase	7.53	23.02	8.41	25.64
6 Onboard	14.86	41.87	13.91	46.57
6 Chase	5.05	13.57	5.66	15.57
7 Onboard	28.26	36.39	11.99	47.88
7 Chase	4.69	13.61	6.47	15.78

experience before the tests (other than the warm up period). It is therefore believed that the lack of change in angular accelerations between views can be attributed to their limited understanding of how the controls effect the aircraft's flight.

The oscillatory motion is much easier to observe in Fig. 12 which shows the angular positions of the aircraft during an example of a subject's flight using the onboard camera view and chase view. During the onboard camera view tests, the subjects tended to move through a larger angular range and at a higher frequency than during the chase view tests. Most subjects after the tests stated that the chase view was much easier to operate with. For some subjects, the onboard camera view was so disorienting that they were unable to complete the course (even with crashing through barriers) after multiple tries. This was more common among subjects who had very little to no prior flight simulator experience. All of these subjects however were able to complete the course using the chase view within 2 trials. The example results presented in Fig. 12 were from a subject with a great deal of prior flight simulator experience. There was still a significant improvement in his operation when

**Fig. 12** Example data of the aircraft angular positions during an onboard camera and chase view test. The *thicker line* represents angles achieved using the onboard camera view and the *thinner line* represents the angles achieved using the chase view



using the chase view over the onboard camera view. The increased field of view and virtual aircraft pose of the chase view decreased the need to oscillate back and forth to establish the aircraft's position in the environment. The surrounding view also helped with the subject's response to the camera motion lag and vibrations from SISTR as the virtual vehicle pose made it easier for the subjects to predict the desired angular position.

## 6 Conclusion and Future Works

### 6.1 Conclusions

Future applications for UAVs will take them into low flying areas populated with obstacles and civilians. Increased situational awareness for the pilots and operators controlling those UAVs will most certainly help decrease the potential for crashes and thereby decrease the chances of property damage or harm to civilians. This paper presented the development and evaluation of implementing a chase viewpoint for UAV operations. Results from the experiments show that the chase view method has potential to increase the situational awareness of UAV pilots. The results also showed that the chase view resulted in smoother motions and flight paths for the UAV.

### 6.2 Future Works

The chase view method can certainly use more validation. Future work involves testing current Predator Pilots and other UAV operators. These are subjects who are experts in UAV operation using onboard camera views. It will also be interesting to study if the chase view method will improve the speed of UAV pilot training. Real world field tests are also desired for complete validation of the chase view. Method I presented in this paper is being developed in parallel with Method II. Future work includes continued development of Method I and human tests for validation. Method I is desired for operations in environments where the terrain and obstacles are not known *a priori*. Method I as stated in this paper can also be adapted to handle areas with no GPS by obtaining vehicle state information from structure from motion algorithms. There may also be benefit to combining both Method I and II into a hybrid of both where Method I can be used to enhance areas where Method II fails such as when the flight environment changes from the model during flight. This will also be evaluated.

## References

1. Endlesy, M.: Design and evaluation for situation awareness enhancements. In: Proceedings of the Human Factors Society 32nd Annual Meeting, pp. 97–101 (1988)
2. Oh, P.Y., Valavanis, K., Woods, R.: Uav workshop on civilian applications and commercial opportunities. (2008)
3. Weibel, R.E., Hansman, R.J.: Safety considerations for operation of unmanned aerial vehicles in the national airspace system. Tech. Rep. ICAT-2005-1, MIT International Center for Air Transportation (2005)
4. Defense, D.o.: Unmanned aircraft systems roadmap 2005-2030. Tech. rep. (2005)

5. Murphy, R.: Human-robot interaction in rescue robotics. *IEEE Trans. Syst. Man Cybern.* **34**(2), 138–153 (2004)
6. Hing, J.T., Oh, P.Y.: Development of an unmanned aerial vehicle piloting system with integrated motion cueing for training and pilot evaluation. *J. Intell. Robot. Syst.* **54**, 3–19 (2009)
7. Williams, K.W.: A summary of unmanned aircraft accident/incident data: Human factors implications. Tech. Rep. DOT/FAA/AM-04/24, US Department of Transportation Federal Aviation Administration, Office of Aerospace Medicine (2004)
8. Calhoun, G., Draper, M.H., Ruff, H.A., Fontejon, J.V.: Utility of a tactile display for cueing faults. In: Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting, pp. 2144–2148 (2002)
9. Ruff, H.A., Draper, M.H., Poole, M., Repperger, D.: Haptic feedback as a supplemental method of altering uav operators to the onset of turbulence. In: Proceedings of the IEA 2000/ HFES 2000 Congress, pp. 3.14–3.44 (2000)
10. Little, K.: Raytheon announces revolutionary new ‘cockpit’ for unmanned aircraft—an industry first (2006)
11. Tadema, J., Koenders, J., Theunissen, E.: Synthetic vision to augment sensor-based vision for remotely piloted vehicles. In: Enhanced and Synthetic Vision, vol. 6226, pp. 62260D–1–10. SPIE-Int. Soc. Opt. Eng. (2006)
12. Sugimoto, M., Kagotani, G., Nii, H., Shiroma, N., Matsuno, F., Inami, M.: Time follower’s vision: a teleoperation interface with past images. *IEEE Comput. Graph. Appl.* **25**(1), 54–63 (2005)
13. Nielsen, C.W., Goodrich, M.A., Ricks, R.W.: Ecological interfaces for improving mobile robot teleoperation. *IEEE Trans. Robot.* **23**(5), 927–941 (2007)
14. Drury, J.L., Richer, J., Rackliffe, N., Goodrich, M.A.: Comparing situation awareness for two unmanned aerial vehicle human interface approaches. Tech. rep., Defense Technical Information Center OAI-PMH Repository. <http://stinet.dtic.mil/oai/oai> (United States) (2006)
15. Quigley, M., Goodrich, M. A., Beard, R.: Semi-autonomous human-uav interfaces for fixed-wing mini-uavs. 28 September–2 October 2004
16. Webb, T.P., Prazenica, R.J., Kurdila, A.J., Lind, R.: Vision-based state estimation for autonomous micro air vehicles. *J. Guid. Control Dyn.* **30**(3), 816–826 (2007)
17. Prazenica, R.J., Watkins, A.S., Kurdila, A.J., Ke, Q.F., Kandae, T.: Vision-based kalman filtering for aircraft state estimation and structure from motion. In: AIAA Guidance, Navigation, and Control Conference, vol. v 3, pp. 1748–1760. American Institute of Aeronautics and Astronautics, Reston (2005)
18. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600. IEEE, Piscataway (1994)
19. Bouguet, J.Y.: Pyramidal implementation of the lucas kanade feature tracker: description of the algorithm. Tech. rep., Intel Corporation Microprocessor Research Labs (2002)
20. Watkins, A.S., Kehoe, J.J., Lind, R.: Slam for flight through urban environments using dimensionality reduction. In: AIAA Guidance, Navigation, and Control Conference, vol. v 8, pp. 5018–5029. American Institute of Aeronautics and Astronautics, Reston (2006)
21. Narli, V., Oh, P.Y.: Hardware-in-the-loop test rig to capture aerial robot and sensor suite performance metrics, p. 2006. In: IEEE International Conference on Intelligent Robots and Systems (2006)
22. Meyer, A.: X-plane by laminar resarch. [www.x-plane.com](http://www.x-plane.com) (2009)
23. Ernst, D., Valavanis, K., Garcia, R., Craighead, J.: Unmanned vehicle controller design, evaluation and implementation: from matlab to printed circuit board. *J. Intell. Robot. Syst.* **49**, 85–108 (2007)

# Design Methodology of a Hybrid Propulsion Driven Electric Powered Miniature Tailsitter Unmanned Aerial Vehicle

Mirac Aksugur · Gokhan Inalhan

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 17 September 2009  
© Springer Science + Business Media B.V. 2009

**Abstract** Contrary to the manned tailsitter aircraft concepts, which have been shelved and forgotten after mid 1960's, the unmanned versions of these concepts have become popular. Since, tailsitter type UAVs combine both vertical takeoff and landing (VTOL) operation and relatively high speed cruise flight capabilities which provide manifest advantages over the other VTOL aircraft concepts, including helicopters and organic air vehicles (OAVs). However, there is no mini class tailsitter UAV with efficient high speed cruise flight capability. This paper presents the design methodology and optimization of ITU Tailsitter UAV concept with hybrid propulsion system approach to fill that gap. The initial design and analysis show the advantageous performance over other mini-class VTOL UAVs.

**Keywords** VTOL · Tailsitter · UAV · Electric propulsion

## 1 Introduction

Tailsitter UAVs combine vertical takeoff and landing (VTOL) operation and relatively high speed cruise capabilities in a single airframe and such a concept provides manifest advantages over the other VTOL aircraft concepts including helicopters and organic air vehicles (OAVs). As a result of the increasing demand for efficient and silent UAV concepts which require no regular “runway” for urban-civilian applications, the design of ITU-BYU tailsitter concept is tailored towards city and urban operations with possible autonomous recharging capability to allow 24 hour

---

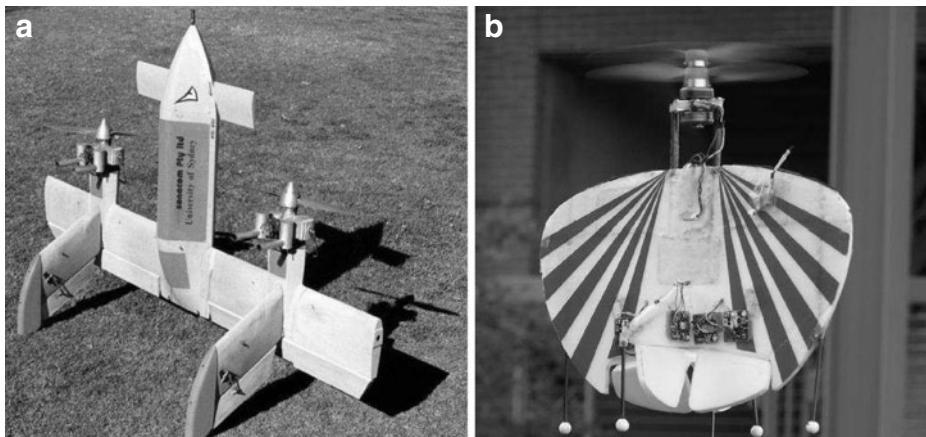
M. Aksugur (✉) · G. Inalhan  
Controls and Avionics Lab, Faculty of Aeronautics and Astronautics,  
Istanbul Technical University, Istanbul, Turkey  
e-mail: aksugur@itu.edu.tr

G. Inalhan  
e-mail: inalhan@itu.edu.tr

on demand reconnaissance and surveillance for various usage areas from traffic/law-enforcement to border patrol.

The design and development of manned tailsitter concepts had begun in the beginning of 1950s and many experimental aircraft were built and tested during the period between 1950 and 1960s. The pilots of such manned aircraft were in charge of the aircraft's attitude control by only looking at the displays and sensing the behaviors of the aircraft in an upside position with unfamiliar control inputs compared to the helicopter pilots. Moreover, such manned aircraft had control problems especially during landing and hover-to-cruise transition phase, because of having no stability augmentation system to help decrease test pilots' extremely high workload. Therefore, none of the experimental tailsitter concepts have demonstrated an apparent advantage over helicopters or fixed wing aircraft. As a result of several accidents mainly due to the high workload of the test pilots, the development of manned tailsitter projects were suspended after mid 1960s. However, with the help of the advances in both hardware and software based technologies [4, 6, 14], the distinct tailsitter concept was finally successfully developed to take its place among the other VTOL UAVs. Likewise, there are many already mature or under development VTOL concepts. At the present time, OAVs are the most popular ones which can be classified under tailsitter concepts. Allied Aerospace's iStar series Organic Air vehicles (OAV) and Honeywell's electric powered mini class OAV are the foremost examples with a wide range of application around the world. On the other hand, although the OAVs show efficient static-low speed flight due to the shroud and duct around the propeller blades, such ducted fan UAV concepts still have some aerodynamic problems, especially in efficient forward flight regime and hovertor-cruise transition phase. In addition, OAVs with internal combustion engines also suffer from high levels of noise during operation which making these types of tailsitter UAVs unsuitable for "silent intelligence" in urban operational areas. Other examples of mini to mid size tailsitter concepts are T-Wing and Heli-wing. T-Wing was developed at University of Sydney at 2000s [21], and Heliwing was developed by Boeing. However, both of these similar concepts are not suitable size wise for urban applications because of their noisy internal combustion engines. Besides, there are also several micro class tailsitter UAV concepts such as Brigham Young University's tailsitter UAV [13] and University of Arizona's coaxial propeller driven UAV [18]. Nevertheless, such micro size UAVs are not suitable for carrying a "useful" payload and for service in severe weather conditions. Two of the discussed tailsitter concepts are seen in Fig. 1.

Since 2001, related to the recent progress in Lithium based batteries, aircraft designers have started to consider electric powered aircraft concepts [16]. Moreover, advances in brushless DC electric motors have accelerated the development period of such kind of aircraft. Selection of an electric propulsion system for ITU Tailsitter was a major pre-design choice and brings with it the advantages of low noise levels, easier maintenance and the unique capability to autonomously recharge the units from base landing stations. However, for an electric powered vehicle within the mini-UAV class, this unique capability calls for a trade-off between speed limited high power propeller configuration and the power limited high speed electric ducted fan (EDF) system. In this study for mini class UAVs, design optimization and an intuitive thrust-power-airspeed trade-off approach, which leads to a hybrid "propeller-ducted fan propulsion system" design that can achieve maximum horizontal flight time and



**Fig. 1** **a** T-wing concept demonstrator from University of Sydney [1], **b** micro tailsitter UAV prototype from University of Arizona [18]

maximum range for the ongoing ITU Tailsitter Project, is presented. As a result, the hybrid propulsion system, consisting of both propulsion systems, was decided to be used in order to design an “all flight regime” efficient aircraft which meets the desired requirements.

Due to the aim of designing an efficient tailsitter UAV, propulsion system has the highest priority among the design requirements. Since electric powered propulsion system has advantages over internal combustion engines in terms of maintenance and noise level; it has been chosen as one of the requirements. Propeller driven system supplies high thrust to power ratio for VTOL operations. However, the thrust is rapidly decreased as the incoming airspeed is increased. Hence, the performance is decreased at high speed and the system becomes insufficient for cruise flight. On the other hand, Electric EDF systems are capable of producing the same thrust with a higher thrust to power gradient than the propeller systems. Although the power consumption is increased, EDF system’s thrust is necessary only for long range cruise operations.

In ITU Tailsitter, a folding propeller system, located on the nose of the aircraft, is used for hovering, vertical take-off, vertical landing and low-speed transition mode, whereas an EDF system, which is placed between the stabilizers, is used for level and high speed flight modes where the propeller folds onto the fuselage in order to reduce drag when it is turned off. In addition, to calculate the approximate empty weight, instead of the classic method of empty weight fraction, an “aircraft-based” weight modeling and optimization study have been conducted so as to see the most efficient design possible. Initial system performance analysis with candidate propulsion units has shown that up to 40m/s cruise speed and maximum 120 min of flight endurance can be achieved while carrying 1 kg of payload in 10kg of flying system with 3 min of vertical take-off and landing operation duration—a distinct performance in comparison to the same class rotary-wing and OAV alternatives. In the following sections, a trade off analysis is presented where the hybrid-dual propulsion approach with a qualitative analysis of the selected propulsion systems,

is described. This is followed by the design overview and the design and design optimization approach. Finally, the control methodology and the results are denoted.

## 2 Trade-off Analysis

For the most-energy-efficient flight, although a hybrid dual propulsion technique seems to be the ideal approach for this class of UAVs, a two-fold alternative trade-off can be considered in detail.

First trade-off in consideration is the usage of a big EDF-only propulsion system instead of the hybrid system. The reason for such a choice is that we can utilize propeller propulsion system's weight for extra batteries for enhanced mission duration. However, even if the EDF-only configuration seems advantageous over the hybrid propulsion system initially, there are two main problems. The first one is the efficiency problem such that during hover conditions, the states of the aircraft are controlled by the aerodynamic force generator surfaces that are influenced by propulsion system's air-wash. Therefore, placing such force generator vanes with servo actuators behind the EDF unit adds not only extra weight to the empty weight of the aircraft but also reduces the efficiency of the propulsion system both in hover and in cruise condition because of the additional drag. As a result, using the hybrid propulsion system increases the flight efficiency and aircraft's controllability during VTOL and transition operations. Because, similar to the 3D-aerobatic model airplanes; hover, vertical ascent and vertical descent maneuvers are controlled via propeller's induced velocity on relatively large and high deflection angle control surfaces at the tail of the aircraft. Moreover, after transition from vertical ascent to cruise flight, as the EDF system is activated, the folding propellers fold onto the fuselage and the drag component of the propeller propulsion system is minimized.

Second is a trade off between larger ducted fan systems called Organic Aerial Vehicles (OAVs) and the tailsitter aircraft with hybrid propulsion system. Ducted fan UAVs (OAVs) which have shrouded propellers are advantageous over the propeller only systems. Nevertheless, there are three main disadvantages of OAVs. First of all, for the forward flight case, OAVs require excessive thrust because of wing loading and drag force that is created by both the large duct and aircraft's weight component. Second, the variable pitch system adds more weight which results in reduced payload capacity. Third, as a result of increasing parasite drag of the duct; ducted system loses its efficiency as the airspeed increases [20].

Besides the two trade-off explanations, figure of merit (FOM) chart consisting of VTOL only systems, can be seen in Fig. 2. Note that, the point rating is limited between  $-1$  and  $1$ , where  $-1$  is a negative and  $1$  is a positive effect on the total points. “VPP” stands for “Variable Pitch Propeller” in the FOM chart. the concept with hybrid propulsion system is the foremost one which is followed by the single propeller driven and twin propeller driven (T-Wing, Heliwing) tailsitter concepts, as evident from the comparison. Note that, OAV concept has the lowest rank among the other concepts being examined. Therefore, before starting the design process, the FOM chart has shown the advantages of the concept equipped with hybrid/dual propulsion system.

Figure of Merit (FOM)	Agility	Payload capacity	Endurance	High speed performance	Cruise efficiency	Development time	Ease of build	Complexity	TOTAL
One propeller driven	1	1	0	-1	-1	1	1	1	3
Coaxial propeller driven	1	0	-1	-1	0	-1	-1	-1	-4
OAV	-1	0	-1	-1	-1	0	-1	0	-5
Twin propeller driven	1	0	0	-1	0	1	1	0	2
Ducted propeller-at-tail	0	0	-1	-1	0	-1	-1	-1	-5
One propeller driven w/ VPP	1	-1	-1	1	1	-1	-1	-1	-2
Coaxial propeller driven w/ VPP	1	-1	-1	1	1	-1	-1	-1	-2
OAV w/ VPP	-1	-1	-1	0	0	-1	-1	-1	-6
Twin propeller driven w/ VPP	1	-1	-1	1	1	-1	-1	-1	-2
Ducted propeller-at-tail w/ VPP	0	-1	-1	0	1	-1	-1	-1	-4
Hybrid propulsion driven	1	1	1	1	1	1	1	1	8

**Fig. 2** Figure of merit of different types of VTOL tailsitter aircraft

### 3 Hybrid-dual Propulsion Approach

In addition to aerodynamic design, selection of the right propulsion system/component is has utmost importance for the determination of the performance of an aircraft. Comparing the propulsion systems of both “heavy lifter” helicopter and “fast and agile” jet fighter, it can be comprehended that relatively high diameter and low weight loading propeller blades are efficient for hovering whereas relatively small diameter blades having high induced velocity are used as an effective way to reach high speeds. After an inspection of the COTS (Commercial Off The Shelf) equipment available on the market, the hybrid propulsion concept is specified with a folding propeller having a 28.5 in. diameter and an EDF system with a 4 in. diameter placed in a single airframe. In this section, both propulsion systems are described with a qualitative comparison. This is followed by the scaled specific thrust concept, which shows the advantages of combined/hybrid propulsion system quantitatively.

#### 3.1 Qualitative Comparison of the Propulsion Systems

The information given in this section is for clarifying the facts about both type of propulsion systems. General characteristics of both propeller and EDF systems are presented based on the wind tunnel test and analytical calculations.

##### 3.1.1 Propeller Propulsion System

To select the appropriate propeller for an aircraft, all the performance data of the candidate propellers should be carefully analyzed. Although there is a large amount

of available performance data about propellers [5, 19, 23], these propellers are mostly used on commercial or military manned aircraft. On the other hand, there are no sufficiently and systematically catalogued propeller performance charts for use on small scale UAVs, except for some test results [3, 15].

As mentioned before, in ITU tailsitter UAV, the propeller system is considered to be used as main lift generating device during VTOL operations. Thus, to get the highest specific thrust (T/P) value in static and low speed climb/transition phases, a propeller with the largest plausible diameter and a relatively low pitch value should be selected. This is because, propellers with high pitch value are designed for high-speed applications and a high percentage of propeller blades are stalled during low speeds and static condition. In light of these considerations, a RASA brand 28.5 × 12 size (28 in. of diameter and 12 in. of pitch) propeller is selected. However, instead of wind tunnel tests, which give unsatisfactory data due to the experiment room size of ITU wind tunnel, an analytical calculation method is employed.

To derive the characteristics of the selected propeller analytically, physical modeling of the propeller's physical variables, which are pitch distribution, chord distribution and propeller's airfoil data in each section, is defined. After completing the modeling study, JavaProp software [9], which is written by Martin Hepperle, is used to analyze the propeller. JavaProp is a java based application which uses blade element theory as background theory and the validation [10] of JavaProp software has been written by Dr. Martin Hepperle with the help of comparison of the calculations using NACA's test results [22].

### *3.1.2 EDF Propulsion System*

There are many COTS EDF units, which have 2 to 7 blades according to their size and are made from plastic or carbon fiber composite material related to the working conditions, available for the radio-control (R/C) hobby markets [8, 11]. Moreover, diameters of the commercially available EDF systems can vary between 2 and 6 in. However, like small size propellers, hobby purpose EDF units also suffer from lack of any systematically catalogued performance data. Moreover, they do not exhibit ideal ducted fan behavior because of having wider gaps between shroud and blade tips than the full size precisely manufactured ducted fans.

The general purpose and usage of the commercially available EDF systems is to mimic the full size jets on such small size R/C model aircraft, which are aimed to reach relatively high speeds such as 200 mph or more. However, such EDF systems produce less thrust than the propeller systems for a given unit power input at low speeds. This is because the blades of EDF units stall in static and low incoming velocity conditions, related to the flight-specific pitch distribution. Therefore, even though the T/P ratio of EDF systems are quite low at the static condition, second derivative of the T/P curve with respect to the airspeed is lower than the propellers' T/P curve's second derivative. Note that, there is only one unit, Schuebeler's DS51 model with available wind tunnel test measurements which is officially published on the manufacturer's website [12]. Therefore, Schuebeler DS51 EDF unit is selected for ITU Tailsitter.

## 3.2 Scaled Specific Thrust; A Quantitative Approach

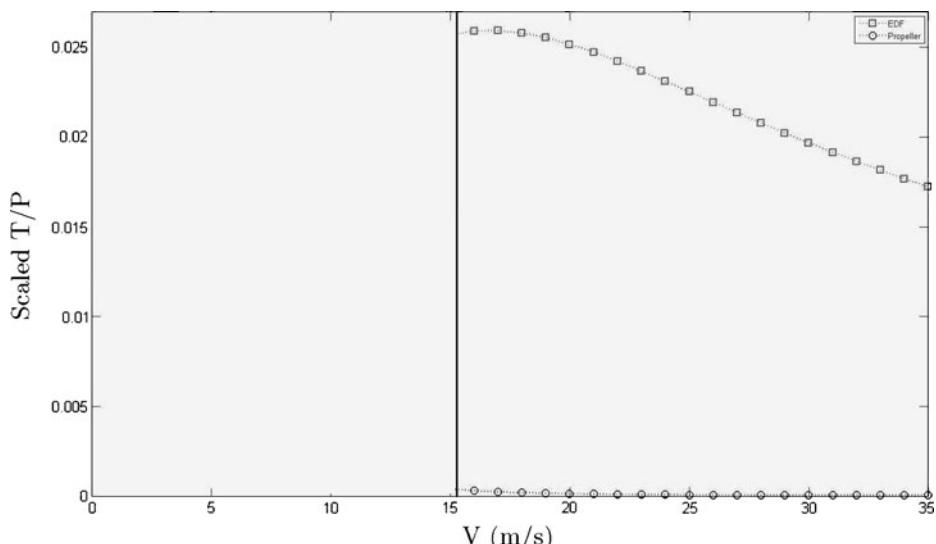
Specific thrust, which is defined as thrust to power ratio, is the preferred method to define and compare the efficiency values of propulsion systems. Moreover, the

specific thrust value is a function of advance ratio that is also a function of angular and airspeed. So, three dimensional complex surface geometry analysis should be conducted in order to see and quantize the performance of both of the propulsion systems. However, with the help of the scaled specific thrust method, which is structured during the design phase of ITU Tailsitter UAV, the three dimensional specific thrust determination problem has scaled down to two dimensional problem. The results are plotted as seen in Fig. 3.

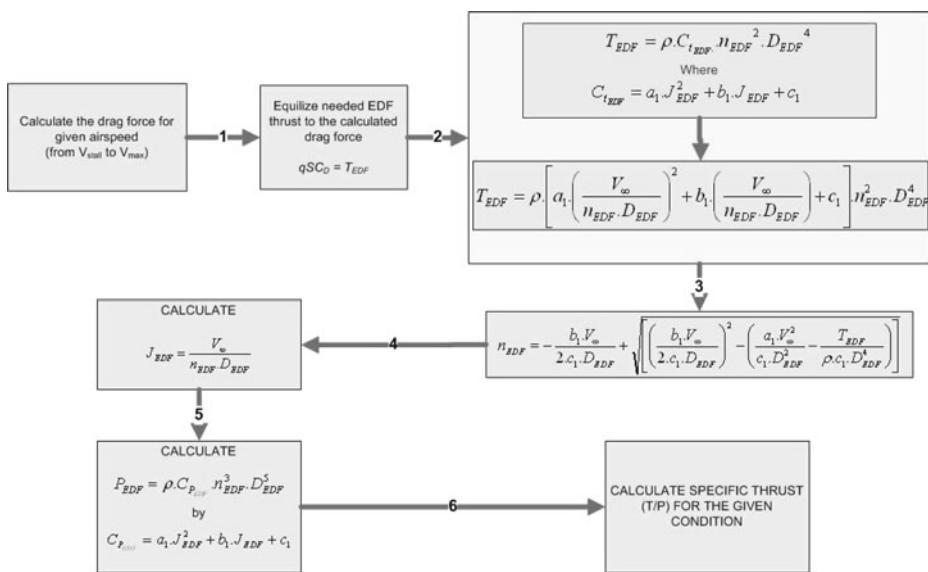
Moreover, the calculation method of scaled specific thrust for both EDF and propeller propulsion systems is seen in Figs. 4 and 5. Note that, for the selected “propeller based” propulsion systems, thrust and power coefficients can be written as a quadratic function of advance ratio. Therefore, the letters  $a_1, b_1, c_1$  and  $a_2, b_2, c_2$ , which are seen in Figs. 4 and 5, are the coefficients of the quadratic equations of EDF and propeller systems respectively.

In Fig. 6, there are two regions, which are valid for our consideration. The first region is zero speed, which is simulated as hovering maneuver; the second region is from the black vertical line, which intersects the stall speed (16 m/s) on the x axis, to the desired maximum airspeed, which is 35 m per second. The pink area between “very-low” speed (0–5 m/s) and stall speed is intentionally left blank. This is because the calculation of exact and optimized transition maneuvers will be done after exact modeling of the aircraft by flight tests.

A more detailed account of the procedure is as follows; the propulsion system only overcomes parasite and induced drag from the stall speed to the maximum airspeed. However, the aircraft’s weight is added to drag component, which propulsion system must overcome during the transition maneuvers. Therefore, the accurate transition phase/airspeed is investigated after calculating the optimized transition maneuvers. As seen from Fig. 6, each of the propulsion systems with any given airspeed condition results in different revolution per minute and advance ratio values. Therefore, the

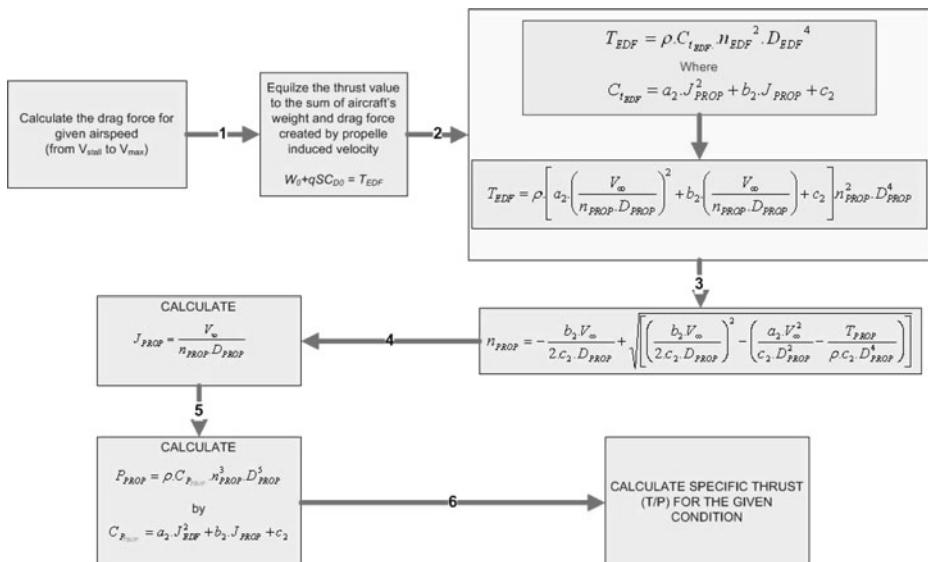


**Fig. 3** Scaled specific thrust comparison of the selected propeller and EDF systems

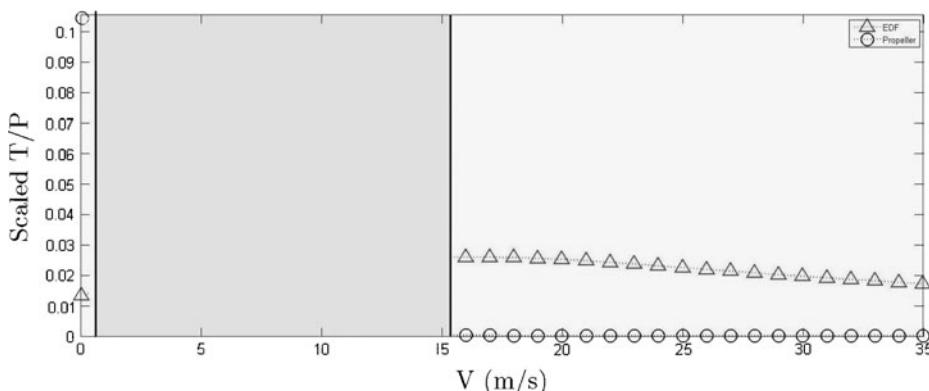


**Fig. 4** Scaled specific thrust calculation methodology for EDF propulsion system

comparison between the propulsion systems can be made outright in Fig. 6. The EDF system is about 64 times more efficient than the propeller system for the whole flight regime from stall to maximum airspeed. Moreover, for static case analysis, specific thrust for both propeller and EDF systems are calculated as 0.1052 and 0.0122. Hence, depending on the static case calculations (for hover and low speed



**Fig. 5** Scaled specific thrust calculation methodology for propeller propulsion system



**Fig. 6** Scaled specific thrust graphic for propeller and EDF propulsion systems in both hover and cruise conditions

climb) it is seen that the propeller system is about 9 times more efficient than the EDF system. However, due to angular velocity restriction, the EDF system can not produce adequate thrust as the propeller system.

As a result, the breakthrough advantage of the hybrid propulsion approach is evident in Fig. 6.

#### 4 Optimization Based Design Methodology

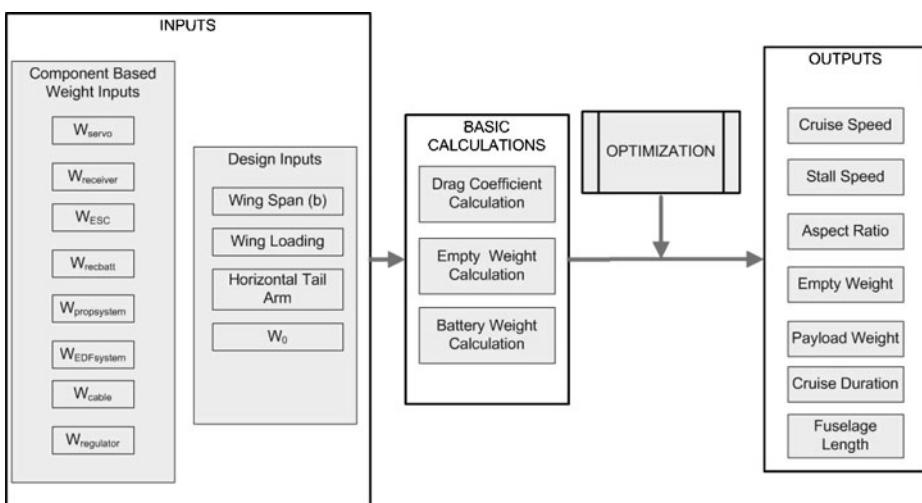
The design philosophy developed hinges on obtaining the maximum possible payload capacity while achieving both high T/W ratio for VTOL, maneuverability and low energy demand per unit operation time (i.e. a low power demand for enhanced endurance). To do this, an aircraft, which has a relatively high cruise speed with vertical take off and landing capabilities, has been delineated. In addition, restrictions placed by the city operational environment were reflected via area and volume limitations before starting the design. In this section, the optimization design methodology for the ITU Tailsitter UAV is summarized.

As indicated in Fig. 7, the design methodology approach consists of three main phases. These phases are inputs, basic calculations and optimization. Input part includes two sub-parts. The first sub-part is called component based constant weight inputs, which includes the weights of the components that are fixed for the design process. The second sub-part is design inputs including wing span, wing loading, maximum take off weight and the horizontal tail arm. In basic calculations part, drag coefficient, empty weight and battery weights are calculated. Optimization part is used to get most desirable design within our criteria and constraints.

##### 4.1 Inputs

###### 4.1.1 Component Based Constant Weight Inputs

In this part, non-variable weight inputs including electronic and power related equipments, are summarized. The components contributing to this category are



**Fig. 7** General design methodology of ITU Tailsitter UAV

servos, a radio receiver, a receiver battery, electronic brushless motor drivers, a voltage regulator, cables and the brushless motors of the propulsion systems. Note that, the sum of the weight of the component based weight inputs are kept constant for each design for whole optimization process. All the components are commercial off-the-shelf (COTS) equipments and seen in Table 1.

#### 4.1.2 Design Inputs

Design inputs consist of four variables; wing span, wing loading, horizontal tail arm and aircraft's maximum takeoff weight. Upper and lower bounds of these variables will be described and shown in optimization subsection.

### 4.2 Basic Calculations

#### 4.2.1 Empty Weight Calculation

During the design process of a manned aircraft, the empty weight fraction value is generally obtained from historical data of the same size/class aircraft. Nevertheless,

**Table 1** Descriptions and weights of the selected components

Component	Specifications	# of piece	Total weight (gr)
Servo	JR-DS8611 digital servo	5	215
Receiver	10 channel, PCM modulation	1	45
Receiver battery	7.4 V lithium-polymer	1	84
Electronic speed controller	Phoenix HV85 brushless ESC	2	230
Voltage regulator	7.4 volts to 6V converter circuit	1	20
Cable	Various size from 12 AWG to 24 AWG	–	57
Propeller unit	Brushless motor and propeller	–	557
EDF unit	Brushless motor and EDF unit	–	450

it's generally hard to find sufficient data to carry out empty weight prediction for relatively novel concepts. Consequently, a quasi exact weight prediction method has been developed to calculate a “more accurate” empty weight value in terms of aircraft size and total weight for ITU Tailsitter.

Empty weight of the aircraft is composed of four main weight components as seen in Eq. 1 and these are described in detail in the following subchapters;

$$W_{empty} = W_{fuselage} + W_{wings} + W_{stabilizers} + W_{structural} \quad (1)$$

**Fuselage Skin Weight Modeling** Before starting the calculations, composite fabrication method had been selected. As mentioned previously, during cruise flight, the folding propeller folds onto the fuselage to reduce the drag and then the EDF system is activated. Moreover, in “cruise to hover” transition phase, EDF system is shut down and the propeller system is reactivated in order to perform a power efficient landing. It is important to note that, unfolding process of the propeller requires a symmetrical fuselage shape to prevent the propeller from damaging the fuselage skin. Therefore, in light of this prediction, a symmetrical airfoil, NACA 642-015, was selected to shape the fuselage geometry. The relation between fuselage surface area and fuselage weight was derived as a next step. The measurements on the selected fuselage airfoil show that the surface area is about 0.303 m<sup>2</sup> for 1 meter of fuselage length. Therefore, the function for the fuselage weight can be written as given in Eq. 2 in terms of surface area and composite skin's surface weight density:

$$W_{fuselage} = 0.303 \cdot l_{fuselage} \cdot \rho_{skincomposite} \quad (2)$$

It should be noted that,  $l_{fuse}$  is the fuselage length in meters and  $\rho_{skincomposite}$  is the composite skin's surface weight density in N/m<sup>2</sup>. However, fuselage length in Eq. 2 is not a sufficient parameter for the optimization problem since fuselage length depends on horizontal tail arm ( $l_{ht}$ ) and root chord of the wing ( $C_{root}$ ) which are variables of the optimization problem. Hence, fuselage length is expressed in terms of these variables as given in Eq. 3:

$$l_{fuselage} = l_{nose} + \frac{1}{4} C_{root} + l_{ht} \quad (3)$$

The variable  $l_{nose}$ , which is determined as a constant value of 0.4 meters, represents the distance, which is assigned by considering propeller's clearance, between the leading edge of the wing root and nose of the fuselage. Therefore, the equation of fuselage weight is derived and seen in Eq. 4.

$$W_{fuselage} = 0.303 \cdot \rho_{skincomposite} \cdot \left( 0.342 + \frac{1}{4} C_{root} + l_{ht} \right) \quad (4)$$

**Wing Weight Modeling** During the calculation of the wing weight function; wing airfoil, taper ratio, wing loading, wingspan are the main parameters. Hence, the calculations have been started with wetted surface area of the wing. Thus, for a meter of chord length, the wetted surface area is calculated as 2.06633 · b m<sup>2</sup>, where b is the wingspan. Following these, the wing skin weight can be expressed as given in Eq. 5:

$$W_{wingskin} = 2.06613 \cdot \bar{c}_{wing} \cdot b \cdot \rho_{skincomposite} \quad (5)$$

It should be noted that, the production method and the material that is planned to be used for both the fuselage and the wing are same. Moreover, the weight of wing

spar, which is made of hollow carbon tube, must be obtained to derive the entire and accurate wing weight. Since the deflection at the tip of the wing is a crucial parameter for the wing stiffness, the tip deflection is restricted as 5 percent of the wingspan in order to be able to sustain 2.5g of loading. According to previous experiences, this loading condition can be modeled by applying a force equal to half the weight of the aircraft to the wing tip when the fuselage is stationary. 2.5g load case for the half wing can be simulated as the tip deflection of a cantilever beam loaded with a concentrated force at the free edge. The deflection is then expressed as given in Eq. 6 in detail:

$$\delta = 0.05 \cdot b = \frac{F \cdot l^3}{12 \cdot E \cdot I} = \frac{\left(\frac{W_0}{2}\right) \cdot \left(\frac{b}{2}\right)^3}{12 \cdot E \cdot I} \quad (6)$$

For the equation given above, E is the modulus of elasticity of spar material and I represents moment of inertia for a circular wing spar section. The moment of inertia can then be calculated in order to find the inner diameter of the spar, which is a hollow carbon tube. Note that, the outer diameter of the spar is restricted as given in Eq. 7:

$$\phi_{outerspar} = \left[ c_{wingtip} \cdot \left( \frac{t}{c} \right) \right] - 0.003 \quad (7)$$

Where  $c_{wingtip}$  represents the wing tip chord,  $\frac{t}{c}$  represents thickness ratio of wing airfoil and the numerical value, 0.003, represents the thickness of skin composite material. Hence, the moment of inertia can be calculated as given in Eq. 8 and the inner diameter of the spar can be expressed as seen in Eq. 9. Finally, the weight of the carbon spar can be calculated in Eq. 10, given as follows:

$$I = \frac{\Pi}{4} \left[ (\phi_{outerspar})^4 - (\phi_{innerspar})^4 \right] = \frac{\left(\frac{W_0}{2}\right) \cdot \left(\frac{b}{2}\right)^3}{12 \cdot E \cdot (b \cdot 0.05)} \quad (8)$$

$$\phi_{innerspar} = \left[ \frac{4 \cdot \left(\frac{W_0}{2}\right) \left(\frac{b}{2}\right)^3}{12 \cdot \Pi \cdot E \cdot (b \cdot 0.05)} - (\phi_{outerspar})^4 \right]^{\frac{1}{4}} \quad (9)$$

$$W_{spar} = \frac{\Pi}{4} \cdot \left[ (\phi_{outerspar})^2 - (\phi_{innerspar})^2 \right] \cdot b \cdot \rho_{spar} \quad (10)$$

For the equation given above,  $\rho_{spar}$  is the density of the carbon material per unit volume. To conclude, after summing skin and spar weights, the weight function for the wing is derived as in Eq. 11 given below:

$$W_{spar} = \frac{\Pi}{4} \cdot \left[ (\phi_{outerspar})^2 - (\phi_{innerspar})^2 \right] \cdot b \cdot \rho_{spar} + (2.06613 \cdot \bar{c}_{wing} \cdot b \cdot \rho_{composite}) \quad (11)$$

**Stabilizer Weight Modeling** The total weight of the stabilizers can be expressed as the summation of horizontal and vertical tail weights. The manufacturing method, used for the stabilizers, is slightly different from the one used for fuselage and wings. In this manufacturing technique, stabilizers are going to be made of high density foam covered with carbon fiber. Thus, wetted area and inner volume of the stabilizers are used to derive the weight function as seen in Eq. 12:

$$W_{stabilizers} = W_{HT} + W_{VT} \quad (12)$$

First of all, the calculation steps for horizontal tail are shown. After that, the calculations of vertical tail weight determination is shown.

$$W_{HT} = S_{HT_{wetted}} \cdot \rho_{skin} + v_{HT} \cdot \rho_{foam} \quad (13)$$

In Eq. 13,  $S_{HT_{wetted}}$  represents the wetted area of horizontal stabilizer,  $\rho_{skin}$  represents the density of stabilizers' composite covering material,  $v_{HT}$  represents the inner volume of horizontal stabilizer and  $\rho_{foam}$  represents the density of the foam that fills the horizontal tail. Horizontal stabilizer area is written in terms of wing area, mean aerodynamic chord, tail arm and horizontal tail volume coefficient and seen in Eq. 14:

$$S_{HT} = \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \quad (14)$$

In the equation given above,  $v_{HT}$  is horizontal tail volume coefficient,  $\bar{c}_{wing}$  is mean aerodynamic chord of the wing,  $S_{wing}$  is wing area and  $l_{HT}$  is horizontal tail arm which is equal to the distance between quarter mean aerodynamic chords of wing and horizontal stabilizer respectively. In ITU Tailsitter airplane, inverted V-tail configuration is used with 25 degree of anhedral on both left and right horizontal stabilizer parts. Therefore, the actual area of horizontal stabilizer in terms of the anhedral angle, can be calculated by using Eq. 15.

$$S_{HT_{actual}} = \frac{1}{\cos 25} \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \quad (15)$$

After the actual horizontal tail area is determined, wetted area can be found with respect to the tail airfoil. As indicated before, NACA0014 is selected for horizontal and vertical stabilizers. Therefore, upon inspecting the geometric properties of the selected airfoil, it is found that wetted area is  $2.133 \text{ m}^2$  per unit planform area. Then, wetted area can easily be found as given in Eq. 16:

$$S_{HT_{wetted}} = 2.133 \cdot \frac{1}{\cos 25} \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \quad (16)$$

After determination of wetted area for horizontal stabilizer, the weight of the composite skin covering material can be found as given in Eq. 17:

$$W_{HT_{skin}} = 2.133 \cdot \frac{1}{\cos 25} \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \cdot \rho_{skin} \quad (17)$$

Next, the weight function for “foam core” must be found in order to derive weight function for horizontal tail. According to the geometrical data, the airfoil side area coefficient for one meter of chord is determined as  $0.000951 \text{ fsm}^2$ . Hence, the volume of horizontal tail is expressed as given in Eq. 18:

$$V_{HT} = \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \left( \frac{1}{\cos 25} \right) \cdot 9.51 \cdot 10^{-4} \quad (18)$$

In order to determine the weight of foam core, the volume is multiplied by the foam density as given in Eq. 19:

$$W_{foamcore} = \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \left( \frac{1}{\cos 25} \right) \cdot 9.51 \cdot 10^{-4} \cdot \rho_{foam} \quad (19)$$

As a result, the total weight of the horizontal stabilizer is derived by summing both foam core and skin composite weights as seen in Eq. 20:

$$\mathbf{W}_{\text{HT}} = \left[ 2.133 \cdot \frac{1}{\cos 25} \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \cdot \rho_{skin} \right] + \left[ \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \left( \frac{1}{\cos 25} \right) \cdot 9.51 \cdot 10^{-4} \cdot \rho_{foam} \right] \quad (20)$$

In similar but slightly different manner, the weight function for vertical tail is derived by using the same steps and obtained as Eq. 21:

$$\mathbf{W}_{\text{VT}} = \left[ \left( \frac{v_{VT} \cdot b \cdot S_{wing}}{l_{VT}} \right) - \left( 2.133 \cdot \tan 25 \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \cdot \rho_{skin} \right) \right] + \left[ \left( \frac{v_{VT} \cdot b \cdot S_{wing}}{l_{VT}} \right) - \left( \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \tan 25 \cdot 9.51 \cdot 10^{-4} \cdot \rho_{foam} \right) \right] \quad (21)$$

Finally, entire weight function for stabilizers is derived by summing horizontal tail and vertical tail weight functions. By inserting Eqs. 20 and 21 in the equation given below, the weight function for stabilizers is obtained as seen in Eq. 22:

$$\mathbf{W}_{\text{stabilizer}} = \left[ 2.133 \cdot \frac{1}{\cos 25} \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \cdot \rho_{skin} \right] + \left[ \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \left( \frac{1}{\cos 25} \right) \cdot 9.51 \cdot 10^{-4} \cdot \rho_{foam} \right] + \left[ \left( \frac{v_{VT} \cdot b \cdot S_{wing}}{l_{VT}} \right) - \left( 2.133 \cdot \tan 25 \cdot \frac{v_{HT} \cdot \bar{c}_{wing} \cdot S_{wing}}{l_{HT}} \cdot \rho_{skin} \right) \right] + \left[ \left( \frac{v_{VT} \cdot b \cdot S_{wing}}{l_{VT}} \right) - \left( \frac{v_{HT} \cdot S_{wing} \cdot c_{wing}}{l_{HT}} \cdot \tan 25 \cdot 9.51 \cdot 10^{-4} \cdot \rho_{foam} \right) \right] \quad (22)$$

**Structural Weight Modeling** Structural weight consists of the weight of spar box, bulkheads, longerons and glue. Spar box weight is assumed to be a constant, while glue weight is approximated as 10% of bulkheads, longerons and spar box weights. Moreover, for building fuselage frame, it is planned to locate four longerons along the fuselage and one bulkhead per each 0.15 m of fuselage length. The total structural weight is then expressed as given in Eq. 23:

$$\mathbf{W}_{\text{structure}} = 1.1 \cdot \left( W_{\text{bulkhead}} \cdot \frac{l_{\text{fuselage}}}{0.15} + 4 \cdot l_{\text{fuselage}} \cdot W_{\text{longeron}} + W_{\text{sparbox}} \right) \quad (23)$$

Where  $W_{\text{bulkhead}}$  is the assigned constant average weight of a bulkhead,  $W_{\text{longeron}}$  is the weight of a carbon longeron per unit length and  $W_{\text{sparbox}}$  is the preassigned constant weight value of the spar box.

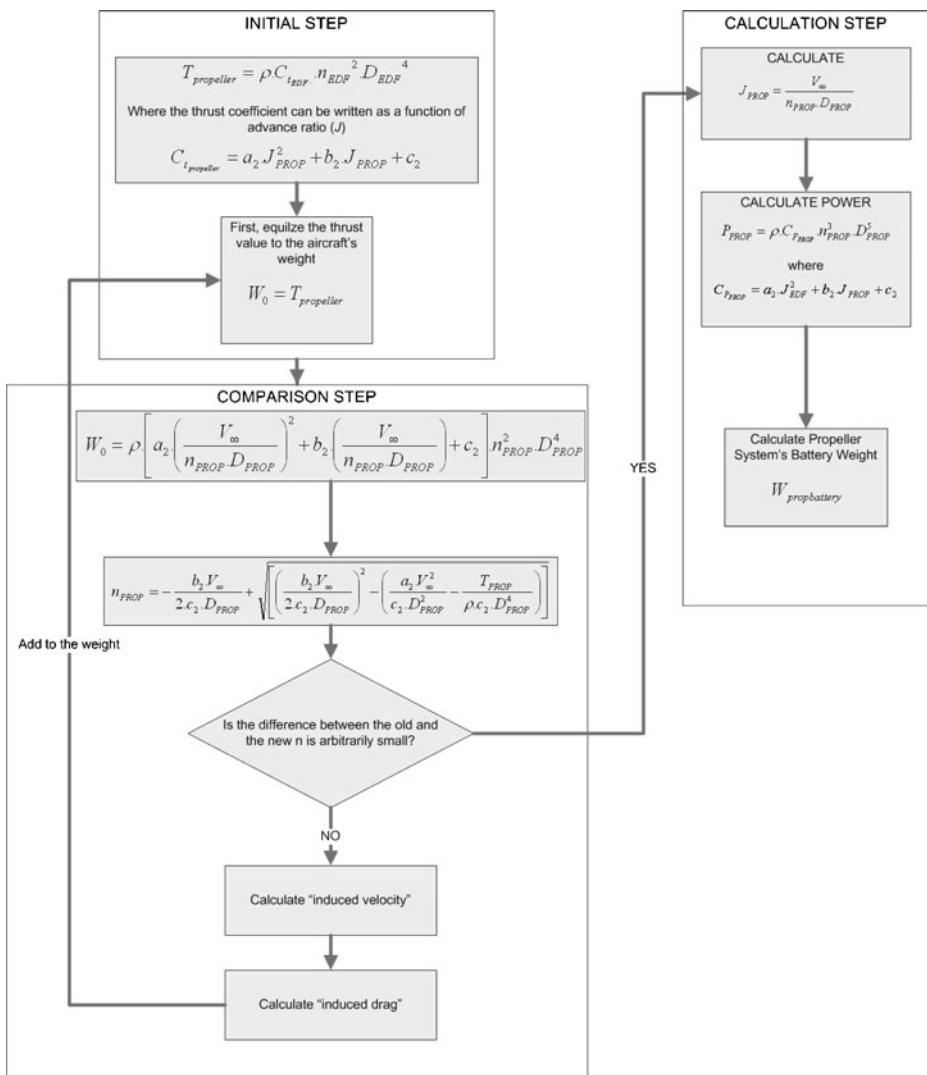
#### 4.2.2 Battery Weight Calculation

**Battery Weight Modeling of The Propeller Propulsion System** The propeller propulsion system is responsible for VTOL operations including hovering, low speed vertical climb and descent. To simplify the calculations, the total VTOL operation duration is set as 3 min with only hover mode. This is because, even though the

propeller consumes more energy than hovering mode during vertical climb; the energy consumption is reduced to a level below that of hovering during the vertical descent maneuver.

For hover mode, the thrust produced by the propeller is equal to the sum of weight of the aircraft and the drag force created by the propeller's induced velocity, Which is calculated using helicopter theory [2]. The battery weight determination logic for the propeller system is shown in Fig. 8 step by step.

The battery weight of the propeller propulsion system is a function of the VTOL operation duration, propeller's characteristics, drag force created by geometry of the aircraft, and the takeoff weight of the aircraft. Note that, in order to calculate the



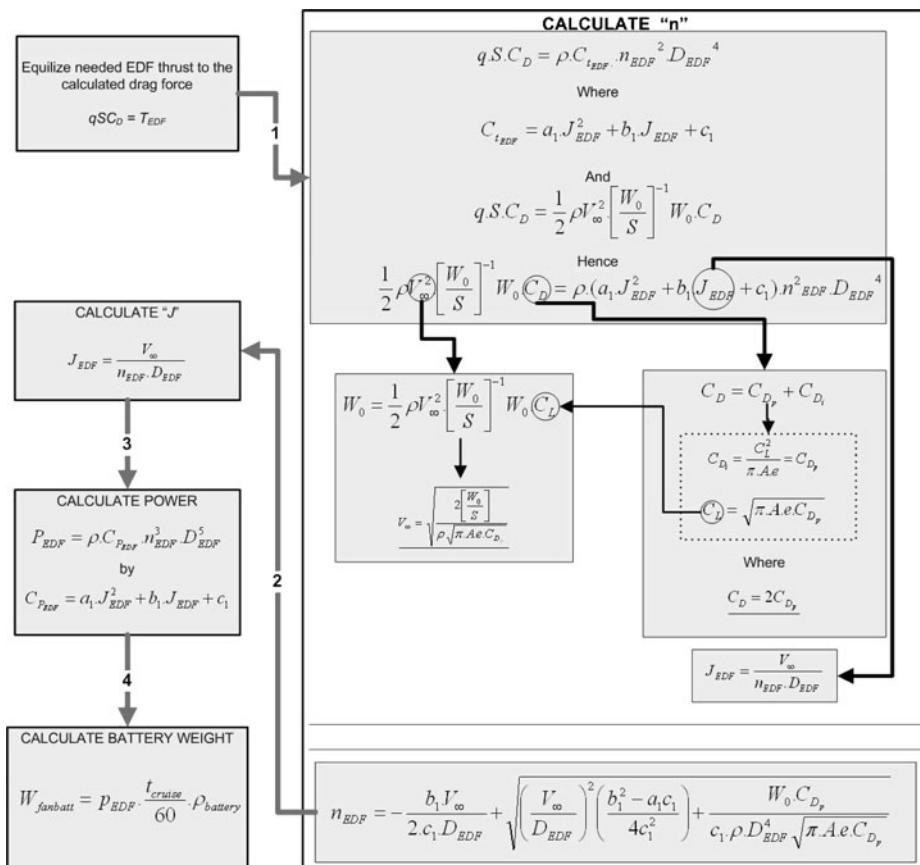
**Fig. 8** Battery weight calculation steps for propeller propulsion system

“worst case” and simplify the calculations, instead of the parasite drag coefficient that is caused by the “induced velocity wetted” part of the aircraft, the parasite drag coefficient for the whole aircraft is used.

**Battery Weight Modeling of The EDF Propulsion System** The selected EDF unit is mainly dedicated to cruise flight. For this reason, thrust generated by the EDF unit is determined as equal to the drag force on the aircraft. Therefore, the battery weight calculation methodology is based on cruise flight and seen in Fig. 9.

#### 4.2.3 Drag Force Modeling

The drag force of an aircraft can be written as the summation of both parasite(zero lift) and induced drags. In ITU Tailsitter design, induced drag coefficient is equalized to the parasite drag coefficient so as to fly at “minimum thrust” level. According to this drag coefficient assumption, parasite drag coefficient is the only variable that must be formulated for each design candidate during optimization process. To do that, the parasite drag coefficient of each component is calculated using the empirical



**Fig. 9** Battery weight calculation for EDF propulsion unit

“component buildup” method described by Raymer [17]. According to Raymer [17] component buildup method is used to calculate drag coefficient for sub-sonic flight and counts on both flat plate skin friction and component form factor (pressure drag due to the viscous separation). Related to component buildup method, the drag coefficient of each component can be described as follow:

$$C_{D_p} = \frac{\sum C_{f_c \cdot FF_c \cdot Q_c \cdot FF_c} S_{ref}}{S_{ref}} + C_{D_{mics}} + C_{D_{LP}} \quad (24)$$

In Eq. 24,  $C_f$  denotes flat-plate skin friction coefficient, FF represents form factors, Q indicates interference factor,  $S_{wet}$  is wetted area of the selected component and  $S_{ref}$  is the reference wing area. Where the subscript “c” indicates that those values are different for each component. In the following descriptions, the equations used for each component of the aircraft are shown.

*Drag Force Modeling for Aerodynamic Surfaces* The parasite drag coefficient for the aerodynamic surfaces,  $C_{D_{aero}}$ , composed of three discrete parts; wing, horizontal stabilizer and vertical stabilizer, and figured out in Eq. 25:

$$C_{D_{aero}} = \frac{1}{S_{ref}} \cdot [C_{f_w \cdot FF_w \cdot Q_w \cdot FF_w} S_{wet_w} + C_{f_{HT} \cdot FF_{HT} \cdot Q_{HT} \cdot FF_c} S_{wet_{HT}} + C_{f_{VT} \cdot FF_{VT} \cdot Q_{VT} \cdot FF_{VT}} S_{wet_{VT}}] \quad (25)$$

Flat-plate skin friction varies depending on the type of flow, laminar or turbulent, over the surfaces. For wing and stabilizers, turbulent flow assumption has been made and flat-plate skin friction coefficient is written as Eq. 26:

$$C_f = \frac{0.455}{(\log_{10} Re)^{2.58} + (1 + 0.144 \cdot M^2)^{0.65}} \quad (26)$$

Where “Re” represents Reynolds number, which is taken 500.000 for wings and 300.000 for stabilizers; “M” indicates mach number, which is selected 0.1 as constant for each design during optimization. To continue, form factor(FF) equation for wing and tails is written in Eq. 27.

$$FF = \left[ 1 + \frac{0.6}{\left(\frac{x}{c}\right)_m} \cdot \left(\frac{t}{c}\right) + 100 \cdot \left(\frac{t}{c}\right)^4 \right] \cdot [1.34 M^{0.18} (\cos \Lambda_m)^{0.28}] \quad (27)$$

In Eq. 27, the term “ $(\frac{x}{c})_m$ ” is the chordwise location of the airfoil maximum thickness point, which is 0.3 for the selected airfoils.  $\Lambda_m$  refers to the sweep of the maximum thickness line, which are 0 and 23 degrees for wing and stabilizers used in ITU Tailsitter respectively. The interference factor, “Q”, is chosen 1.1 for both wing and stabilizers.

*Drag Force Modeling for Fuselage* To calculate the drag coefficient for fuselage, the steps must be followed are similar to the steps followed in wing and stabilizer calculations. However, there’s a change on form factor estimation. The form factor for fuselage or smooth canopy is calculated using Eq. 28.

$$FF = \left( 1 + \frac{60}{f^3} + \frac{f}{400} \right) \quad (28)$$

Where;

$$f = \frac{l}{d} = \frac{l}{\sqrt{\left(\frac{4}{\pi}\right) A_{max}}} \quad (29)$$

### 4.3 Optimization

#### 4.3.1 Defining the Optimization Problem

During the preliminary design phase, it has been observed that an aircraft which carries much more payload and flies longer would be challenging to design. Therefore, optimization process is focused on maximizing payload weight and cruise duration. Since it is not possible to optimize all the design parameters, the parameters with crucial effect are selected as primal variables of the optimization problem. Maximum Takeoff Weight ( $W_0$ ), Wing Loading ( $\frac{W}{S}$ ), Wing Span ( $b$ ), Horizontal Tail Arm ( $l_{HT}$ ) and Fan Battery Weight ( $W_{fanbattery}$ ) are determined as primal variables. The boundary constraints of those variables are going to be discussed in the 'Formulation Section'. In addition, the other design constraints for the optimization problem will also form a part of the discussion in the same section. Consequently, the optimization problem can be classified as multiobjective, multidisciplinary, constrained and continuous. It is a multiobjective problem because the objective is having a maximum payload capacity with maximum cruise duration. It is a multidisciplinary optimization problem because it consists of aerodynamics, propulsion, structure and design. It is a constrained optimization since it includes both boundary and design constraints which are going to be discussed later. It is a continuous problem since the variables are free to change within the constraint space. As it was stated before, the objectives of the optimization for the ITU Tailsitter UAV can be listed as follows;

- Maximization of payload weight
- Maximization of cruise duration

Assuming the Maximum Takeoff Weight (MTOW) as a constant; the objectives, given above, are in contradiction with each other. Since maximization demand on payload weight gives the minimum cruise duration while maximization of cruise duration, which also means maximization of the battery weight, minimizes the payload weight. In light of such relations between the given objectives, the aim is to find the most suitable configuration by the optimization variables satisfying the design constraints.

#### 4.3.2 Requirements, Variables and Constraints of the Optimization Problem

Before beginning the design process, design requirements were determined as seen in Table 2.

At the beginning of the optimization problem, MTOW, wing loading, wingspan, horizontal tail arm and fan battery weight, which are listed in Table 3, are determined to be primal variables. The constraints of the primal variables are then considered as given in Table 3 with the reasoning behind them.

In addition to the side constraints, some operational and aerodynamic restrictions are also applied to the optimization problem to obtain the desired design. At the beginning, stall and cruise speeds are assumed to be crucial parameters for

**Table 2** Design requirements for the optimization problem

Minimum range	20 km
Minimum operation duration	30 min
Operation altitude	1 km
Maximum airspeed	50 m/s
Maximum operation condition wind	15 m/s
Maximum VTOL operation area	2 m × 2 m
Minimum payload weight	0.8 kg

operational capability. Stall speed is limited up to 20 m/s whereas cruise speed is limited up to 50 m/s. Moreover, in order to prevent aircraft stall due to the gust effects during cruise flight or approach, the cruise speed must be at least 5 m/s more than the stall speed. Second, although cruise duration and payload weight are being maximized via the optimization algorithm, there are lower limits which come from the design requirements.

In Table 2, it was stated that payload weight must be more than 0.8 kg, where the cruise duration must be at least 30 min. Next, in order to have an efficient, easily controllable and non-stubby design, the fuselage length is determined to be less than the wingspan. Moreover, aspect ratio's minimum value is set as 4. Consequently, all design constraints can be classified into three groups as operational capability, design requirements and geometrical limits. The determined design constraints of the optimization problem are then summarized in Table 4.

#### 4.3.3 Mathematical Formulation of the Problem and Objective Function

In ITU Tailsitter design process, the main aim is to design an airplane which is configurable with different payload weights. To see the performance of the aircraft with different types (weight) of payloads, two objective functions have been defined; payload weight and cruise time. Therefore, in this multi-objective optimization problem, maximizing both of our objective functions is the main purpose.

**Table 3** Primal variables and side constraints for the optimization problem

Primal variable	Lower bound	# upper bound	Explanation (gr)
MTOW ( $W_0$ )	30 N	100 N	Lower bound is determined according to the known weights where the upper bound is set by the structural restrictions of the propeller.
Wing loading $\left(\frac{W_0}{S}\right)$	100 $\frac{N}{m^2}$	$\frac{N}{m^2}$	The limits are based on the similar types of UAVs
b	1 m	2 m	The lower limit is set by wing efficiency issues and the upper limit is restricted by the VTOL operation area.
$l_{HT}$	0.6 m	1.5 m	The lower limit is determined by considering the tail size and the upper limit is set by the structural limits during vertical landing phase.
$W_{fanbattery}$	3 N	30 N	The boundaries are set by the previous experiences

**Table 4** Design constraints for the optimization problem

Operation capability	$V_{stall} \leq 30 \frac{m}{s}$
	$V_{cruise} \leq 50 \frac{m}{s}$
	$V_{cruise} - V_{stall} \geq 5 \frac{m}{s}$
Design requirements	$t_{cruise} \geq 30 \text{ min}$
	$W_{payload} \geq 8N$
Geometrical limits	$A \geq 4$
	$l_{fuselage} \leq b$

Before going into the details of objective functions, some descriptions are necessary to lay out the components of the functions. Aircraft's maximum takeoff weight ( $W_0$ ) can be expressed as in Eq. 30:

$$W_0 = W_{empty} + W_{payload} + W_{fanbattery} + W_{propbattery} + W_{inputs} \quad (30)$$

Where  $W_{inputs}$  is described in Section 4.1.1 and can be reproduced here as Eq. 31:

$$\begin{aligned} W_{inputs} = & W_{servo} + W_{rec} + W_{recbatt} + W_{ESC} + W_{regulator} + W_{cable} \\ & + W_{propunit} + W_{EDFunit} \end{aligned} \quad (31)$$

In Eq. 31;  $W_{servo}$ ,  $W_{rec}$ ,  $W_{recbatt}$ ,  $W_{ESC}$ ,  $W_{regulator}$ ,  $W_{cable}$ ,  $W_{propunit}$ ,  $W_{EDFunit}$  denotes the weights of servos, receiver, electronic speed controller of the brushless motors, voltage regulator, cables, propeller propulsion unit and EDF unit respectively. In the light of the given formulations, one of the objective functions can be expressed in a compact form in Eq. 32. Note that,  $W_{inputs}$ ,  $W_{propbattery}$ ,  $W_{fanbattery}$  and  $W_{empty}$  were described in the previous chapters as functions or invariant values.

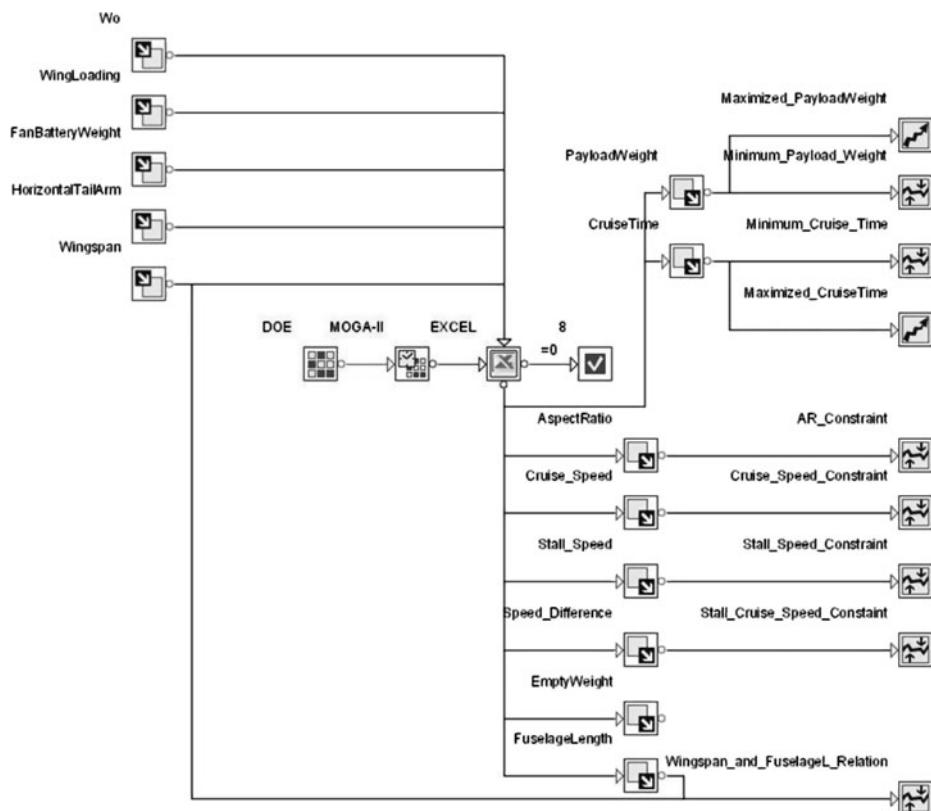
$$\begin{aligned} \text{First Objective Function} \longrightarrow W_{payload} = & W_0 - [W_{empty} + W_{payload} + W_{fanbattery} \\ & + W_{propbattery} + W_{inputs}] \end{aligned} \quad (32)$$

The second objective function is the cruise time for EDF propulsion system. As seen in Fig. 9, the general process to determine the battery weight for the EDF system was summarized, but not formulated. Therefore, cruise time formulation, which is seen in Eq. 33, is obtained with the help of the battery weight equation.

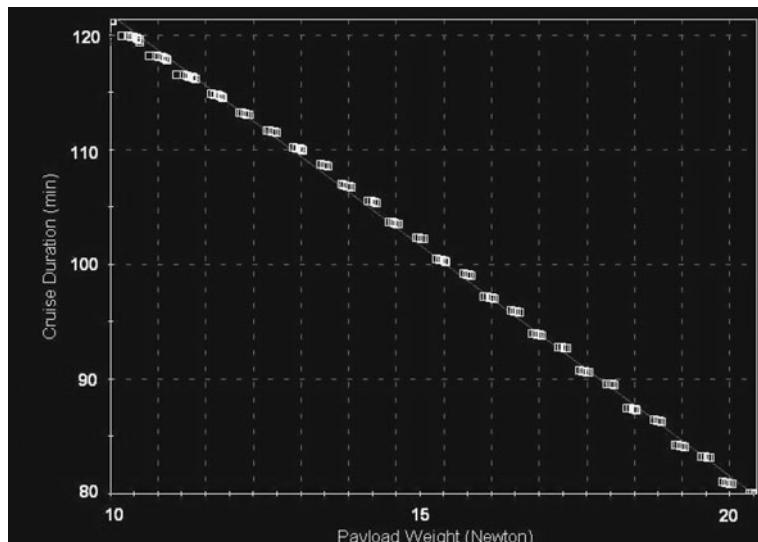
$$\begin{aligned} t_{cruise} = & \frac{60 \cdot W_{fanbattery}}{\rho \cdot \rho_{battery} \cdot (a_2 \cdot J_{EDF}^2 + b_2 \cdot J_{EDF} + c_2) \cdot D_{EDF}^5} \\ & \cdot \frac{1}{\left[ \frac{-b_1 \cdot V_\infty}{2c_1 \cdot D_{EDF}} + \sqrt{\left( \frac{V_\infty}{D_{EDF}} \right)^2 \left( \frac{b_1^2 - a_1 c_1}{4c_1^2} \right) + \frac{W_0 \cdot C_{Dp}}{c_1 \rho D_{EDF}^4 \sqrt{\pi \cdot A \cdot e \cdot C_{Dp}}} } \right]^3} \end{aligned} \quad (33)$$

#### 4.3.4 Optimization Methodology

After the optimization problem is discussed in detail, optimization process becomes ready to be carried through.



**Fig. 10** Flowchart for the optimization problem



**Fig. 11** Pareto graphic resulted by the optimization study

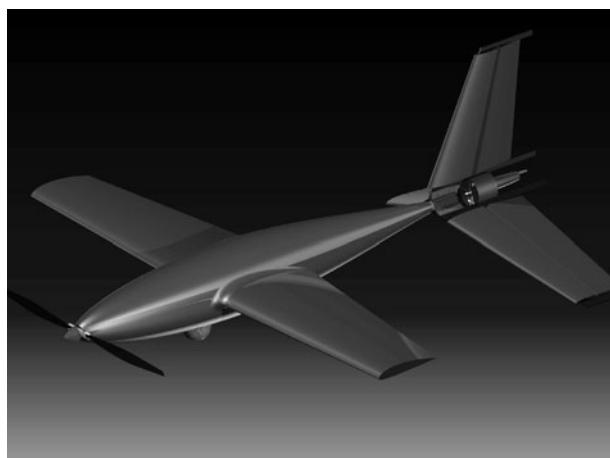
**Table 5** Specifications of ITU Tailsitter UAV

Wing span	1.7	m
Wing loading	200	N/m <sup>2</sup>
$W_0$	10	kg
Payload weight	~1	kg
Cruise duration	2	hours
Range	~90	km

Instead of developing a new code, commercially available software is preferred for the multidisciplinary design optimization of ITU Tailsitter UAV. The optimization process is decided to be performed using Esteco ModeFRONTIER 3.2 [7] as a result.

As seen in Fig. 10, the flowchart is started by adding five input nodes, which are also the previously considered optimization variables ( $W_0$ , Wing Loading, Fan Battery Weight, Horizontal Tail Arm and Wing Span). The side constraints of optimization variables are then applied to these input nodes and seven output nodes are added to the flowchart. Aspect Ratio, Cruise Speed, Stall Speed, Speed Difference and Fuselage Length output nodes are used in order to apply design constraints which are previously given in Table 4. More, Payload Weight and Cruise Time output nodes are the results of objective functions. In addition, minimum mission requirements of the optimization problem, which are given in Table 4, are also applied to these output nodes. The last output node, Empty Weight, is connected in order to monitor the empty weight of the aircraft. Multi Objective Genetic Algorithm (MOGA) is preferred for the optimization algorithm.

The results for the variables strongly depend on the optimization methodology. Thus, the methodology, which is going to be used with the optimization driver, is very important to successfully obtain an optimum design. Moreover, the Design of Experiments (DOE) node and optimization algorithm are considered carefully before starting the optimization process.

**Fig. 12** 3D CAD drawing of ITU Tailsitter UAV

## 5 Results

As stated previously, two objective functions had been determined in order to see the performance of the aircraft with different payload weights. With the help of the pareto chart, the optimization results for different configurations are seen in Fig. 11.

The final specifications and CAD drawing screenshot of ITU Tailsitter UAV are seen in Table 5 and Fig. 12, respectively.

According to the results, the tailsitter design with hybrid propulsion system has both payload and cruise duration advantages over same-class rotary wing and fixed wing alternatives. So, it shows that the accurate selection of propulsion system is one of the most vital parameters in aircraft design.

## 6 Conclusions

In this work, the design optimization study of a tailsitter aircraft with a revolutionary hybrid/dual propulsion system has been described. The results obtained in this paper are based on analytical calculations on the propeller propulsion system, experimental data on the EDF propulsion system and the design inputs which are in close relation with both design constraints and design criteria. Initial system performance analyses with candidate propulsion units indicate that up to 27.5 m/s cruise speed and maximum 2 hours of flight endurance, including 3 min of vertical take-off and landing duration, can be achieved while carrying a 1 kg payload and 90 km of range - a markedly superior performance in comparison to the similar class rotary-wing and OAV alternatives in the same class. As seen in Figs. 13 and 14 Prototyping of ITU Tailsitter is almost completed and flight tests are going to be conducted so as to compare real and the calculated performance.



**Fig. 13** Upper fuselage and left wing photograph of ITU Tailsitter

**Fig. 14** ITU Tailsitter UAV, completed view



## References

1. University of Sydney: T-wing Tailsitter UAV webpage. <http://www.aeromech.usyd.edu.au/uav/twing/>. Accessed 20 Jan 2009 (2006)
2. Mayers, G.C., Gessow, A.: Aerodynamics of the Helicopter, 8th edn. Frederick Ungar, New York (1985)
3. Asson, K., Dunn, P.: Compact dynamometer system that can accurately determine propeller performance. *J. Aircr.* **29**(1), 8–9 (1992)
4. Ates, S., Bayezit, I., Inalhan, G.: Design and hardware-in-the-loop integration of a UAV microavionics system in a manned—unmanned joint airspace flight network simulator. *J. Intell. Robot. Syst.* **54**(1–3), 359–386 (2009)
5. Bass, R.M.: Small scale wind tunnel testing of model propellers. In: Aerospace Sciences Meeting, 24th, Reno, NV (1986)
6. Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., Goodrich, M.: Autonomous vehicle technologies for small fixed wing UAVs. *AIAA J. Aerosp. Comput. Inf. Commun.* **2**(1), 92–108 (2005)
7. Esteco: Esteco modefrontier official webpage. <http://www.esteco.com/> (2009)
8. Electric Ducted Fans: Ductedfans.com online store. [http://www.ductedfans.com/impellers\\_ducted\\_fans.html](http://www.ductedfans.com/impellers_ducted_fans.html). Accessed 20 Jan 2009 (2009)
9. Hepperle, M.: Javaprop software. <http://www.mh-aerotools.de/airfoils/javaprop.htm> (2006)

10. Hepperle, M.: Javaprop software validation tests. [http://www.mh-aerotools.de/airfoils/jp\\_validation.htm](http://www.mh-aerotools.de/airfoils/jp_validation.htm) (2008)
11. Hobby-Lobby International Inc.: Electric ducted fans. <http://www.hobby-lobby.com/ductfan.htm>. Retrieved 20 Jan 2009 (2009)
12. Schuebeler Jets: Schuebeler ds51 wind tunnel test. [http://www.schuebeler-jets.com/images/stories/vergleich\\_windkanal2.jpg](http://www.schuebeler-jets.com/images/stories/vergleich_windkanal2.jpg). Retrieved 20 Jan 2009 (2009)
13. Knoebel, N., Osborne, S., Snyder, D., McLain, T., Beard, R., Eldredge, A.: Preliminary modeling, control, and trajectory design for miniature autonomous tailsitters. AIAA Guidance, Navigation, and Control Conference and Exhibit (2006)
14. Knoebel, N.B.: Adaptive quaternion control of a miniature tailsitter UAV. Ph.D. thesis, Brigham Young University (2007)
15. Merchant, M.P., Miller, L.S.: Propeller performance measurement for low Reynolds number UAV applications. AIAA **1127**, 2006 (2006)
16. Torres, G.E., Mueller, T.J., Srull, D.W.: Introduction to the Design of Fixed-Wing Micro Aerial Vehicles, chap. eee., pp. 39–107. AIAA (2006)
17. Raymer, D.P.: Aircraft Design: A Conceptual Approach, chap. Aerodynamics, 2nd edn., pp. 280–288. American Institute of Aeronautics and Astronautics, New York (1992)
18. Shkarayev, S., Moschetta, J.M., Bataille, B.: Aerodynamic design of micro air vehicles for vertical flight. *J. Aircr.* **45**(5), 1715–1724 (2008)
19. Steffko, G.L., Bober, L.J., Neumann, H.E.: New Test Techniques and Analytical Procedures for Understanding the Behavior of Advanced Propellers. NTIS, Springfield, VA (USA), 24 (1983)
20. Stinton, D.: The Design of the Aeroplane, chap. Reciprocating Engines, pp. 308–310. BSP Professional, Oxford (1993)
21. Stone, R.H.: The T-wing tail-sitter research UAV. In: International Powered Lift Conference, Williamsburg, Virginia (2002)
22. Theodorsen, T., Stickle, G.W., Brevoort, M.J., Langley Aeronautical Laboratory: Characteristics of six propellers including the high-speed range. National Advisory Committee for Aeronautics (1937)
23. Wood, D.H., NACA, Langley Research Center: Full-scale tests of metal propellers at high tip speeds. National Advisory Committee for Aeronautics, Washington, DC (1932)