

# Unmanned Aerial Vehicles (UAV) and Drones



Edited by: Zoran Gacovski



# **Unmanned Aerial Vehicles (UAV) and Drones**



# **Unmanned Aerial Vehicles (UAV) and Drones**

*Edited by:*

**Zoran Gacovski**



[www.arclerpress.com](http://www.arclerpress.com)

## **Unmanned Aerial Vehicles (UAV) and Drones**

*Zoran Gacovski*

### **Arcler Press**

224 Shoreacres Road  
Burlington, ON L7L 2H2  
Canada  
[www.arcлерpress.com](http://www.arcлерpress.com)  
Email: [orders@arcлерeducation.com](mailto:orders@arcлерeducation.com)

### **e-book Edition 2021**

ISBN: 978-1-77407-976-8 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated. Copyright for individual articles remains with the authors as indicated and published under Creative Commons License. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data and views articulated in the chapters are those of the individual contributors, and not necessarily those of the editors or publishers. Editors or publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

**Notice:** Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

**© 2021 Arcler Press**

ISBN: 978-1-77407-773-3 (Hardcover)

Arcler Press publishes wide variety of books and eBooks. For more information about Arcler Press and its products, visit our website at [www.arcлерpress.com](http://www.arcлерpress.com)

# **DECLARATION**

Some content or chapters in this book are open access copyright free published research work, which is published under Creative Commons License and are indicated with the citation. We are thankful to the publishers and authors of the content and chapters as without them this book wouldn't have been possible.



## ABOUT THE EDITOR



**Dr. Zoran Gacovski** has earned his PhD degree at Faculty of Electrical engineering, Skopje. His research interests include Intelligent systems and Software engineering, fuzzy systems, graphical models (Petri, Neural and Bayesian networks), and IT security. He has published over 50 journal and conference papers, and he has been reviewer of renowned Journals. Currently, he is a professor in Computer Engineering at European University, Skopje, Macedonia.



# TABLE OF CONTENTS

---

|  |           |
|--|-----------|
| <i>List of Contributors</i> .....  | xv        |
| <i>List of Abbreviations</i> .....   | xxiii     |
| <i>Preface</i> .....   | xxv       |
| <br><b>Section 1 Design and Development of UAV-s</b>   |           |
| <b>Chapter 1 Design and Development of a Fly-by-Wireless UAV Platform.....</b>                       | <b>3</b>  |
| Introduction.....  | 3         |
| Wireless Network Technologies .....  | 5         |
| Global Electronics Architecture.....   | 7         |
| Onboard Wireless System.....   | 8         |
| Experimental Results.....  | 13        |
| Conclusion .....   | 15        |
| References .....   | 17        |
| <b>Chapter 2 A Review of Control Algorithms for Autonomous Quadrotors .....</b>                      | <b>19</b> |
| Abstract .....   | 19        |
| Introduction .....   | 20        |
| Mathematical Model .....   | 21        |
| Survey of Control Algorithms .....   | 22        |
| Comparison of Control Algorithms .....   | 30        |
| Discussion and Conclusion .....  | 30        |
| References .....   | 32        |
| <b>Chapter 3 Central Command Architecture for High-Order Autonomous Unmanned Aerial Systems.....</b> | <b>37</b> |
| Abstract .....   | 37        |
| Introduction.....  | 38        |
| Method.....  | 40        |

|  |            |
|--|------------|
| Results .....  | 46         |
| Summary And Conclusions .....  | 51         |
| Appendix: The Shortest Lines Between Two Point Sets Do Not Cross.....            | 52         |
| References .....   | 56         |
| <b>Chapter 4 Quadcopter Design for Payload Delivery.....</b>                     | <b>59</b>  |
| Abstract .....   | 59         |
| Introduction .....   | 60         |
| Methodology .....  | 61         |
| Simulation .....   | 67         |
| Simulation Results And Discussion .....  | 70         |
| Conclusion And Future Work.....  | 72         |
| Conflicts Of Interest.....   | 74         |
| References .....   | 75         |
| <br><b>Section 2 Trajectory Control Techniques</b>                               |            |
| <b>Chapter 5 Advanced UAV Trajectory Generation: Planning and Guidance .....</b> | <b>79</b>  |
| Introduction .....   | 79         |
| Motion-Planning Methodologies .....  | 81         |
| Trajectory Generation And Guidance Module - $Tg^2m$ .....                        | 86         |
| $Tg^2m$ Simulation .....   | 105        |
| Final Observations .....   | 109        |
| References .....   | 110        |
| <b>Chapter 6 Modelling and Control Prototyping of Unmanned Helicopters .....</b> | <b>113</b> |
| Introduction .....   | 113        |
| Helicopters Flight Principles .....  | 115        |
| Model Description .....  | 117        |
| Identification Results .....   | 127        |
| Control Prototyping .....  | 130        |
| Control Results .....  | 136        |
| Final Observations And Future Work .....   | 138        |
| References .....   | 140        |

|  |  |            |
|--|--|------------|
| <b>Chapter 7</b>   | <b>Contour Based Path Planning with B-Spline Trajectory Generation for Unmanned Aerial Vehicles (UAVs) over Hostile Terrain.....</b> | <b>141</b> |
|  | Abstract .....   | 141        |
|  | Introduction .....   | 142        |
|  | Modelling.....   | 144        |
|  | Problem Formulation .....  | 149        |
|  | Generation Of Nodes .....  | 151        |
|  | Implementation .....   | 151        |
|  | Conclusions And Future Works .....   | 158        |
|  | Acknowledgements .....   | 158        |
|  | References .....   | 159        |
| <b>Chapter 8</b>   | <b>Trajectory Tracking of Quadrotor Aerial Robot Using Improved Dynamic Inversion Method.....</b>                                    | <b>163</b> |
|  | Abstract .....   | 163        |
|  | Introduction.....  | 164        |
|  | Quadrotor Mechanism.....   | 165        |
|  | Dynamic Model of Quadrotor .....   | 166        |
|  | Improved Dynamic Inverse Controller .....  | 169        |
|  | Simulation Results .....   | 172        |
|  | Conclusion .....   | 173        |
|  | References .....   | 175        |
| <b>Section 3 Quadrotors and Small-size Aerial Vehicles</b> |  |            |
| <b>Chapter 9</b>   | <b>Neural Network Control and Wireless Sensor Network-based Localization of Quadrotor UAV Formations .....</b>                       | <b>179</b> |
|  | Introduction .....   | 179        |
|  | Background .....   | 181        |
|  | Leader-Follower Formation Tracking Control .....   | 184        |
|  | Optimized Energy-Delay Sub-Network Routing (OEDSR)<br>Protocol For Uav Localization And Discovery .....                              | 200        |
|  | Simulation Results .....   | 205        |
|  | Conclusions .....  | 207        |
|  | References .....   | 208        |

|  |            |
|--|------------|
| <b>Chapter 10 Dubins Waypoint, Navigation of Small-Class Unmanned Aerial Vehicles.....</b> | <b>209</b> |
| Abstract .....   | 209        |
| Introduction.....  | 210        |
| Method.....  | 212        |
| Results.....   | 220        |
| Conclusion .....   | 223        |
| Acknowledgements .....   | 224        |
| Nomenclature.....  | 224        |
| Conflicts Of Interest.....   | 225        |
| References .....   | 226        |
| <b>Chapter 11 Modelling Oil-Spill Detection with Swarm Drones .....</b>                    | <b>229</b> |
| Abstract .....   | 229        |
| Introduction.....  | 230        |
| Pollutant Dispersion Model.....  | 231        |
| Swarm Design .....   | 234        |
| Experimentation.....   | 237        |
| Macroscopic Model.....   | 244        |
| Discussion .....   | 250        |
| Conflict Of Interests .....  | 252        |
| Acknowledgment.....  | 252        |
| References .....   | 253        |
| <b>Chapter 12 A Survey of Modelling and Identification of Quadrotor Robot.....</b>         | <b>255</b> |
| Abstract .....   | 255        |
| Introduction.....  | 256        |
| Characteristics Of Quadrotor .....   | 258        |
| 6-Dof Airframe Dynamics .....  | 260        |
| Basic Dynamic Model Of A Quadrotor .....   | 263        |
| Aerodynamic Effects .....  | 270        |
| Identification Of A Quadrotor Model .....  | 275        |
| Conclusion And Perspective.....  | 284        |
| Conflict Of Interests .....  | 285        |
| Acknowledgments .....  | 285        |
| References .....   | 286        |

|   |            |
|---|------------|
| <b>Chapter 13 Visual Flight Control of a Quadrotor Using Bioinspired Motion Detector.....</b> | <b>295</b> |
| Abstract .....  | 296        |
| Introduction.....   | 296        |
| Bioinspired Image Processing .....  | 298        |
| Multicamera 3D Pose Estimation .....  | 300        |
| Controller .....  | 302        |
| Experiments And Results .....   | 304        |
| Conclusions And Future Works .....  | 310        |
| Acknowledgments .....   | 310        |
| References .....  | 311        |

#### **Section 4 Applications of UAV-s and Drones**

|  |            |
|--|------------|
| <b>Chapter 14 Development of Rescue Material Transport UAV (Unmanned Aerial Vehicle) .....</b> | <b>315</b> |
| Abstract .....   | 315        |
| Introduction.....  | 316        |
| Development Of Software And Hardware.....  | 317        |
| Research And Development Results .....   | 321        |
| Conclusion .....   | 324        |
| Acknowledgements .....   | 324        |
| Conflicts Of Interest.....   | 324        |
| References .....   | 325        |

|   |            |
|---|------------|
| <b>Chapter 15 Railway Transport Infrastructure Monitoring by UAVs and Satellites ...</b>          | <b>327</b> |
| Abstract .....  | 327        |
| Introduction.....   | 328        |
| Advantages And Disadvantages Of Remotely Piloted Aircrafts.....                                   | 329        |
| Analysis Of Satellite Resources For Obtaining Information.....                                    | 331        |
| The Databases Of Satellite Information .....  | 332        |
| Comparison Of Information Received From Different Sources .....                                   | 333        |
| Reconstruction Of 3D Models Of The Railway Infrastructure<br>Objects From A Set Of 2D Images..... | 336        |
| Conclusion .....  | 339        |
| Acknowledgements .....  | 340        |
| Conflicts Of Interest.....  | 340        |

|   |            |
|---|------------|
| References .....  | 341        |
| <b>Chapter 16 Assessment of UAV Based Vegetation Indices for Nitrogen Concentration Estimation in Spring Wheat .....</b>  | <b>343</b> |
| Abstract .....  | 344        |
| Introduction .....  | 345        |
| Materials And Methods .....   | 348        |
| Results And Discussions.....  | 353        |
| Conclusions.....  | 362        |
| Acknowledgements .....  | 362        |
| Conflicts Of Interest.....  | 362        |
| References .....  | 363        |
| <b>Chapter 17 Research and Teaching Applications of Remote Sensing Integrated with GIS: Examples from the Field .....</b> | <b>369</b> |
| Abstract .....  | 369        |
| Introduction.....   | 370        |
| Wildlife Monitoring On Catalina Island, Pilot 2018 .....  | 372        |
| Autonomous Terrain Modeling.....  | 377        |
| Integrating Remote Sensing And Uas Into Curriculum .....  | 379        |
| Discussion .....  | 382        |
| Conclusion .....  | 383        |
| Acknowledgements .....  | 383        |
| Conflicts Of Interest.....  | 383        |
| References .....  | 384        |
| <b>Chapter 18 Development of Drone Cargo Bay with Real-Time Temperature Control.....</b>                                  | <b>385</b> |
| Abstract .....  | 385        |
| Introduction.....   | 386        |
| Related Works .....   | 387        |
| Configuration Of Temperature Control System .....   | 387        |
| Hardware Design.....  | 388        |
| Software Design.....  | 390        |
| Prototype Design .....  | 392        |
| Conclusion .....  | 394        |

|                            |            |
|----------------------------|------------|
| Acknowledgements .....     | 395        |
| Conflicts of Interest..... | 395        |
| References .....           | 396        |
| <b>Index .....</b>         | <b>399</b> |



# LIST OF CONTRIBUTORS

---

**Paulo Carvalhal**

University of Minho Portugal

**Cristina Santos**

University of Minho Portugal

**Manuel Ferreira**

University of Minho Portugal

**Luís Silva**

University of Minho Portugal

**José Afonso**

University of Minho Portugal

**Andrew Zulu**

Department of Mechanical and Marine Engineering, Polytechnic of Namibia, Windhoek, Namibia

**Samuel John**

Department of Mechanical and Marine Engineering, Polytechnic of Namibia, Windhoek, Namibia

**Larry M. Silverberg**

Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, USA

**Chad Bieber**

Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, USA

**Gordon Ononiwu**

Department of Electrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria

**Ondoma Onojo**

Department of Electrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria

**Oliver Ozioko**

Department of Electrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria

**Onyebuchi Nosiri**

Department of Electrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria

**Antonio Barrientos**

Universidad Politécnica de Madrid – (Robotics and Cybernetics Group), Spain

**Pedro Gutiérrez**

Universidad Politécnica de Madrid – (Robotics and Cybernetics Group), Spain

**Julián Colorado**

Universidad Politécnica de Madrid – (Robotics and Cybernetics Group), Spain

**Jaime del-Cerro**

Universidad Politécnica de Madrid – Robotics and Cybernetics Group, Spain

**Alexander Martínez**

Universidad Politécnica de Madrid – Robotics and Cybernetics Group, Spain

**Ee-May Kan**

Nanyang Technological University, Singapore City, Singapore

**Meng-Hiot Lim**

Nanyang Technological University, Singapore City, Singapore

**Swee-Ping Yeo**

National University of Singapore, Singapore City, Singapore

**Jiun-Sien Ho**

Temasek Polytechnic, Singapore City, Singapore

**Zhenhai Shao**

University of Electronic Science Technology of China, Chengdu, China.

**Lifeng Wang**

Field Bus Technology & Automation Lab, North China University of Technology, Beijing, China

**Yichong He**

Field Bus Technology & Automation Lab, North China University of Technology,  
Beijing, China

**Zhixiang Zhang**

Field Bus Technology & Automation Lab, North China University of Technology,  
Beijing, China

**Congkui He**

Field Bus Technology & Automation Lab, North China University of Technology,  
Beijing, China

**Travis Dierks**

Missouri University of Science and Technology, United States of America

**S. Jagannathan**

Missouri University of Science and Technology, United States of America

**Dahan Xu**

Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, USA

**F. Aznar**

Department of Computer Science and Artificial Intelligence, University of Alicante,  
03090 Alicante, Spain

**M. Sempere**

Department of Computer Science and Artificial Intelligence, University of Alicante,  
03090 Alicante, Spain

**M. Pujol**

Department of Computer Science and Artificial Intelligence, University of Alicante,  
03090 Alicante, Spain

**R. Rizo**

Department of Computer Science and Artificial Intelligence, University of Alicante,  
03090 Alicante, Spain

**M. J. Pujol**

Department of Applied Mathematics, University of Alicante, 03090 Alicante, Spain

**Xiaodong Zhang**

School of Automation, Shenyang Aerospace University, Shenyang 110136, China

**Xiaoli Li**

School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China

**Kang Wang**

School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China

**Yanjun Lu**

School of Automation, Shenyang Aerospace University, Shenyang 110136, China

**Lei Zhang**

Institute of Automatic Control Engineering (LSR), Technische Universität München, 80290 München, Germany

**Tianguang Zhang**

Institute of Automatic Control Engineering (LSR), Technische Universität München, 80290 München, Germany

**Haiyan Wu**

Institute of Automatic Control Engineering (LSR), Technische Universität München, 80290 München, Germany

**Alexander Borst**

Department of Systems and Computational Neurobiology, Max Planck Institute of Neurobiology, Am Klopferspitz 18, D-82152 Martinsried, Germany

**Kolja Kühnlenz**

Institute of Automatic Control Engineering (LSR), Technische Universität München, 80290 München, Germany

Institute for Advanced Study (IAS), Technische Universität München, 80290 München, Germany

**Daeil Jo**

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon, South Korea

**Sergey I. Ivashov**

Remote Sensing Laboratory, Bauman Moscow State Technical University, Moscow, Russia

**Alexander B. Tataraidze**

Remote Sensing Laboratory, Bauman Moscow State Technical University, Moscow, Russia

**Vladimir V. Razevig**

Remote Sensing Laboratory, Bauman Moscow State Technical University, Moscow, Russia

**Eugenia S. Smirnova**

Remote Sensing Laboratory, Bauman Moscow State Technical University, Moscow, Russia

**Olga S. Walsh**

Department of Plant Sciences, Southwest Research and Extension Center, University of Idaho, Moscow, ID, USA.

**Sanaz Shafian**

Department of Plant Sciences, Southwest Research and Extension Center, University of Idaho, Moscow, ID, USA.

**Juliet M. Marshall**

Department of Entomology, Plant Pathology, and Nematology, Idaho Falls Research and Extension Center, University of Idaho, Idaho Falls, ID, USA.

**Chad Jackson**

Department of Entomology, Plant Pathology, and Nematology, Idaho Falls Research and Extension Center, University of Idaho, Idaho Falls, ID, USA.

**Jordan R. McClintick-Chess**

Department of Plant Sciences, Southwest Research and Extension Center, University of Idaho, Moscow, ID, USA.

**Steven M. Blanset**

BASF-Chemical Co., Caldwell, ID, USA.

**Kristin Swoboda**

Take Flight UAS, LLC, Boise, ID, USA.

**Craig Thompson**

Take Flight UAS, LLC, Boise, ID, USA.

**Kelli M. Belmont**

Seminis, Payette, ID, USA.

**Willow L. Walsh**

Vallivue High School, Caldwell, ID, USA.

**Laura C. Loyola**

Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.

**Jason T. Knowles**

Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.  
GeoAcuity, Los Angeles, CA, USA.

**Andrew J. Marx**

Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.

**Ryan McAlinden**

Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.  
Institute for Creative Technologies, University of Southern California, Los Angeles,  
CA, USA.

**Steven D. Fleming**

Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.

**Sedam Lee**

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon,  
South Korea

**Yongjin Kwon**

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon,  
South Korea

# LIST OF ABBREVIATIONS

---

|       |  |
|-------|--|
| AP    | Access Point                                 |
| AVCL  | Aerial Vehicle Control Language              |
| AOA   | Angle of Attack                              |
| AED   | Automatic Cardioverter Defibrillator         |
| AUAS  | Autonomous Unmanned Aerial Systems           |
| BET   | Blade Element Theory                         |
| CRP   | Calibrated Reflectance Panel                 |
| CIC   | Catalina Island Conservancy                  |
| COA   | Civil Certificate of Waiver Authorization    |
| CW    | Clockwise                                    |
| CHs   | Cluster Heads                                |
| COTS  | Commercial off the Shelf                     |
| CASI  | Compact Airborne Spectrographic Imager       |
| DLS   | Downwelling Light Sensor                     |
| DWN   | Dubins Waypoint Navigation                   |
| ERS   | Earth Remote Sensing                         |
| ESC   | Electronic Speed Controllers                 |
| EMD   | Elementary Motion Detector                   |
| ERR   | Experimental Railway Ring                    |
| FAA   | Federal Aviation Administration              |
| FSM   | Finite State Machine                         |
| FL    | Formation Leader                             |
| GIST  | Geographic Information Sciences & Technology |
| GIS   | Geographic Information System                |
| GPS   | Global Positioning System receivers          |
| GUI   | Graphical User Interface                     |
| GNDVI | Green Normalized Difference Vegetation Index |
| GTS   | Guaranteed Time Slot                         |

|       |  |
|-------|--|
| IMU   | Inertial Measurement Units             |
| ICT   | Institute for Creative Technologies    |
| ICE   | Internet Communications Engine         |
| IFT   | Iterative Feedback Tuning              |
| ILC   | Iterative Learning Control             |
| LQR   | Linear Quadratic Regulator             |
| MM    | Master Node                            |
| MAVs  | Microaerial Vehicles                   |
| MP    | Mission Planner                        |
| MFAC  | Model Free Adaptive Control            |
| MT    | Momentum Theory                        |
| NN    | Neural Networks                        |
| NDVI  | Normalized Difference Vegetation Index |
| OFN   | Obstacle Field Navigation              |
| OWT   | One World Terrain                      |
| PLR   | Packet Loss Ratio                      |
| PLS   | Partial Least Squares                  |
| PCA   | Principal Component Analysis           |
| PID   | Proportional Integral Derivative       |
| PWM   | Pulse Width Modulation                 |
| RCS   | Radar Cross Section                    |
| RN    | Relay Nodes                            |
| RPL   | Remote License Pilots                  |
| RMSE  | Root Mean Squared Error                |
| SMC   | Sliding Mode Control                   |
| SSI   | Spatial Sciences Institute             |
| TIFF  | Tag Image File Format                  |
| UAS   | Unmanned Aerial Systems                |
| VI    | Vegetation Indices                     |
| VTOL  | Vertical Take-Off And Landing          |
| VM    | Virtual Machine                        |
| VR/AR | Virtual Reality/Augmented Reality      |
| VRFT  | Virtual Reference Feedback Tuning      |
| WN    | Waypoint Navigation                    |

# PREFACE

---

Drones are defined as unmanned aerial vehicles (UAVs) that can fly over space from a few tens of meters - up to thousands of kilometers, or small (micro and nano) quadrotors that can move indoors. Today – there is a rising need of these types of flying vehicles - for civil and for military applications. There is also significant interest for development of new types of drones that can fly autonomously – around different locations and environments, and will accomplish different technical missions (agriculture, public events – sports/ concerts, space research etc.).

Within the past decade – wide spectra of applications have emerged that resulted in many variations in the size and the design of the UAV-s (drones). The year of 2015 was declared as a year that exponentially boosted the drone applications in many human-oriented fields and environments, especially in agriculture and forestry (75% of all applications).

The drones categories differ among themselves by their configuration, platform or application. There are 3 major classes defined in the literature, and the class I has 4 subclasses (a, b, c, d):

Class I-a) – Nano drones, weight < 200 g.

Class I-b) – Micro drones, weight between 200g – 2 kg.

Class I-c) – Mini drones, weight between 2 kg – 20 kg.

Class I-d) – Small drones, weight between 20 kg – 150 kg.

Class II – Tactical drones, weight between 150-600 kg.

Class III – Strike/ Hale drones, weight over 600 kg.

This edition covers different topics from UAV-s and drones – such as design and development of UAV-s, trajectory control techniques, small drones and quadrotors, and application scenarios of UAV-s and drones.

Section 1 focuses on design and development of UAV-s, describing design and development of a fly-by-wireless UAV platform, review of control algorithms for autonomous quadrotors, central command architecture for

high-order autonomous unmanned aerial systems, and quadcopter design for payload delivery.

Section 2 focuses on trajectory control techniques, describing development of rescue material transport UAV, railway transport infrastructure monitoring by UAVs and satellites, assessment of UAV based vegetation indices for nitrogen concentration estimation in spring wheat, research and teaching applications of remote sensing integrated with GIS: examples from the field, and development of drone cargo bay with real-time temperature control.

Section 3 focuses on small drones and quadrotors, describing neural network control and wireless sensor network-based localization of quadrotor UAV formations, Dubin's waypoint, navigation of small-class unmanned aerial vehicles, modelling oil-spill detection with swarm drones, a survey of modelling and identification of quadrotor robot, and visual flight control of a quadrotor using bioinspired motion detector.

Section 4 focuses on application scenarios of UAV-s and drones, describing .

# **SECTION 1**

## **DESIGN AND**

## **DEVELOPMENT OF UAV-S**



# **CHAPTER**

# **1**

## **Design and Development of a Fly-by-Wireless UAV Platform**

---

**Paulo Carvalhal, Cristina Santos, Manuel Ferreira, Luís Silva and José Afonso**

University of Minho Portugal

### **INTRODUCTION**

The development of unmanned aerial vehicles (UAVs) has become an active area of research in recent years, and very interesting devices have been developed. UAVs are important instruments for numerous applications, such as forest surveillance and fire detection, coastal and economic exclusive zone surveillance, detection of watershed pollution and military missions.

---

**Citation:** Paulo Carvalhal, Cristina Santos, Manuel Ferreira, Luis Silva and Jose Afonso (January 1st 2009). “Design and Development of a Fly-by-Wireless UAV Platform”, Aerial Vehicles, Thanh Mung Lam, IntechOpen, DOI: 10.5772/6464.

**Copyright:** © 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License

The work described in this chapter is part of a larger project, named AIVA, which involves the design and development of an aerial platform, as well as the instrumentation, communications, flight control and artificial vision systems, in order to provide autonomous takeoff, flight mission and landing maneuvers. The focus of the chapter is on one of the main innovative aspects of the project: the onboard wireless distributed data acquisition and control system. Traditionally, UAVs present an architecture consisting of one centralized and complex unit, with one or more CPUs, to which the instrumentation devices are connected by wires. At the same time, they have bulky mechanical connections. In the approach presented here, dubbed “fly-by-wireless”, the traditional monolithic processing unit is replaced by several less complex units (wireless nodes), spread out over the aircraft. In that way, the nodes are placed near the sensors and controlled surfaces, creating a network of nodes with the capacity of data acquisition, processing and actuation.

This proposed fly-by-wireless platform provides several advantages over conventional systems, such as higher flexibility and modularity, as well as easier installation procedures, due to the elimination of connecting cables. However, it also introduces several challenges. The wireless network that supports the onboard distributed data acquisition and control system needs to satisfy demanding requirements in terms of quality of service (QoS), such as sustainable throughput, bounded delay and reliable packet delivery. At the same time, it is necessary to guarantee that the power consumption of the battery powered wireless nodes is small, in order to increase the autonomy of the system. Currently there are many different wireless network technologies available in the market. Section 2 presents an overview of the most relevant technologies and discusses their suitability to meet the above requirements. Based on this analysis, we chose the Bluetooth wireless network technology as the basis for the design and development of a prototype of the fly-by-wireless system. The system was implemented using commercial off-the-shelf components, in order to provide a good trade-off between development costs, reliability and performance. Some other objectives were also pursued in the development of the system, namely the design of a framework where communication between nodes is effective and independent of the technology adopted, the development of a design approach to model the embedded system and the development of an application oriented operating system with a modular structure.

The following sections are organized as follows: section 2 presents an overview of available wireless network technologies, taking into account

the requirements of the application; section 3 presents the global electronics architecture of the UAV platform, while section 4 describes the developed onboard wireless system; section 5 presents experimental performance results obtained with this system, and section 6 presents the conclusions and addresses the future work.

## WIRELESS NETWORK TECHNOLOGIES

Most wireless networks technologies available nowadays can be subdivided in a few categories: satellite networks, mobile cellular networks, broadband wireless access, wireless local networks (WLAN) and wireless personal networks (WPAN). The former three differ substantially from the latter two. One difference is that the network infrastructure does not belong to the user, but to the network operator, which charges the user for the services provided. Other difference is that they provide coverage over a large area. On the other hand, WLAN and WPAN are short range technologies in which all the communications equipment usually belongs to the user. These characteristics are more adequate for the intended application, so the remainder of this section will focus on wireless network technologies belonging to these two categories.

The most widespread type of WLAN nowadays is the IEEE 802.11 (IEEE, 2007), also known as WiFi. These networks are available on multiple physical options and operating frequency bands. However, all these versions use the same MAC (Medium Access Control) protocol; a contention based CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) mechanism known as DCF (Distributed Control Function). Given its statistical nature, this protocol is not adequate to provide the QoS guarantees required by the onboard wireless data acquisition and control system due to the probability of collisions.

In order to support real-time traffic, the 802.11 standard defines an alternative MAC protocol known as PCF (Point Coordination Function), based on a polling scheme, which is capable of providing QoS guarantees. However, unlike the DCF protocol, the implementation of PCF is not mandatory, and the availability of products that support it is scarce. More recently, a newer standard, the IEEE 802.11e (IEEE, 2007), designed to improve the efficiency and QoS support of 802.11 networks was released, but its availability on the market is also low.

Concurrently to the development of the 802.11, the European Telecommunications Standards Institute (ETSI) has developed another

WLAN standard: HIPERLAN/2 (ETSI, 2002). HIPERLAN/2 networks are designed to operate at the 5 GHz band using OFDM (Orthogonal Frequency Division Modulation). Its physical layer is similar to the one used by the IEEE 802.11a due to agreements made by the two standard bodies. On the other hand, the MAC protocols used by these networks are radically different. HIPERLAN/2 uses a demand based dynamic TDMA (Time Division Multiple Access) protocol, which is able to provide extensive support of QoS to multiple types of traffic, including those generated by data acquisition and control systems (Afonso & Neves, 2005). However, the 802.11 standard won the battle for the wireless LAN market and as such no available HIPERLAN/2 products are known at the moment.

Due to its design characteristics, 802.11 and HIPERLAN/2 modules present relatively high power consumption. Although these networks can be suitable to interconnect devices like computers, there is an enormous potential market to provide wireless communication capabilities to smaller and cheaper devices running on batteries without the need of frequent recharging. Such devices include computer peripherals, biomedical monitoring appliances, surveillance units and many other sensing and actuation devices. To provide communication capabilities to such devices, various low cost short range networks, known collectively by the term wireless personal area network (WPAN), are being developed.

At the IEEE, the task of standardization of WPAN networks is under the scope of the IEEE 802.15 group. One of these standards, the IEEE 802.15.4 (IEEE, 2006) defines the physical and MAC layer of ZigBee (ZigBee, 2006), which aims to provide low power and low bit rate WPANs with the main purpose of enabling wireless sensor network applications. At the physical layer, the IEEE 802.15.4 relies on direct sequence spread spectrum (DSSS) to enhance the robustness against interference, and provides gross data rates of 20/40 kbps, at the 868/915 band, and 250 kbps, at the 2.4 GHz band. As in 802.11 networks, the basic ZigBee MAC protocol is a contention based CSMA/CA mechanism. A complementary mechanism defined in the 802.15.4 standard, the guaranteed time slot (GTS), enables the provision of some QoS guarantees to real-time traffic.

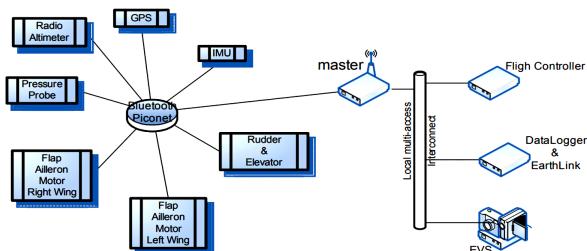
Bluetooth (Bluetooth, 2003) is another WPAN technology. It operates in the 2.4 GHz band using frequency hopping spread spectrum (FHSS) and provides a gross data rate of 1 Mbps. Bluetooth operates using a star topology, called piconet, formed by one master and up to seven active slaves. Transmissions can achieve a range of 10 or 100 m, depending of the class

of the device. At the MAC layer, the Bluetooth devices uses a polling based protocol that provides support for both real-time and asynchronous traffic.

Bluetooth provides better overall characteristics than the other networks discussed here for the desired application. It drains much less power than 802.11 and HIPERLAN/2, uses a MAC protocol that provides support for real-time traffic, and provides a higher gross data rate than ZigBee. Bluetooth spread spectrum covers a bandwidth of 79 MHz while ZigBee operates in a band of less than 5 MHz, what makes the former more robust against interference. Moreover, Bluetooth provides an adaptive frequency hopping mechanism that avoids frequency bands affected by interference. Given these characteristics and the availability of the technology at the time of development, Bluetooth was chosen as the supporting wireless network technology for the development of the prototype of the system described in the following sections.

## GLOBAL ELECTRONICS ARCHITECTURE

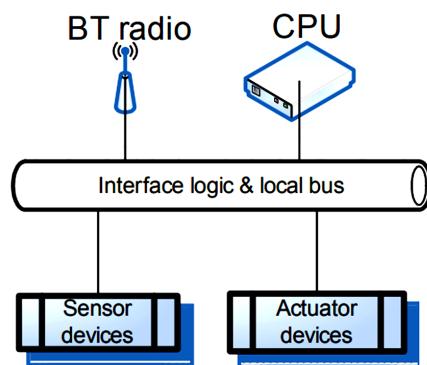
The global view of architectural model of the onboard computing and communication system of the AIVA fly-by-wireless UAV platform is presented in Figure 1. It is a multitasking/multiprocessor based system connected by an asynchronous local bus that allows for speed adaptation of different tasks/processors. The system architecture supports one processing unit for a Bluetooth piconet master node, one flight controller unit, one data logger and earth link, and one embedded vision system (EVS). In each of these nodes many critical processes are permanently running.



**Figure 1.** Global electronics architecture of the AIVA UAV platform This architecture allows an easy way to introduce or remove processing units from the platform. For instance new sensors or new vision units can be included. In the first case a new module must be connected to the Bluetooth piconet, and in the second case the new module is connected to the local multi-access bus.

## ONBOARD WIRELESS SYSTEM

The AIVA UAV platform implements an onboard wireless distributed data acquisition and control system based on Bluetooth (BT) wireless network technology, represented by the Bluetooth piconet of Figure 1. The general architecture of a wireless node is presented on Figure 2. Each node is composed by a commercial off-the-shelf Bluetooth module that contains the radio electronics, a microcontroller that runs the code that controls the behavior of the node, and a local bus that provides interfacing between the node components, as well as specific sensors and/or actuators according to the purpose of the node.



**Figure 2.** Architecture of a Bluetooth wireless node

## Physical Architecture

The physical part of the platform is built around a low power Texas Instruments MSP430 microcontroller, a Von-Neumann 16 bit RISC architecture with mixed program, data and

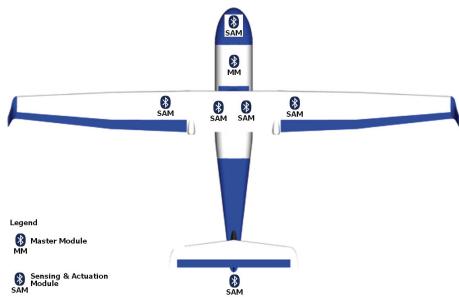
I/O in a 64Kbytes address space. Besides its low power profile, which uses about  $280 \mu\text{A}$  when operating at  $1 \text{ MHz} @ 2.2 \text{ Vdc}$ , MSP430 offers some interesting features, like single cycle register operations, direct memory-to-memory transfers and a CPU independent hardware multiplication unit. From the flexibility perspective, a flexible I/O structure capable of independently dealing with different I/O bits, in terms of data direction, interrupt programming, and edge triggering selection; two USARTs supporting SPI or UART protocols; an onboard 12 bit SAR ADC with 200 kHz rate; and PWM capable timers, are all relevant features.

The Bluetooth modules chosen for the implementation of the wireless nodes are OEM serial adapter devices manufactured by connectBlue. The master node uses an OEMSPA33i module and the slave nodes use OEMSPA13i modules (connectBlue, 2003). These modules include integrated antennas; nevertheless, we plan to replace them with modules with external antennas in future versions of the platform, to be able to shield the modules in order to increase the reliability of the system against electromagnetic interference.

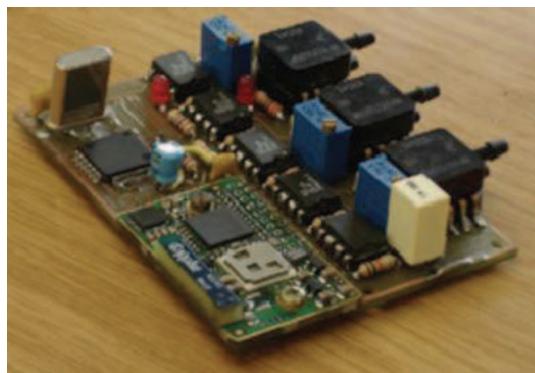
While the module used on the master (OEMSPA33i) allows up to seven simultaneous connections, the module used on the slaves (OEMSPA13i) has a limitation of only three simultaneous connections. However, this limitation does not represent a constraint to the system because the slaves only need to establish one connection (to the master).

The connectBlue modules implement a virtual machine (VM) that enables the provision of a serial interface abstraction to the microcontroller, so Bluetooth stack details can be ignored and focus can be directed to the application. The manufacturer's virtual machine implements a wireless multidrop access scheme where the master receives all frames sent by the slaves and all slaves can listen to the frames sent by the master, in a point-to-multipoint topology. The AIVA onboard wireless system is composed by one Bluetooth piconet containing seven nodes: one master (MM - Master Module) and six slaves (SAM - Sensing & Actuation Modules). The nodes are spread over the aircraft structure, as shown in Figure 3. The master node (MM) is placed at the fuselage body, and acts as the network and flight controller, onboard data logger, and communications controller for the link with the ground station. On each wing, there is a SAM node for an electrical propulsion motor and for control surfaces (ailerons and flaps). These wing nodes are responsible for motor speed control and operating temperature monitoring, as well as control surfaces actuation and position feedback.

In the fuselage body, there are other two SAM nodes, one for a GPS module and other for an inertial measurement unit (IMU), which provide information assessment for navigational purposes. At the tail, there is another SAM node for elevator and rudder control, and position feedback. Finally, at the nose there is a SAM node connected to a nose probe consisting of a proprietary design based on six independent pressure sensors that give valuable digital information for flight control. This node also contains an ultrasonic probe that provides information for support of the automatic take-off and landing system. Figure 4 displays the physical layout of the nose node. The Bluetooth module is in the lower corner, the microcontroller is on the left hand side and the sensor hardware on the right hand side of the board.



**Figure 3.** Node distribution on the aircraft structure

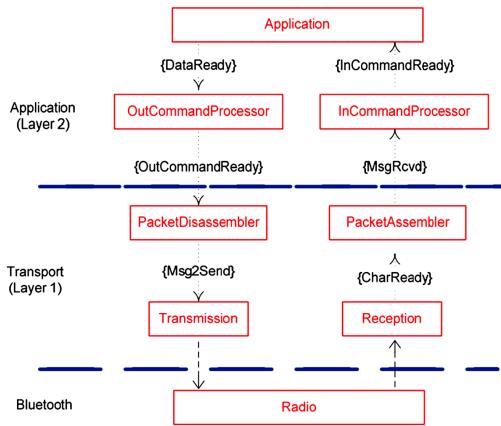


**Figure 4.** Physical layout of the nose node

## Logical Architecture

The logical architecture of the developed system is a two layered state machine implementation, composed by a transport layer and an application layer. The transport layer provides a packet delivery service, under control of master node, capable of transparent delivery of data packets across the network.

The transport layer is application independent, and interfaces with the top level application layer by means of a data space for buffering and a set of signaling control bits that allow total synchronization between the two layers. The hierarchy and signaling between the two layers is represented in Figure 5.



**Figure 5.** Hierarchy and signaling at the logical level of the platform.

The asynchronous reception process delivers characters to upper processes. Analyzing the hierarchy from the lower level to the upper level, CharReady condition goes TRUE every time a new character arrives to the interface. The next process in the hierarchy is PacketAssembler, a state machine that performs packet re-assembly, reconstructing the original packet from a group of segments, and delivers packets for the next process in the hierarchy. When MsgRcvd (message received) goes TRUE, a new message is ready for processing. Thus, for incoming data, the model at layer 1 receives characters and delivers ready-to-process messages to the application layer. When the application layer understands that the message is ready to process, a command processor for incoming messages is activated in order to decode the embedded command and semantics contained in the message, to eventually execute some action, and to pass relevant information for the final application.

For outgoing data, the resident application eventually makes available some data to transmit to the master, signaling this event with a DataReady signal. This causes the output command processor to execute its cycle, preparing one message to be sent. When the message is ready, OutCommandReady goes TRUE, signaling to the lower layer that there is a message to send to the network. At this phase, frame segmentation starts (if needed) by means of a state machine for packet disassembly. This state machine breaks the original message in smaller segments prepared to be serialized. Each time a segment is ready to be sent, Msg2Send goes TRUE and serialization is triggered. So, for outgoing data, the transport layer

receives messages from application layer, and sends segments to the radio module in order to be sent over the wireless medium.

Layer 2 is application dependent, and has no knowledge of the lower layer internal mechanisms, just the services and interfaces available from it. That means that its logical architecture can be used in other applications. For the fly-by-wireless application, its main goal is to replicate a system table among all network nodes, at the maximum possible frequency. This system table maintains all critical system values, describing the several sensors and actuators, status parameters, and loop feedback information. Each network node is mapped to a table's section, where all related variables from sensing, actuators and metering are located. This layer is responsible for cyclic refreshing the respective table contents, based on local status, and also for cyclic actuation according to data sent from the master node (flight controller orders). This way, the whole system is viewed as a resident two-dimensional array located at master, with different partial copies distributed and synchronized among the slave nodes.

## Other Design Issues

All the Bluetooth modules in the developed platform are configured in non discoverable mode, which contributes to the security of the system. The node discovery process of Bluetooth is a slow process, in the order of seconds, however it is not a problem since the master stores the addresses of all slave nodes that should participate in the piconet, so this process is avoided. The piconet formation is performed on the ground before the takeoff procedure, so the associated delay does not constitute a problem as well.

The use of Bluetooth technology limits the piconet operation to a maximum of seven active slaves; however, this limitation is not of major concern on the developed system, since only six slaves are used, and could only impose some restraints if node number should be raised. The number of slaves in the network could be increased by interconnecting a number of piconets to form a scatternet. That way, a device participating in two piconets could relay traffic between both piconets. However, this architecture would probably have a negative impact in the performance of the network, making it more difficult to provide QoS guarantees to the application. Moreover, currently there are very few actual implementations of scatternets available.

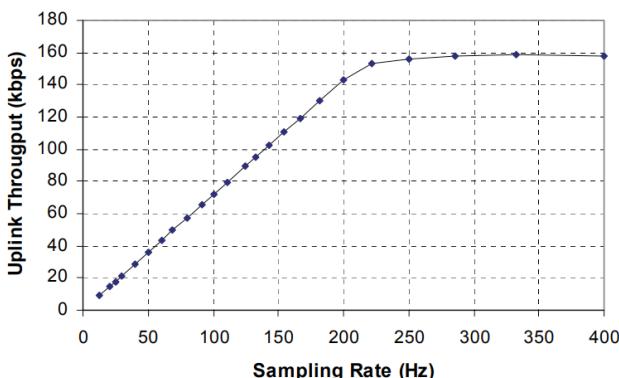
Given that free space propagation loss is proportional to the square of the distance, it is not expected that the onboard wireless network will either suffer or induce interference on other networks operating in the same frequency band, such as the widely deployed WiFi networks, since the

former operates in the sky most of time, while the later are normally based on the ground.

## EXPERIMENTAL RESULTS

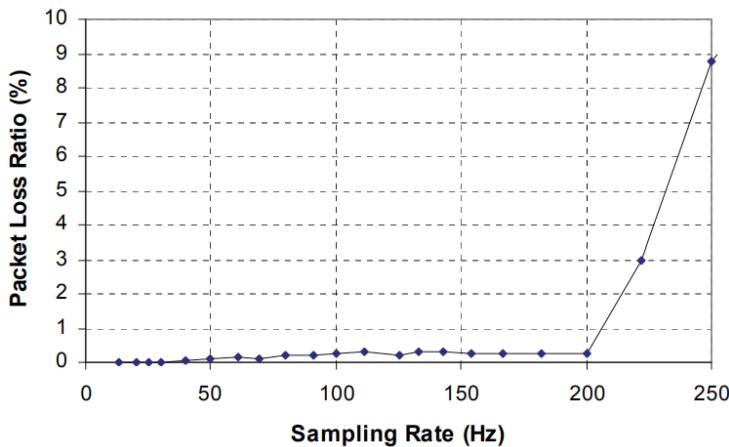
The performance of the developed wireless system was evaluated in laboratory. The experimental setup used to achieve the results presented in this section is composed by 6 slaves sending data periodically to the master (uplink direction) at the same predefined sampling rate. Each sampling packet has a length of 15 octets, which is the maximum packet length supported by the transport layer due to a limitation imposed by the virtual machine used by the Bluetooth module.

Figure 6 presents the aggregated uplink throughput that reaches the master node as a function of the sampling rate used by the 6 slaves. Since Bluetooth uses a contention-free MAC protocol, the uplink transmissions are not affected by collisions, so the network throughput increases linearly with the offered load until the point it reaches saturation, which in this scenario corresponds to the situation where the slaves transmit data at sampling rates higher than 200 Hz. As this figure shows, the maximum throughput available to the application is about 160 kbps, which is significantly lower than the gross data rate provided by Bluetooth (1 Mbps). This difference can be explained by the overhead introduced by the Bluetooth protocol and the virtual machine, including the gap between the packets, the FEC (Forward Error Correction) and ARQ (Automatic Repeat reQuest) mechanisms, the packet headers, as well as the overhead introduced by control packets such as the POLL packet, that is sent by the master to grant permission to slaves to transmit.

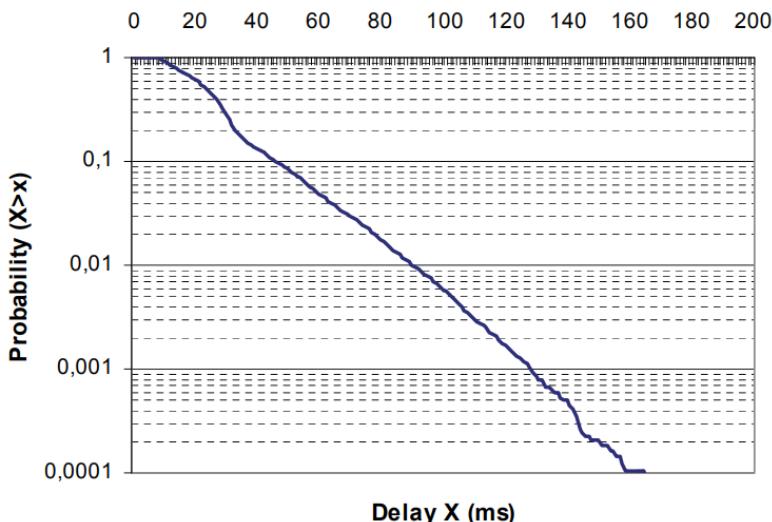


**Figure 6.** Uplink throughput as a function of the sampling rate

Figure 7 presents the packet loss ratio (PLR) averaged over the 6 slaves as a function of the sampling rate. As the figure shows, the PLR is limited to less than 0.5 % in the region where the network is not congested, but increases rapidly after the saturation point. The flight control application should be able to tolerate such small losses; otherwise a change in the supporting wireless technology should be made in the attempt to obtain higher link reliability.



**Figure 7.** Packet loss ratio as a function of the sampling rate



**Figure 8.** Complementary cumulative distribution of the delay

Concerning to the delay experienced by the packets as they travel from the slaves to the master, results showed that the delay is not adversely affected by the rise in the offered load, as long as the network operates below the saturation point. For sampling rates up to 200 Hz, the registered average delay was 27 ms and the standard deviation was 16 ms.

Figure 8 presents the complementary cumulative distribution ( $P\{X>x\}$ ) for the random variable  $X$ , which represents the delay for all the samples collected using sampling rates in the range from 0 to 200 Hz. With this chart, it is possible to see the probability that the delay exceeds a given delay bound, which is an important metric for real-time applications such as the one considered in this chapter. The chart shows, for instance, that less than 1 % of the sample packets suffer a delay higher than 90 ms, while less than 0.1 % of the packets suffer a delay higher than 120 ms.

Experimental tests were also made with a varying number of slaves in the piconet (from 1 to 6), both in the uplink and downlink direction. The average delay measured in the downlink direction (from the master to the slaves) was slightly higher than the one registered in the uplink direction, but below 40 ms, for the measurements made with up to 4 slaves. However, the average master-to-slave delay with 5 slaves in the network ascended to 600 ms, while with 6 slaves the performance was even worse, with the average delay reaching 1000 ms.

## CONCLUSION

This chapter presented the design and development of a fly-by-wireless UAV platform built on top of Bluetooth wireless technology. The developed onboard wireless system is composed by one master node, connected to the flight controller and six slave nodes spread along the aircraft structure and connected to several sensors and actuators.

In order to assess the suitability of the developed system, several performance evaluation tests were carried out. The experimental results showed that, for the slave-to-master direction, the system prototype is able to support a sampling rate of up to 200 Hz for each of the 6 slaves simultaneously without significant performance degradation in terms of throughput, loss or delay. On the other hand, although the master-to-slave delay with 1 to 4 slaves in the network is low, its value increases significantly with 5 and 6 slaves, which is unacceptable given the real-time requirements of the desired application.

This problem is caused by implementation issues related to the proprietary embedded virtual machine provided by the manufacturer of the Bluetooth module that is used in the master node of the prototype.

The approach of relying on the virtual machine provided by the manufacturer, which hides the Bluetooth protocol stack functionality, allowed the development focus to be directed to the application, reducing the development costs. The disadvantage, however, is the lack of control of the behavior of the system at the Bluetooth stack level, which impedes the optimization of the performance of the system at this level and the correction of problems such as the verified with the master-to-slave delay. The solution to the detected problem can pass either by the replacement of the Bluetooth module by a newer version (already available) from the same manufacturer or by the direct interaction with the Bluetooth stack, with the bypass of the virtual machine.

Despite the limitations of the current prototype, the overall results provided by the experimental tests are satisfactory. Nevertheless, further tests are needed in order to evaluate the behavior of the system under more harsh interference conditions, as well as in a real scenario onboard the aircraft.

## REFERENCES

1. Afonso, J. A. & Neves, J. E. (2005), Fast Retransmission of Real-Time Traffic in HIPERLAN/2 Systems, *Proceedings of Advanced Industrial Conference on Telecommunications (AICT2005)*, pp. 34-38, ISBN 0-7695-2388-9, Lisbon, Portugal, July 2005, IEEE Computer Society.
2. Bluetooth SIG (2003), Specification of the Bluetooth system. Available at <http://www.bluetooth.org/>.
3. connectBlue (2003), Serial Port Adapter – 2nd Generation, *User Manual*. Available at <http://www.connectblue.com/>.
4. ETSI TR 101 683 V1.1.1 (2000), Broadband Radio Access Networks (BRAN)—HIPERLAN Type 2—Data Link Control (DLC) Layer—Part 1: Basic Data Transport Functions.
5. IEEE Std 802.11 (2007), IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
6. IEEE Std 802.15.4(2006), IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).
7. ZigBee Standards Organization (2006), ZigBee Specification. Available at <http://www.zigbee.org/>.



# **CHAPTER**

# **2**

## **A Review of Control Algorithms for Autonomous Quadrotors**

---

**Andrew Zulu, Samuel John**

Department of Mechanical and Marine Engineering, Polytechnic of Namibia,  
Windhoek, Namibia

### **ABSTRACT**

The quadrotor unmanned aerial vehicle is a great platform for control systems research as its nonlinear nature and under-actuated configuration make it ideal to synthesize and analyze control algorithms. After a brief explanation of the system, several algorithms have been analyzed including their advantages and disadvantages: PID, Linear Quadratic Regulator

---

**Citation:** Zulu, A. and John, S. (2014), "A Review of Control Algorithms for Autonomous Quadrotors". *Open Journal of Applied Sciences*, **4**, 547-556. doi: 10.4236/ojapps.2014.414053.

**Copyright:** © 2014 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

(LQR), Sliding mode, Backstepping, Feedback linearization, Adaptive, Robust, Optimal,  $L_1$ ,  $H_\infty$ , Fuzzy logic and Artificial neural networks. The conclusion of this work is a proposal of hybrid systems to be considered as they combine advantages from more than one control philosophy.

**Keywords:** Control, Algorithms, Quadrotors, Intelligent Control, Optimal Control, Robust Control, Adaptive Control, Linear Control, Nonlinear Control

## INTRODUCTION

The quadrotor unmanned aerial vehicle (UAV) are helicopters with four rotors typically designed in a cross configuration with two pairs of opposite rotors rotating clockwise and the other rotor pair rotating counter-clockwise to balance the torque. The roll, pitch, yaw and up-thrust actions are controlled by changing the thrusts of the rotors using pulse width modulation (PWM) to give the desired output.

Civilian use of unmanned/micro aerial vehicles (UAVs/MAVs) include aerial photography for mapping and news coverage, inspection of power lines, atmospheric analysis for weather forecasts, traffic monitoring in urban areas, crop monitoring and spraying, border patrol, surveillance for illegal imports and exports, fire detection and control, search and rescue operations for missing persons and natural disasters.

In this work, the focus is to realize the best configuration and control scheme of the quadrotor MAV for application in game counting in the protected game reserves in Africa. The current situation in most African countries is that researchers and game rangers either count from road vehicles by driving by the road or use conventional helicopters. The former is ineffective as the count is limited to only animals coming close to the road whereas the later leads to inaccurate counts as the animals run for safety when a helicopter hovers above them. Several advantages accrue to the quadrotor which has led to much attention from researchers. This is mainly due to its simple design, high agility and maneuverability, relatively better payload, vertical take-off and landing (VTOL) ability.

The quadrotor does not have complex mechanical control linkages due to the fact that it relies on fixed pitch rotors and uses the variation in motor speed for vehicle control [1]. However, these advantages come at a price as controlling a quadrotor is not easy because of the coupled dynamics and its commonly under-actuated design configuration [2]. In addition, the

dynamics of the quadrotor are highly non-linear and several uncertainties are encountered during its missions [3], thereby making its flight control a challenging venture. This has led to several control algorithms proposed in the literature. In this work, a review of the prominent controllers applied to the quadrotor is reviewed.

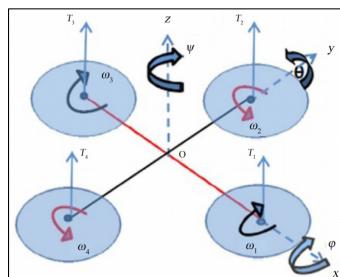
The rest of this paper is organized as follows: Section 2 provides a brief description of the mathematical model of the quadrotor; Section 3 gives a review of the popular controllers proposed for the quadrotor and a discussion of hybrid control systems; in Section 4 a tabular comparison is presented and the conclusions are given in Section 5.

## MATHEMATICAL MODEL

In this section a brief description of the main mathematical equation for the quadrotor is explained.

In flight mode, the quadrotor is able to generate its lift forces by controlling the fixed pitch angle rotor speeds. Figure 1 presents the schematic diagram. The center of mass is the origin O of the coordinate system (see Figure 1) and the forward direction is arbitrarily fixed in the x-axis. To lift off from ground, all four rotors are rotated at the same speed in the sense shown. A total thrust equal to the weight of the system stabilizes the system in hover. To roll about the x-axis, differential thrust of rotors 2 and 4 is applied. To pitch about the y-axis, differential thrust of rotors 1 and 3 is applied. The yaw motion is achieved by the differential thrusts of the opposite rotors (1/3 or 2/4) while also adjusting the constant thrusts of the remaining opposite pairs to maintain altitude.

Using the Newton-Euler formalism, the equations of motion of the quadrotor are given in [4]:



**Figure 1.** Schematic of quadrotor showing rotor configurations.

$$\begin{aligned}\ddot{\phi} &= \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J_r}{I_x}\dot{\phi}\Omega + \frac{l}{I_x}U_2, \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) - \frac{J_r}{I_y}\dot{\theta}\Omega + \frac{l}{I_y}U_3, \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}U_4, \\ \ddot{z} &= -g + (\cos\phi\cos\theta)U_1/m, \\ \ddot{x} &= (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)U_1/m, \\ \ddot{y} &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)U_1/m.\end{aligned}$$

where  $\Phi$ ,  $\theta$  and  $\psi$  are the roll, pitch and yaw angles respectively;  $I_x$ ,  $I_y$ ,  $I_z$  are the mass moments of inertia in the x, y, and z axes respectively;  $J_r$  is the rotor inertia;  $\Omega$  is the angular velocity of the rotor; l is the length of the rotor arm from the origin of the coordinate system; and the inputs to the four rotors are given by the expressions:

$$\begin{aligned}U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \\ U_2 &= b(-\Omega_2^2 + \Omega_4^2), \\ U_3 &= b(\Omega_1^2 - \Omega_3^2), \\ U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2).\end{aligned}$$

where b and d are thrust and drag coefficients respectively. For detailed derivation of the equations, consult [4].

## SURVEY OF CONTROL ALGORITHMS

Due to the nature of the dynamics of the quadrotor, several control algorithms have been applied to it. As to be expected, each control scheme has its advantages and disadvantages. The control schemes used could be broadly categorized as linear and non-linear control schemes. In this review a broad range of controllers within these categories are discussed.

### Proportional Integral Derivative (PID)

The PID controller has been applied to a broad range of controller applications. It is indeed the most applied controller in industry [5]. The classical PID linear controller has the advantage that parameter gains are easy to adjust, is simple to design and has good robustness. However some of

the major challenges with the quadrotor include the non-linearity associated with the mathematical model and the imprecise nature of the model due to unmodeled or inaccurate mathematical modeling of some of the dynamics. Therefore applying PID controller to the quadrotor limits its performance.

A PID controller was used for the attitude control of a quadrotor in [6], while a dynamic surface control (DSC) was used for the altitude control. Applying Lyapunov stability criteria, Lee et al. were able to prove that all signals of the quadrotor were uniformly ultimately bounded. This meant that the quadrotor was stable for hovering conditions. From the simulation and experimental plots however, it reveals the PID controller to have performed better in the pitch angle tracking, whereas large steady state errors could be observed in the roll angle tracking.

In another work by Li and Li [7], a PID controller was applied to regulate both position and orientation of a quadrotor. The PID parameter gains were chosen intuitively. The performance of the PID controller indicated relatively good attitude stabilization. The response time was good, with almost zero steady state error and with a slight overshoot.

It is generally established in the literature that the PID controller has been successfully applied to the quadrotor though with some limitations. The tuning of the PID controller could pose some challenges as this must be conducted around the equilibrium point, which is the hover point, to give good performance.

Figure 2 shows the general block diagram of a PID controller for the quadrotor.

## Linear Quadratic Regulator/Gaussian-LQR/G

The LQR optimal control algorithm operates a dynamic system by minimizing a suitable cost function. Boubdallar and co-researchers applied the LQR algorithm to a quadrotor and compared its performance to that of the PID controller in [8]. However, the PID is applied on the quadrotor simplified dynamics and the LQR on the complete model. Both approaches provided average results but it implicitly was clear that the LQR approach had better performance considering the fact that it was applied to a more complete dynamic model.

A simple path-following LQR controller was applied by Cowling et al. in [9] on a full dynamic model of the quadrotor. It was shown that accurate path following was achieved in simulation using optimal real-time trajectories despite the presence of wind and other disturbances. The controller seemed

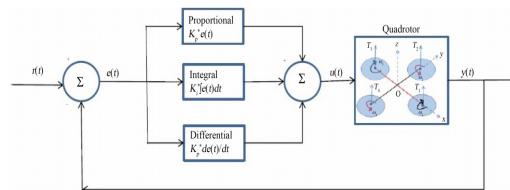
to lose tracking after avoiding an obstacle. Its performance in the presence of many obstacles still needed to be analyzed.

In combination with a Linear Quadratic Estimator (LQE) and Kalman Filter, the LQR algorithm transforms into the Linear Quadratic Gaussian (LQG). This algorithm is for systems with Gaussian noise and incomplete state information. The LQG with integral action was applied in [10] for stabilization of attitude of a quadrotor with good results in hover mode. The advantage of this LQG controller is the fact that you do not need to have complete state information to implement it.

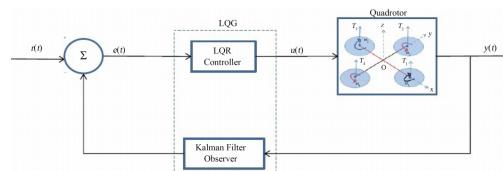
Figure 3 shows the general block diagram of an LQG controller for the quadrotor.

## Sliding Mode Control (SMC)

Sliding mode control is a nonlinear control algorithm that works by applying a discontinuous control signal to the system to command it to slide along a prescribed path. Its main advantage is that it does not simplify the dynamics through linearization and has good tracking. The sliding mode controller was applied to stabilize cascaded under-actuated systems in [11]. The quadrotor system was subdivided into the full-actuated and underactuated systems. The under-actuated system, to which SMC was applied, was further subdivided into underactuated subsystems. Results showed good stability and robustness of the system. Chattering effect of SMC was observed but minimized with a continuous approximation of a pre-determined “sign” function.



**Figure 2.** Block diagram of PID controller applied to the quadrotor.



**Figure 3.** Block diagram of LQG controller applied to the quadrotor.

A sliding mode controller based on Lyapunov stability theory is articulated by Runcharoon and Srichatrapimuk in [12]. The SMC controller was able to stably drive the quadrotor to a desired position and yaw. Tracking was equally good with injected noise, which showed good robustness for the SMC controller.

Figure 4 shows the general block diagram of an SMC controller for the quadrotor.

### **(Integrator) Backstepping Control**

Backstepping control is a recursive algorithm that breaks down the controller into steps and progressively stabilizes each subsystem. Its advantage is that the algorithm converges fast leading to less computational resources and it can handle disturbances well. The main limitation with the algorithm is its robustness is not good. Madani and co-researchers [13] applied backstepping control to stabilize a quadrotor system consisting of an under-actuated, fully actuated and propeller subsystems. Good tracking was achieved for position and yaw angle. Using Lyapunov stability theory, roll and pitch angles were stabilized.

The backstepping approach was also applied in [1] for attitude stabilization of a quadrotor. Using Lyapunov stability analysis, the closed-loop attitude system was found to be asymptotically stable with all states uniformly ultimately bounded in the presence of external disturbance. It was also implicit that the quaternion formulation also helped in the computational side for stabilization in addition to avoiding singularity.

To increase robustness (to external disturbances) of the general backstepping algorithm, an integrator is added and the algorithm becomes Integrator backstepping control as articulated by Fang and Gao in [14]. The integral approach was shown to eliminate the steady-state errors of the system, reduce response time and restrain overshoot of the control parameters.

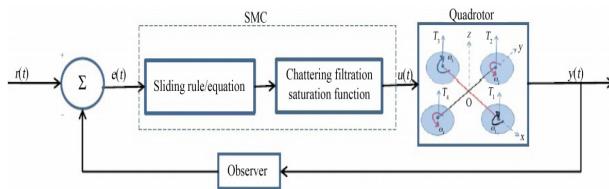
### **Adaptive Control Algorithms**

Adaptive control algorithms are aimed at adapting to parameter changes in the system. The parameters are either uncertain or varying with time. A continuous time-varying adaptive controller, which shows good performance, was implemented by Diao et al. in [15] with known uncertainties in mass, moments of inertia and aerodynamic damping coefficients.

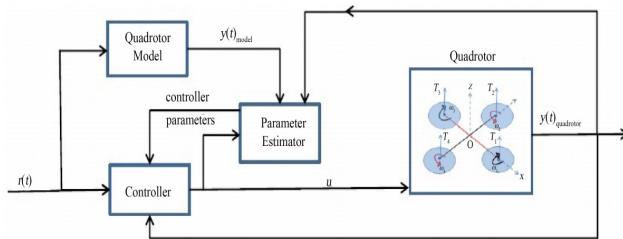
In other work by Palunko et al., an adaptive control scheme using feedback linearization (FBL) is implemented [16] for quadrotors with dynamic changes in the center of gravity. It was observed that when the center of gravity changes, PD and regular feedback linearization techniques were not able to stabilize the system but the adaptive controller was able to stabilize it. The paper touches on a very important futuristic quadrotor that will self-configure in real time when the center of gravity changes.

An adaptive control technique based on rectilinear distance ( $L_1$ ) norm was used in [17] with a tradeoff between control performance and robustness. The modified (linearized) model was able to compensate for constant and moderate wind gusts.

Figure 5 shows the general block diagram of an adaptive controller for the quadrotor clearing showing the parameter estimator and quadrotor model.



**Figure 4.** Block diagram of an SMC controller applied to the quadrotor.



**Figure 5.** Block diagram of an adaptive controller applied to the quadrotor.

## Robust Control Algorithms

Robust control algorithms are designed to deal with uncertainty in the system parameters or disturbances. This guarantees controller performance within acceptable disturbance ranges or un-modeled system parameters. A major limitation normally observed with robust controllers is poor tracking ability. A robust controller was implemented by Bai et al. in [18] for an attitude

controller subsystem based on linear control and robust compensation. Experimental results validated the effectiveness of the controller for controlling attitude.

In another work by Tony and Mackunisy a robust tracking algorithm is presented [19] and achieves asymptotic stability in the presence of parametric uncertainties and unknown nonlinear disturbances. This was achieved with inexpensive control law, no observers, functional approximators or online adaptive updating laws.

## Optimal Control Algorithms

Optimization algorithms are designed to minimize a variable and get the best cost function from a set of alternatives. A specialized case of optimization is known as convex optimization. This is a mathematical optimization technique that deals with minimizing a convex variable in a convex set. The common optimal algorithms include LQR,  $L_1$ ,  $H_\infty$  and Kalman filter. A major limitation of optimization algorithms is generally their poor robustness. A controller that is both relatively robust and  $L_1$ -optimal was applied in [20] for tracking of both attitude and heading. The performance of the controller was experimentally validated and was efficient in error minimization and rejection of persistent disturbances.

An optimization algorithm based on  $H_\infty$  looping was applied by Falkenberg et al. in [21] to a quadrotor with simplified dynamics for iterative parameter identification and attitude control. The algorithm had superior performance with respect to disturbance rejection even under disturbance saturation conditions. However, the ant-windup compensator based on Riccati equations did not seem to show computational efficiency.

An integral optimal predictive  $H_\infty$  controller was implemented by Raffo et al. in [22] for stabilization of rotational movement of a quadrotor and for path following (using model predictive control). The controller achieved sustained disturbance rejection and good robustness. Integral action was key to achieving good tracking in this algorithm.

## Feedback Linearization

Feedback linearization control algorithms transform a nonlinear system model into an equivalent linear system through a change of variables. Some limitation of feedback linearization is the loss of precision due to linearization and requiring an exact model for implementation. Output feedback linearization was implemented as an adaptive control strategy in

[16] for stabilization and trajectory tracking on a quadrotor that had dynamic changes of its center of gravity. The controller was able to stabilize the quadrotor and reconfigure it in real time when the center of gravity changed.

Feedback linearization and input dynamic inversion were implemented by Roza and Maggiore in [23] to design a path-following controller which allowed the designer to specify the speed profile and yaw angle as a function of displacement along the path. Two simulation cases with the quadrotor travelling at different speeds along the path were considered. Both cases showed convergence of velocity and yaw angle.

Feedback linearization was compared with adaptive sliding mode control in [24]. The feedback controller, with simplified dynamics, was found to be very sensitive to sensor noise and not robust. The sliding mode controller performed well under noisy conditions and adaptation was able to estimate uncertainties such as ground effect.

Hence, feedback linearization nonlinear control shows good tracking but poor disturbance rejection. However, feed-back linearization applied with another algorithm with less sensitivity to noise give good performance.

## **Intelligent Control (Fuzzy Logic and Artificial Neural Networks)**

Intelligent control algorithms apply several artificial intelligence approaches, some biologically-inspired, to control a system. Examples include fuzzy logic, neural networks, machine learning, and genetic algorithm. They typically involve considerable uncertainty and mathematical complexity. This complexity and abundant computational resources required are limitations to the use of intelligent systems.

Intelligent control is not limited to fuzzy logic and neural networks but the two are the most widely used. Many other algorithms exist and continue to be formulated. Fuzzy logic algorithms deal with many-valued logic, not discrete levels of truth. An intelligent fuzzy controller was applied by Santos et al. in [25] to control the position and orientation of a quadrotor with good response in simulation. However, a major limitation of this work was the trial and error approach for tuning of input variables.

Artificial neural networks are biologically inspired by the central nervous system and brain. A robust neural networks algorithm was applied to a quadrotor by Nicol et al. in [26] to stabilize against modeling error and considerable wind disturbance. The method showed improvements

with respect to achieving desired attitude and reducing weight drift. Output feedback control was implemented on a quadrotor using neural networks [27] for leader-follower quadrotor formation to learn the complete dynamics of the UAV including un-modeled dynamics. A virtual neural network control was implemented to control all six degrees of freedom from four control inputs. Using Lyapunov stability theory, it was shown that the position, orientation, velocity tracking errors, observer estimation errors and virtual control were semi-globally uniformly ultimately bounded.

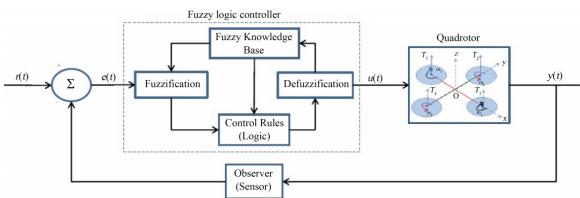
An adaptive neural network scheme was applied in [28] for quadrotor stabilization in the presence of a sinusoidal disturbance. The proposed solution of two parallel single hidden layers proved fruitful as reduced tracking error and no weight drift were achieved.

Figure 6 shows the general block diagram of a fuzzy logic controller implemented on the quadrotor.

## Hybrid Control Algorithms

It is evident that even the best linear or nonlinear algorithms had limitations and no single controller had it all. Researchers have tackled this by combining the philosophies of one or more algorithms. Here are few examples, which are not in any way exhaustive of what is in literature.

A hybrid fuzzy controller with backstepping and sliding mode control was implemented in [29] and successfully eliminated chattering effect of the sliding mode control algorithm. A feedback linearization controller was applied parallel with a high-order sliding mode controller to a quadrotor in [30]. The sliding mode controller was used as an observer and estimator of external disturbances. The system showed good disturbance rejection and robustness.



**Figure 6.** Block diagram an FLC controller applied to the quadrotor.

Adaptive control via backstepping combined with neural networks was presented by Madani and co-researchers in [31]. The backstepping was used to achieve good tracking of desired translational positions and yaw angle

whilst maintaining stability of roll and pitch angles. Neural networks were used to compensate for unmodeled dynamics. The major contribution of the paper was the fact that the controller did not require the dynamic model and other parameters. This meant greater versatility and robustness.

## COMPARISON OF CONTROL ALGORITHMS

Table 1 summarizes the comparison of the various algorithms as applied to quadrotors with all things being equal. The performance of a particular algorithm depends on many factors that may not even be modeled. Hence, this table serves as “fuzzy” guide in accordance with what is presented in this paper and common knowledge.

## DISCUSSION AND CONCLUSION

This survey acts as a stepping stone into further work that will be conducted by the authors on control of quadrotors with tilting propellers. The tilting propellers further increase the number of actuated inputs to give six control inputs for six control outputs, which makes the quadrotor fully actuated. This research direction is relatively new with few examples in literature but with very promising performance achievement (see [32]-[34]). This work will focus on optimal control algorithms to achieve asymptotic stability of the quadrotor under any conditions.

**Table 1.** Comparison of quadrotor control algorithms.

| Control Algorithm   | Characteristic |          |         |             |                  |                           |           |            |                       |                               |               |                |                        |
|---------------------|----------------|----------|---------|-------------|------------------|---------------------------|-----------|------------|-----------------------|-------------------------------|---------------|----------------|------------------------|
|                     | Robust         | Adaptive | Optimal | Intelligent | Tracking ability | Fast convergence/response | Precision | Simplicity | Disturbance rejection | Unmodelled parameter handling | Manual tuning | (Signal) noise | Chattering/energy loss |
| 1. PID              | 1              | 0        | 0       | 0           | 1                | 1                         | 1         | 2          | 0                     | 0                             | 2             | 2              | 0                      |
| 2. Intelligent PID  | 1              | 0        | 0       | 2           | 1                | 1                         | 1         | 1          | 0                     | 0                             | 0             | 1              | 0                      |
| 3. LQR              | 0              | 2        | 1       | 0           | 1                | 1                         | 0         | 1          | 1                     | 0                             | 1             | 1              | 0                      |
| 4. LQG              | 0              | 2        | 2       | 0           | 1                | 1                         | 0         | 0          | 2                     | 0                             | 1             | 0              | 0                      |
| 5. $L_i$            | 0              | 2        | 2       | 0           | 1                | 2                         | 2         | 0          | 1                     | 0                             | 0             | 0              | 0                      |
| 6. $H_i$            | 2              | 1        | 2       | 0           | 2                | 0                         | 1         | 0          | 1                     | 1                             | 0             | 0              | 0                      |
| 7. SMC              | 1              | 2        | 1       | 0           | 2                | 2                         | 2         | 1          | 2                     | 1                             | 0             | 0              | 2                      |
| 8. FBL              | 1              | 1        | 0       | 0           | 2                | 2                         | 2         | 1          | 1                     | 1                             | 0             | 1              | 0                      |
| 9. Backstepping     | 0              | 2        | 0       | 0           | 2                | 0                         | 1         | 0          | 2                     | 1                             | 0             | 0              | 0                      |
| 10. Fuzzy logic     | 1              | 1        | 1       | 2           | 1                | 1                         | 1         | 1          | 1                     | 0                             | 1             | 0              | 0                      |
| 11. Neural networks | 1              | 2        | 2       | 2           | 1                | 1                         | 1         | 0          | 1                     | 1                             | 0             | 0              | 0                      |
| 12. Genetic         | 1              | 2        | 2       | 2           | 1                | 1                         | 1         | 0          | 1                     | 2                             | 0             | 0              | 0                      |

Legend: 0—low to none; 1—average; 2—high. Also, 1 through 5 (Linear); 6 through 12 (Nonlinear).

This paper has reviewed several common control algorithms that have

been used on quadrotors in literature. As evident from the review, no single algorithm presents the best of the required features. It also been discussed that getting the best performance usually requires hybrid control schemes that have the best combination of robustness, adaptability, optimality, simplicity, tracking ability, fast response and disturbance rejection among other factors. However, such hybrid systems do not guarantee good performance; hence a compromise needs to be found for any control application on which of the factors would be most appropriate. The jury is still out as to what mathematical model would give the best overall performance.

The designed quadrotor will be used for game counting at the National Parks; hence it should have the following important characteristics: high endurance, low noise, high agility, high cruising and VTOL ability. The quadrotor will be a great aid for nature conservationists that are looking for innovative ways to combat poaching and get accurate animal statistics. Future possibilities with this type of quadrotor would include the possibility of equipping it with a manipulator that can do other things such as spraying of medicines or tagging.

## REFERENCES

1. Huo, X., Huo, M. and Karimi, H.R. (2014) Attitude Stabilization Control of a Quadrotor UAV by Using Backstepping Approach. Mathematical Problems in Engineering, 2014, 1-9.
2. Mahony, R., Kumar, V. and Corke, P. (2012) Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. Robotics Automation Magazine, 19, 20-32. <http://dx.doi.org/10.1109/MRA.2012.2206474>
3. Lee, B.-Y., Lee, H.-I. and Tahk, M.-J. (2013) Analysis of Adaptive Control Using On-Line Neural Networks for a Quadrotor UAV. 13th International Conference on Control, Automation and Systems (ICCAS), 20-23 October 2013, 1840- 1844.
4. Bouabdallah, S. (2007) Design and Control of Quadrotors with Application to Autonomous Flying. Ph.D. Dissertation, Lausanne Polytechnic University, Lausanne.
5. John, S. (2013) Artificial Intelligent-Based Feedforward Optimized PID Wheel Slip Controller. AFRICON, 12 September 2013, Pointe-Aux-Piments, 1-6.
6. Lee, K.U., Kim, H.S., Park, J.-B. and Choi, Y.-H. (2012) Hovering Control of a Quadroto. 12th International Conference on Control, Automation and Systems (ICCAS), 17-21 October 2012, 162-167.
7. Li, J. and Li, Y. (2011) Dynamic Analysis and PID Control for a Quadrotor. International Conference on Mechatronics and Automation (ICMA), 7-10 August 2011, 573-578.
8. Bouabdallah, S., Noth, A. and Siegwart, R. (2004) PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), Vol. 3, 28 September-2 October 2004, 2451-2456.
9. Cowling, I.D., Yakimenko, O.A., Whidborne, J.F. and Cooke, A.K. (2007) A Prototype of an Autonomous Controller for a Quadrotor UAV. European Control Conference, July 2007, Kos, 1-8.
10. Minh, L.D. and Ha, C. (2010) Modeling and Control of Quadrotor MAV Using Vision-Based Measurement. International Forum on Strategic Technology (IFOST), 13-15 October 2010, 70-75.
11. Xu, R. and Ozguner, U. (2006) Sliding Mode Control of a Quadrotor Helicopter. Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, 13-15 December 2006, 4957-4962.
12. Runcharoon, K. and Srichatrapimuk, V. (2013) Sliding Mode Control

- of Quadrotor. Proceedings of the 2013 International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAECE), Konya, 9-11 May 2013, 552-557.
- 13. Madani, T. and Benallegue, A. (2006) Backstepping Control for a Quadrotor Helicopter. Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, 9-15 October 2006, 3255-3260.
  - 14. Fang, Z. and Gao, W. (2011) Adaptive Integral Backstepping Control of a Micro-Quadrotor. Proceedings of the 2nd International Conference on Intelligent Control and Information Processing (ICICIP), Harbin, 25-28 July 2011, 910- 915.
  - 15. Diao, C., Xian, B., Yin, Q., Zeng, W., Li, H. and Yang, Y. (2011) A Nonlinear Adaptive Control Approach for Quadrotor UAVs. Proceedings of the 8th Asian Control Conference (ASCC), Kaohsiung, 15-18 May 2011, 223-228.
  - 16. Palunko, I. and Fierro, R. (2011) Adaptive Control of a Quadrotor with Dynamic Changes in the Center of Gravity. Proceedings of the 18th IFAC World Congress, Milan, 28 August-2 September 2011, 2626-2631.
  - 17. De Monte, P. and Lohmann, B. (2013) Position Trajectory Tracking of a Quadrotor Helicopter Based on L1 Adaptive Control. Proceedings of the 2013 European Control Conference (ECC), Zurich, 17-19 July 2013, 3346-3353.
  - 18. Bai, Y., Liu, H., Shi, Z. and Zhong, Y. (2012) Robust Control of Quadrotor Unmanned Air Vehicles. Proceedings of the 31st Chinese Control Conference (CCC), Hefei, 25-27 July 2012, 4462-4467.
  - 19. Tony, C. and Mackunisy, W. (2012) Robust Attitude Tracking Control of a Quadrotor Helicopter in the Presence of Uncertainty. Proceedings of the IEEE 51st Annual Conference on Decision and Control (CDC), Maui, 10-13 December 2012, 937-942.
  - 20. Satici, A., Poonawala, H. and Spong, M. (2013) Robust Optimal Control of Quadrotor UAVs. IEEE Access, 1, 79-93. <http://dx.doi.org/10.1109/ACCESS.2013.2260794>
  - 21. Falkenberg, O., Witt, J., Pilz, U., Weltin, U. and Werner, H. (2012) Model Identification and H1 Attitude Control for Quadrotor MAVs. Intelligent Robotics and Applications, 460-471.
  - 22. Raffo, G.V., Ortega, M.G. and Rubio, F.R. (2010) An Integral Predictive/

- Nonlinear h? Control Structure for a Quadrotor Helicopter. *Automatica*, 46, 29-39. <http://dx.doi.org/10.1016/j.automatica.2009.10.018>
- 23. Roza, A. and Maggiore, M. (2012) Path Following Controller for a Quadrotor Helicopter. Proceedings of the American Control Conference (ACC), Montreal, 27-29 June 2012, 4655-4660. <http://dx.doi.org/10.1109/ACC.2012.6315061>
  - 24. Lee, D., Kim, H.J. and Sastry, S. (2009) Feedback Linearization vs. Adaptive Sliding Mode Control for a Quadrotor Helicopter. *International Journal of Control, Automation and Systems*, 7, 419-428. <http://dx.doi.org/10.1007/s12555-009-0311-8>
  - 25. Santos, M., Lopez, V. and Morata, F. (2010) Intelligent Fuzzy Controller of a Quadrotor. Proceedings of the 2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Hangzhou, 15-16 November 2010, 141-146.
  - 26. Nicol, C., Macnab, C.J.B. and Ramirez-Serrano, A. (2008) Robust Neural Network Control of a Quadrotor Helicopter. Proceedings of the Canadian Conference on Electrical and Computer Engineering, Niagara Falls, 4-7 May 2008, 1233-1237.
  - 27. Dierks, T. and Jagannathan, S. (2010) Output Feedback Control of a Quadrotor UAV Using Neural Networks. *IEEE Transactions on Neural Networks*, 21, 50-66. <http://dx.doi.org/10.1109/TNN.2009.2034145>
  - 28. Boudjedir, H., Yacef, F., Bouhali, O. and Rizoug, N. (2012) Adaptive Neural Network for a Quadrotor UnmannedAerial Vehicle. *International Journal in Foundations of Computer Science and Technology*, 2, 1-13. <http://dx.doi.org/10.5121/ijfcst.2012.2401>
  - 29. Zeghlache, S., Saigaa, D., Kara, K., Harrag, A. and Bouguerra, A. (2012) Backstepping Sliding Mode Controller Improved with Fuzzy Logic: Application to the Quadrotor Helicopter. *Archives of Control Sciences*, 22, 315-342.
  - 30. Benallegue, A., Mokhtari, A. and Fridman, L. (2006) Feedback Linearization and High Order Sliding Mode Observer for a Quadrotor UAV. Proceedings of the International Workshop on Variable Structure Systems, Alghero, 5-7 June 2006, 365-372.
  - 31. Madani, T. and Benallegue, A. (2008) Adaptive Control via Backstepping Technique and Neural Networks of a Quadrotor Helicopter. Proceedings of the 17th World Congress of the International Federation of Automatic Control, Seoul, July 6-11 2008, 6513-6518.

32. Cetinsoy, E., Dikyar, S., Hancer, C., Oner, K.T., Sirimoglu, E., Unel, M. and Aksit, M.F. (2012) Design and Construction of a Novel Quad Tilt-Wing UAV. *Mechatronics*, 22, 723-745. <http://dx.doi.org/10.1016/j.mechatronics.2012.03.003>
33. Ryll, M., Bulthoff, H. and Giordano, P. (2012) Modeling and Control of a Quadrotor UAV with Tilting Propellers. *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, 14-18 May 2012, 4606-4613.
34. Senkul, F. and Altug, E. (2013) Modeling and Control of a Novel Tilt—Roll Rotor Quadrotor UAV. *Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, 28-31 May 2013, 1071-1076.



# **CHAPTER**

# **3**

## **Central Command Architecture for High-Order Autonomous Unmanned Aerial Systems**

---

**Larry M. Silverberg, Chad Bieber**

Mechanical and Aerospace Engineering, North Carolina State University,  
Raleigh, USA

### **ABSTRACT**

This paper is the first in a two-part series that introduces an easy-to-implement central command architecture for high-order autonomous unmanned aerial systems. This paper discusses the development and the second paper presents the flight test results. As shown in this paper, the central command architecture consists of a central command block, an autonomous planning

---

**Citation:** Silverberg, L. and Bieber, C. (2014), “Central Command Architecture for High-Order Autonomous Unmanned Aerial Systems”. *Intelligent Information Management*, **6**, 183-195. doi: 10.4236/iim.2014.64019.

**Copyright:** © 2014 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

block, and an autonomous flight controls block. The central command block includes a staging process that converts an objective into tasks independent of the vehicle (agent). The autonomous planning block contains a non-iterative sequence of algorithms that govern routing, vehicle assignment, and deconfliction. The autonomous flight controls block employs modern controls principles, dividing the control input into a guidance part and a regulation part. A novel feature of high-order central command, as this paper shows, is the elimination of operator-directed vehicle tasking and the manner in which deconfliction is treated. A detailed example illustrates different features of the architecture.

Keywords: Unmanned Aerial Vehicles, Autonomy, Central Command, High-Order Systems, Deconfliction

## INTRODUCTION

Unmanned aerial vehicles (UAV) are gaining interest with relaxing restrictions in civilian airspaces. Looking ahead, a systematic approach is needed for autonomous unmanned aerial systems (AUAS) consisting of a large number of vehicles (agents). This paper, which is the first in a series of two papers, presents an easy-to-implement architecture for high-order AUAS; the second paper presents flight test results. The different AUAS approaches are referred to in this paper as command approaches with central command at one extreme and individual command at the other. In central command, the AUAS authority lies in one entity, and the vehicle requires only limited information about its environment. Central command is naturally suited to problems dictated by global objectives, such as search and surveillance in precision agriculture, border security, law enforcement, and wildlife and forestry services, to name a few. In individual command, the authority lies in the individual vehicle. The vehicle requires local situational awareness, communication with other vehicles, and understanding of system objectives. Individual command is naturally suited to problems dictated by local objectives that require coordination, like those found in air traffic control problems and in coordinated pick and place problems.

When an AUAS operates multiple vehicles in close proximity some method of preventing collisions becomes necessary. Collision avoidance, or the real-time act of recognizing and maneuvering to avoid an impending conflict, also called “see and avoid”, is a fundamental principle of current flight operations and will be a requirement of future operations that share manned airspace. In contrast, deconfliction, or the act of planning paths in

advance that do not conflict, is a central command level method of preventing conflicts as events are planned. Collision avoidance and deconfliction are not mutually exclusive; deconfliction prevents collisions between planned or known events whereas collision avoidance prevents collisions between unplanned events. Future systems will incorporate both methods. This paper develops a robust method of deconfliction based on an assignment method that produces non-crossing paths. This method is presented within an easy-to-implement architecture that is scalable to high-order AUAS. The method section introduces the basic architecture and the results section provides a detailed example.

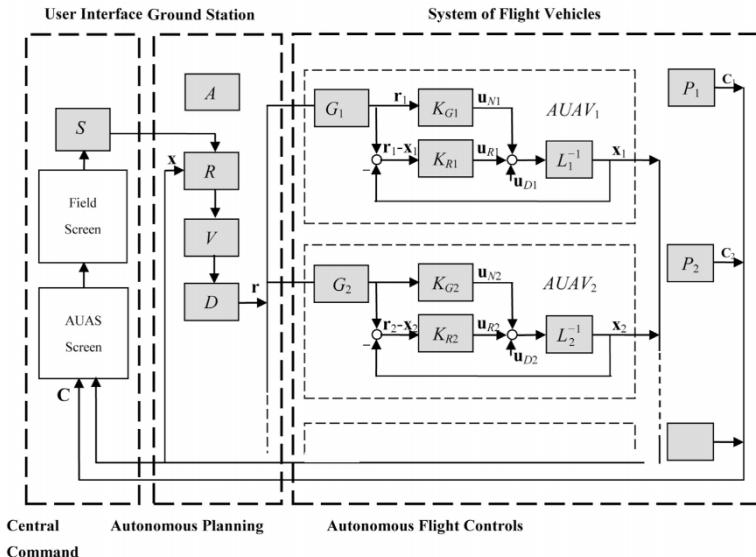
The requirements of an architecture under which high-order AUAS become feasible is the subject of ongoing research. Bellingham, Tillerson, et al., 2003 [1] separated their architecture into 1) routing, 2) vehicle assignment of tasks, and 3) deconfliction, after which the vehicles receive their directions. Cummings, Bruni, et al., 2007 [2] separated their architecture into 1) central mission and payload management, 2) navigation, 3) vehicle autopilots, and 4) flight controls. Shima and Rassmussen [3] describe a pantheon of architectures across the three axes of command, information availability and performance. Concerning user load, Adams, Humphrey, et al., 2009 [4] studied the problems that arise with human attention when one or more users give commands and monitor data. Donmez, Nehme, et al., 2010 [5] showed that wait time increases with the number of vehicles in AUAS that do not have full autonomous planning. These studies suggest that full autonomy is a necessary requirement in high-order central command. About computational requirements, routing and task assignment are combinatorial problems that have prohibitive computational requirements for high-order AUAS, necessitating heuristics. Gardiner, Ahmed, et al., 2001 [6] reviewed real-time collision avoidance algorithms for multiple UAVs and illustrated the computational complexities that limit the number of vehicles and flight time. How, King, et al., 2004 [7] demonstrated a simultaneous task assignment and deconfliction approach with 8 vehicles. These studies suggest the use of heuristics and non-iterative algorithms to reduce computational complexity. Pertaining to the question of the division of labor between remote and on-board processing, it is natural to perform the flight controls on-board to reduce communication throughput requirements to remote transmission of off-board command decisions at perhaps 1 Hz or less. Edwards and Silverberg, 2010 [8] exploited this division of labor in an autonomous soaring flight demonstration. Concerning the choice between acquiring on-board states from synthesized relative measurements

(using situational awareness) or from inertial measurements (like IMU and GPS), Levedahl and Silverberg, 2005 [9] showed that acquiring on-board states from inertial measurements is particularly well-suited to the central command problem because of the sensitivity issues that arise in synthesized relative measurements in high-order AUAS.

## METHOD

### The High-Order Central Command (HOCC) Architecture

The HOCC architecture is described by the block diagram shown in Figure 1. Drawing from the work of Bellingham, Tillerson, Richards, and Howe [1], the HOCC architecture is divided into central command, autonomous planning, and autonomous flight controls. The central command block directs the execution of objectives. It contains a staging process that converts the high-level objective into assignable tasks. The autonomous planning block is an efficient, non-iterative sequence of three processes called Routing (R), Vehicle Assignment (V), and Deconfliction (D). A unique feature of the architecture introduced in this paper, as shown below, is that while the processes have internal iterations, they never require going back to a previous process (external iterations).



**Figure 1.** Block diagram for the HOCC architecture.

The complexity of the individual processes used, namely the Visibility Graph method, the A\* method, the Hungarian method, are well known. For example, the number of calculations in A\* is on the order of  $n m E \log(V)$  where  $n$  is number of vehicles,  $m$  is number of tasks,  $E$  is number of obstacle edges, and  $V$  is number of obstacle vertices and the number of calculations in the Hungarian method is on the order of  $n^4$ . The elimination of external iterations, as described below, leads to a robust, computationally efficient algorithm. Throughout this paper, we only consider vehicles that are assignable to every task (homogeneity).

## Staging

As shown in Figure 1, a user is provided information from an operational area and other field parameters, and data  $C_j$  ( $j = 1, 2, \dots$ ) generated by AUAV payload  $P_j$  ( $j = 1, 2, \dots$ ). Using this and other information, the user commands an objective. In absence of fully autonomous planning algorithms that remove the vehicle from the calculus, user load would be, at the very least, proportional to the number of vehicles, which is prohibitive in high-order systems. Thus a critical feature of the HOCC architecture introduced in this paper is the decoupling of the objective from the vehicles and, instead, the expression of the commands in terms of a relatively small number of parameters. Operator commanded dynamic re-tasking of individual vehicles no longer applies.

The staging process converts an objective into a set of tasks. This paper considers spatial staging, such as passing over a point (for imagery or delivery), a line scan (following a line segment) which could be part of covering a wider region, and loitering (orbiting a point) which could arise when a vehicle is waiting for further instructions or to monitor a point of interest. The spatial tasks have starting points, so the conversion of an objective into a set of tasks determines a set of spatial points that vehicles need to reach. During the staging process the candidate routes are unknown and the assignment of vehicles to tasks is unknown.

## Routing

Autonomous planning is the second block of the HOCC architecture and routing is its first process. Support tasks, such as launching or recovering and refueling, are conducted by the administrator (A) who operates in the background. The routing process itself is performed in two parts, mapping the environment and choosing a path [10]. The first part produces a graph of

a vehicle-task route segment. In this paper, the Visibility Graph method [11] is employed. Briefly, this method produces a graph of possible connections between vehicles and obstacle vertices, tasks and vertices, and between the vertices themselves, representing a graph of possible routes through the environment. Note that search areas are regarded as obstacles to prevent routes from passing through them, which can otherwise lead to conflicts. After determining the graph between a vehicle point and a task, the second part is to find the shortest route between them. This part is performed by the A\* method [12]. This method does not first find the different paths from which the shortest path could be found. Instead, it searches in the direction of the task, and can leave portions of the graph unsearched. The shortest vehicle-task routes are collected into a matrix of vehicle-task route lengths used in vehicle assignment. New routing is calculated each time a vehicle completes a task and when a new objective is initiated.

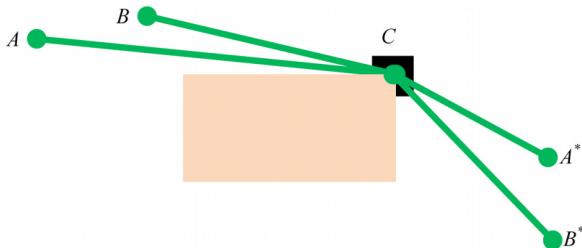
## Vehicle Assignment

A frequent goal in AUAS path planning, because of limited fuel and flight duration requirements, is to determine vehicle routes by minimizing distance of travel of individual vehicles. An important variation on this goal, which is introduced in this paper, is the minimization of total distance of travel of all of the vehicles (a min-sum assignment). The minimization of total distance of travel leads to an importing routing principle: The routes determined by minimizing total distance of travel do not cross. The mathematical proof of this routing principle is presented in the Appendix. Although non-crossing is guaranteed, the paths can still touch obstacles. Indeed, vehicle-task segments can share vertices of obstacles, as shown in Figure 2 below.

As shown, two vehicle task routes share vertex C. The vehicle at A travels to C and then to A\*. The vehicle at B travels to C and then to B\*. Notice that the total distance of travel after the shared vertex,  $CA^* + CB^*$ , is the same whether A travels to A\* or to B\*. Indeed, the solution to the minimization problem is indeterminate when two routes share a vertex. The presence of this indeterminacy, however, presents an opportunity. It allows the algorithm to ensure that the vehicles do not conflict after passing the shared vertex C, such as a task at A\* requiring a vehicle to loiter for a short time, thereby staying very near the path of the vehicle proceeding to B\*. Assume that the vehicles begin at the same time and that vehicle B reaches C before vehicle A. The algorithm selects the vehicle that first reaches vertex C to travel to the farthest point, in this case, the vehicle at B travels to B\*. The inspection of a potential conflict associated with a separation distance falling below

a minimal level is thus reduced to checking separation distances while the vehicles are traveling along the two segments AC and BC before the shared vertex. The desire to ensure that vehicle separation distances exceed required minimums makes total distance of travel an important cost function in AUAS path planning. Thus, the min-sum assignment enables rapid deconfliction of planned paths by checking only shared vertices.

The assignments that minimize the total distance of travel are determined in this paper by the Hungarian method [13]. It is likely that an objective will have more tasks than the system has vehicles, or that a new objective might be added. This leads to repeated application of the assignment method. The assignments are updated each time a task is completed or when a new objective arrives. Therefore, the number of assignments is equal to the total number of tasks plus the number of objectives. The computational cost of the Hungarian method is smaller than the computational cost of the Visibility method and the A\* method when obstacle vertices outnumber tasks or vehicles.



**Figure 2.** Shared vertex.

## Deconfliction

As described above and proven in the Appendix, potential conflicts in this HOCC architecture are treated mainly by avoiding them in the first place (in contrast with collision avoidance). As stated earlier, this was accomplished by utilizing the non-crossing property of a min-sum assignment. Non-crossing is a more powerful statement than deconfliction. A deconflicted route means two vehicles must not occupy the same space at the same time, but the non-crossing property shows that the entire length of the paths do not cross at any time. This decouples vehicle speed from the path planning and is useful in solving the potential conflicts in the vicinity of obstacle vertices. By loosening any initial assumptions about vehicle speed, shared

vertices can be deconflicted by small speed adjustments without changing the path of the vehicle. Because the assigned path is known not to intersect with other paths, this small speed adjustment cannot cause other routing conflicts, eliminating the need to re-check the final solution for conflicts. The elimination of route intersections leaves potential conflicts only in the neighborhood of boundaries of obstacles that are shared by more than one vehicle. When a potential conflict is identified, the first step is to determine whether or not a conflict arises. The simplest action is to eliminate a conflict by adjusting the speed of one or more vehicles without changing any routes. Minor speed adjustments around the cruise speed are almost always feasible. In extreme cases timing can be corrected by placing vehicles in holding patterns, and is scalable to high-order AUAS when route segments are sufficiently long. Other remedies, like route adjustment [9], become a last resort.

## Autonomous Flight Controls

The autonomous planning block leads to vehicle assignments. The vehicle assignments represent low frequency and bandwidth information that can be performed off-board. The autonomous flight controls, in contrast, represent higher frequency and bandwidth processes that are ideally suited for on-board implementation. Best practices in autonomous flight controls are reviewed below. The first step is to convert assigned routes to guidance paths ( $G$ ). At this stage routes are refined to account for turning radii and other properties to enable vehicles to follow the routes precisely. Indeed, slopes and curvatures of connected line segments are discontinuous at route vertices and can't be followed precisely. Circular arcs between line segments [14] produce paths whose directions of travel (first derivatives) are continuous. Further refinement produces paths whose turning radii (second derivatives) are continuous, which allows time for the AUAV to change its flight direction through roll. The next step is to regulate the guidance paths. The  $k$ -th vehicle regulates its guidance path  $r_k$  independent and without awareness of other vehicles. An external disturbance  $u_{Dk}$ , like a wind load, causes deviations from the actual flight path  $x_k$ . The control surfaces are adjusted by a guidance input  $u_{Nk}$  and a regulation input  $u_{Rk}$ . The guidance input serves to fly the vehicle along its guidance path in the absence of external disturbances and the regulation part compensates for errors. The primary errors are caused by external disturbances but others errors originate from prescribing non-smooth guidance paths, transducer error, and model

errors in the plant  $L_k$ . The guidance state  $\mathbf{x}_{Nk} = \begin{bmatrix} \mathbf{r}_{Gk}^T & \theta_{Gk}^T & \mathbf{v}_{Gk}^T & \omega_{Gk}^T \end{bmatrix}^T$  is received while following the actual state  $\mathbf{x}_k = \begin{bmatrix} \mathbf{r}_k^T & \theta_k^T & \mathbf{v}_k^T & \omega_k^T \end{bmatrix}^T$ . Let  $\mathbf{u}_k$  represent the control input vector. The equations governing the motion of the vehicle, the autonomous navigator, the autonomous regulator and the form of the control input are

$$L_k \mathbf{x}_k = \mathbf{u}_k \quad K_{Gk} \mathbf{r}_k = \mathbf{u}_{Nk} \quad K_{Rk} (\mathbf{r}_k - \mathbf{x}_k) = \mathbf{u}_{Rk} \quad \mathbf{u}_k = \mathbf{u}_{Nk} + \mathbf{u}_{Rk} + \mathbf{u}_{Dk}$$

where  $K_{Gk}$  is the guidance operator,  $K_{Rk}$  is the regulator operator, and where the control input is the sum of a guidance input  $\mathbf{u}_{Gk}$ , a regulator input  $\mathbf{u}_{Rk}$ , and a disturbance input  $\mathbf{u}_{Dk}$ .

This is called the modern control form [15]. Note that the operator notation above does not specify the realization, that is, whether the system is represented by differential equations, difference equations, algebraic equations, or a hybrid. Under ideal conditions ( $K_{Gk} = L_k$  and the guidance input is smooth), the guidance input causes the vehicle to follow the guidance state vector exactly. It follows that the input-output relationship for the vehicle, the error, and the characteristic equation are:

$$(L_k + G_{Rk}) \mathbf{e}_k = \mathbf{u}_{Dk} \quad \mathbf{e}_k = \mathbf{x}_k - \mathbf{r}_k \quad 0 = \det[zI - (L_k + G_{Rk})]$$

where  $I$  is the identity operator and  $\mathbf{e}_k$  is the state error vector. The characteristic equation is independent of the guidance state vector. Therefore, the regulator gain matrix  $G_{Rk}$  determines the vehicle's stability characteristics (settling time, peak-overshoot, and steady-state error) independent of the guidance state vector  $\mathbf{r}_k$ .

These best practices pertaining to autonomous flight controls are summarized as an AUAS Flight Controls Principle: Assume that a realizable guidance path has been produced and that a vehicle has the control authority capable of maintaining the guidance path within certain limits placed on settling time, peak overshoot, and steady-state error.

Then autonomous guidance and autonomous regulation can be designed by separate and distinct methods.

Following best practices in autonomous flight controls lead to vehicle paths that more closely follow their guidance paths, which is of particular interest in autonomous systems that are not instrumented with on-board collision avoidance systems.

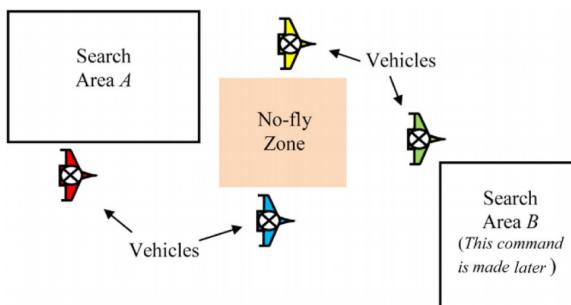
## RESULTS

### Problem Statement

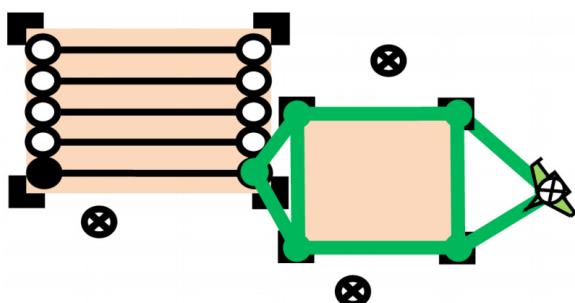
Four vehicles loiter in an air space that contains a no-fly zone (obstacle), as shown in Figure 3. At time  $t = 0$ , an AUAS user commands the AUAS to search area A. At time  $t = T$ , while the vehicles are searching area A, the user gives a second command to search area B. The AUAS, once completing its two objectives, is commanded to return to the original loiter points.

### HOCC Solution

The scenario begins with the AUAS user commanding the first objective to survey area A. The command to survey area B is made later. The staging process converts the objective to search area A into 5 line segment tasks after which the vehicles are tasked to return to their original loiter points (see Figure 4). Each line segment is regarded as a task and a vehicle can follow any line segment from the left or from the right.



**Figure 3.** Problem statement.



**Figure 4.** Part of the graph between a vehicle and the end point of a task.

Figure 4 shows the first step in determining the shortest route between any one vehicle point and the end points associated with a task. There are two end points associated with each task since the vehicle is allowed here to follow a line segment from the left or the right. Consider the vehicle point and the pair of end points associated with the bottom line segment, as shown. The graph connects solid dots that designate allowable points and the squares that designate obstacle vertices (points designated by circles are not allowed). This figure only shows some of the possible connections in the graph. The first step in determining the best route, when employing the visibility graph method, is to produce a graph that connects the vehicle point to the two end points. The number of different routes along the segments, as shown, is a combinatorial problem, and even enumerating all of the possibilities is computationally expensive in high-order AUAS. As an alternative, the visibility graph method is used, recognizing that it does not consider all of the possibilities.

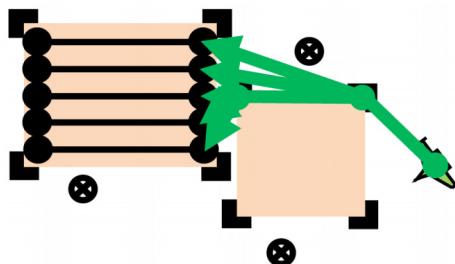
After determining the possible graphs between a vehicle point and a task, the next step is to find the shortest route between them. The shortest vehicle-task routes between one of the vehicles and each of the five tasks are shown in Figure 5. There are comparable shortest vehicle-task routes between each of the other vehicles and the five tasks, as well (not shown). It happens that the best end points are each on the right side of the search area for the vehicle shown. You will see in Figure 6 that there is a different vehicle for which its shortest vehicle task route is to an end point located on the left side of the search area.

Once the shortest vehicle-task routes are determined, vehicles need to be assigned to tasks. This is done by minimizing the total distance of travel of the vehicles (see Figure 6).

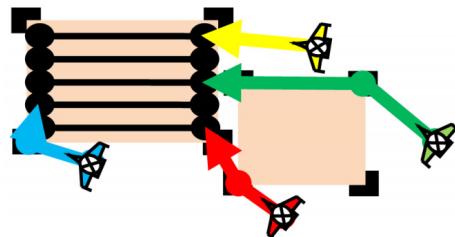
In Figure 7, the vehicles are shown on their way to their first task when a second objective arrives. The staging process turns the second search area into 5 new line segment tasks. The vehicles are shown at the instant that the new objective arrives.

As shown, two tasks are currently being executed, and their end-points are no longer available to be assigned. The points that are available are shown as solid dots. When no longer available, the end-points are indicated by circles. Also, notice that past routes are indicated by thin lines and that the proposed routes are indicated by thick lines. As shown, there will be a total of ten tasks and two objectives so the total number of time intervals will be twelve.

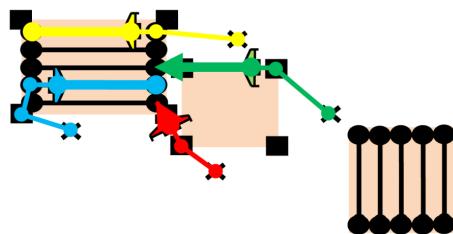
Figure 8 is shown at the same instant as Figure 7 but the newly calculated routes are now shown. As shown, only the assignment of the green vehicle changes and heads to the new search area. Note that vehicles in the process of executing a task are assigned to new tasks from the end point of the task currently being executed. When the new objective arrived, the blue and yellow vehicles were performing tasks, so their proposed paths beyond their current tasks were also included in the calculation. The proposed route of the yellow vehicle is to the left end point of the task just below its current task and the proposed route of the blue vehicle is to the right end of the task just above its current task. The red vehicle, after the new calculation, is assigned to the same end point as in the previous time interval.



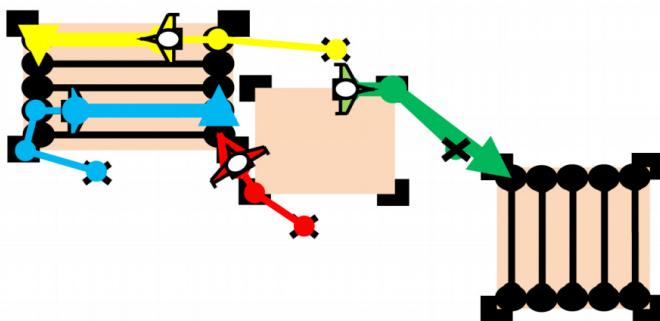
**Figure 5.** Five shortest vehicle-task paths for one of the vehicles during the 1<sup>st</sup> time interval.



**Figure 6.** Routes over the 1<sup>st</sup> time interval.



**Figure 7.** A new objective during the 1<sup>st</sup> time interval initiates the 2<sup>nd</sup> time interval.



**Figure 8.** Routes over 2<sup>nd</sup> time interval.

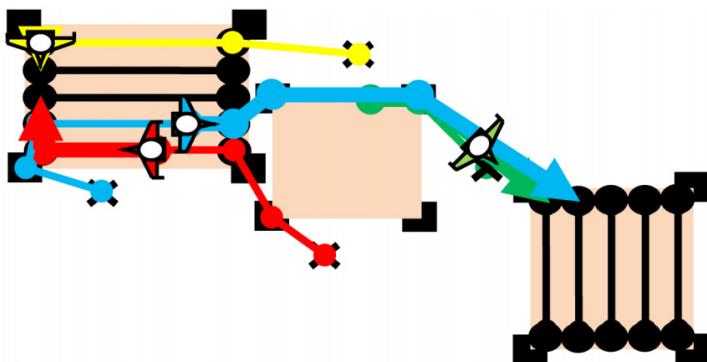
The yellow vehicle is the first vehicle to complete a task. The completion of its task triggers the start of the third time interval. Figure 9 shows the vehicles at this instant, the past routes (thin lines), and the routes proposed over this time interval (thick lines). Notice that, because the red vehicle is executing its task, its next assignment is being considered. As the remaining tasks in the original search area are assigned, it is easy to see that the blue vehicle has a shorter route to the new area than the red or yellow vehicles. Thus the red vehicle is assigned the task previously allotted to blue, and blue's assignment changes to the new search area. The assignments given to the green and the yellow vehicles are the same as in the previous time interval.

In Figure 10, the yellow vehicle has just completed its 2<sup>nd</sup> task. The red and blue vehicles completed their first tasks, so a total of 4 tasks have been completed at this instant. Indeed, Figure 10 shows the start of the 6<sup>th</sup> time interval. As stated earlier, routes never cross but they can share obstacle vertices (squares). Indeed, the blue, red, and yellow vehicles are approaching the top right vertex. The presence of shared vertices prompts actions in the deconfliction block. The first action is to check whether there is a conflict in the neighborhood of the shared vertex. The checking of this potential conflict reveals that the blue vehicle reaches the vertex first, then the yellow vehicle and finally the red vehicle. Moreover, the separation distances were all determined to be larger than the required minimum. Recall to avoid potential conflicts, that the vehicle that first reaches the shared vertex is assigned the route to the farthest end point. Thus, the proposed route of the blue vehicle is to the right-most task, the proposed route of the yellow vehicle is to the next right-most task, and the proposed route of the red vehicle is to the third right-most task.

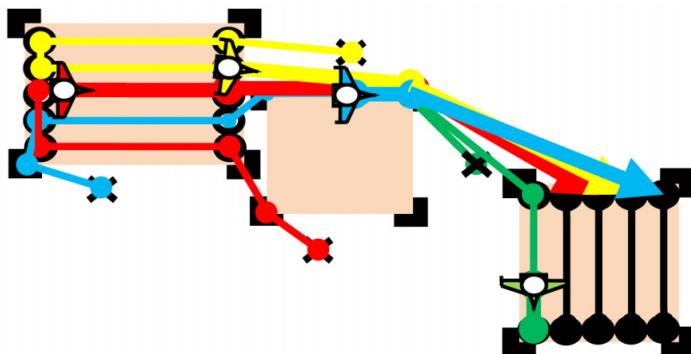
In Figure 11, the green vehicle has just completed its task, triggering the 9<sup>th</sup> time interval. The green vehicle has been assigned the task of following the middle task in search area B, completing the available tasks and leaving the red vehicle unassigned. This unassigned vehicle is assigned to a loiter point to await further tasks.

In Figure 12, the green vehicle has just completed a task, triggering the 10<sup>th</sup> time interval. During this time interval, the bottom left vertex of the 2<sup>nd</sup> search area is a shared vertex. The green vehicle arrives at the vertex before the blue vehicle so the green vehicle returns to the loiter point that is farther from the vertex.

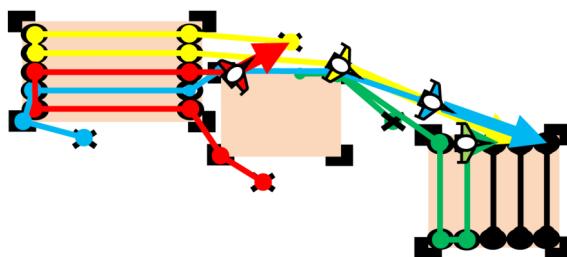
In Figure 13, the yellow vehicle completes the last task, triggering the 12<sup>th</sup> time interval. The red vehicle is already at a loiter point, the green vehicle is on route to a loiter point leaving the yellow and blue vehicles that happen to be tasked to pass the bottom left vertex in search area B. The blue vehicle reaches the vertex first, so it is routed to the farthest remaining vertex.



**Figure 9.** Routes over the 3<sup>rd</sup> time interval.



**Figure 10.** Routes over the 6<sup>th</sup> time interval.

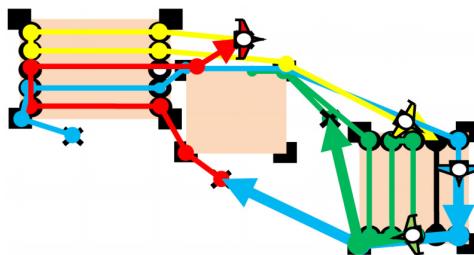


**Figure 11.** Routes over the 9<sup>th</sup> time interval.

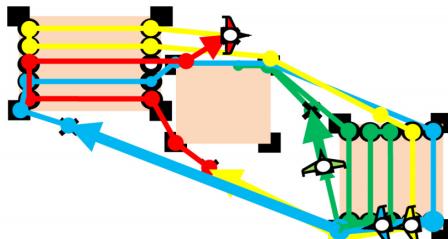
Figure 14 shows the complete routes of all four vehicles, with the vehicles having returned to the loiter points.

## SUMMARY AND CONCLUSIONS

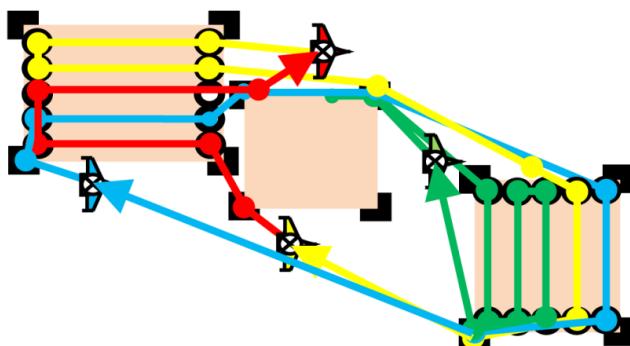
This paper presented an easy-to-implement central command architecture that is well suited to a large number of unmanned aerial vehicles—overcoming present limitations pertaining to operator task load, computational complexity and vehicle deconfliction. The architecture consists of a central command block, an autonomous planning block, and an autonomous flight controls block. Building on results found in the literature pertaining to



**Figure 12.** Routes over the 10<sup>th</sup> time interval.



**Figure 13.** 12<sup>th</sup> (last) time interval.



**Figure 14.** The end.

Larry M. Silverberg, Chad Bieber AUAS, the central command block is fully autonomous thereby overcoming user load limitations associated with a large number of vehicles. The autonomous planning part is a non-iterative sequence of algorithms that govern routing, vehicle assignment, and deconfliction. The non-iterative nature of the algorithms overcomes computational limitations. The routing exploits a new routing principle that naturally prevents vehicle routes from crossing. An example illustrates the different features of the architecture. This paper is the first in a series of two papers; the second will report on flight test results.

## APPENDIX: THE SHORTEST LINES BETWEEN TWO POINT SETS DO NOT CROSS

Definition: Let two curved line segments intersect. The intersection point is a crossing point if the corresponding tangent lines are not parallel and the point is located in the interior of the line segments (is not an end point of a line segment).

Definition: A planar domain  $D$  in  $\mathbb{R}^2$  refers to a subset of  $\mathbb{R}^2$  that can be bounded externally by a closed curve and internally by closed curves.

Definition: Refer to the set of points  $\{\mathbf{Q}_i\}_{i=1}^{i=n}$  as point set  $A(n)$ , and to the set of points  $\{\mathbf{q}_i\}_{i=1}^{i=m}$  as point set  $B(m)$ , in which points  $Q_i$  and  $q_i$  are contained in a planar domain  $D$  and  $n \leq m$ . The curved line segments in  $D$  that connect each point in  $A(n)$  to a unique point in  $B(m)$  compose the connecting set  $C(n, m)$  of  $A(n)$  and  $B(m)$ . Each curved line segment is called a connecting line segment.

Definition: The connecting line segments in a connecting set  $C(n, m)$  of  $A(n)$  and  $B(m)$  of the shortest total length  $L = \sum_i^n L_i$  are called minimal connecting line segments.

## Lemma: Line Segments

Consider the following three well-known results:

- 1). The shortest length  $L$  of a curved line segment is produced by the straight line segment between the two end points (see Figure A1(a)). This is more simply stated as “The shortest distance between two points is a straight line.”
- 2). The length of any one side of a triangle is smaller than the total length of the other two sides (see Figure A1(b)). This result follows from the Pythagorean Theorem.
- 3). Any point inside a triangle can be connected to any two of its vertices to form a pair of connected line segments. The length of these line segments is larger than the length of the side that connects the two vertices and is smaller than the length of the other two sides (see Figure A1(c)). This result follows from the second result.

## Theorem: The Shortest Lines between Two Point Sets Do Not Cross

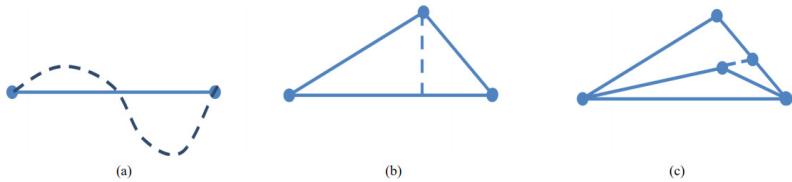
Minimal connecting line segments in any connecting set  $C(n, m)$  do not cross. This is more simply stated as “The shortest lines between two point sets do not cross.”

Proof: Figure A2 illustrates the theorem for a planar domain that is bounded internally by rectangular-shaped closed curves. The connecting line segments are colored and the minimizing line segments are black. The actual proof begins next.

Begin the proof by ordering the points in  $A(n)$  and  $B(m)$  so that the minimal connecting line segments in  $C(n, m)$  are  $\{\mathbf{Q}_i \mathbf{q}_i\}_{i=1}^n$ . The shortest total length is  $L_{\min}(n, m) = \sum_{i=1}^{i=n} |\mathbf{Q}_i \mathbf{q}_i|$ . Next, remove connecting line segment  $\mathbf{Q}_1 \mathbf{q}_1$  from  $C(n, m)$ , remove  $\mathbf{Q}_1$  from  $A(n)$ , and remove  $\mathbf{q}_1$  from  $B(m)$  to produce the connecting set  $C(n - 1, m - 1)$  of  $A(n - 1)$  and  $B(m - 1)$ . It is now shown that  $\{\mathbf{Q}_i \mathbf{q}_i\}_{i=2}^n$  are minimal connecting line segments in  $C(n - 1, m - 1)$ .

In other words, let's show that the following key equation holds:

$$L_{\min}(n, m) = |\mathbf{Q}_1 \mathbf{q}_1| + L_{\min}(n-1, m-1) \quad L_{\min}(n-1, m-1) = \sum_{i=2}^{i=n} |\mathbf{Q}_i \mathbf{q}_i|$$



**Figure A1.** Three well-known results.

To show this, assume that  $\{\mathbf{Q}_i \mathbf{q}_i\}_{i=2}^n$  are not minimal connecting line segments in  $C(n \geq 1, m \geq 1)$ , in which

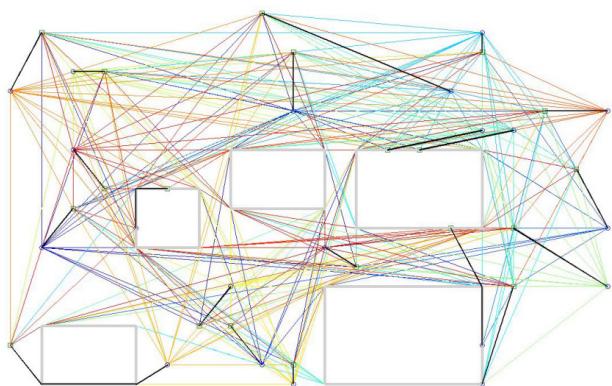
case there exists a total length  $L^*$  that is smaller than  $L_{\min}(n \geq 1, m \geq 1)$ . This leads to the contradictory result:

$L_{\min}(n, m) = |\mathbf{Q}_1 \mathbf{q}_1| + L^* < |\mathbf{Q}_1 \mathbf{q}_1| + L_{\min}(n \geq 1, m \geq 1) = L_{\min}(n, m)$ . Indeed,  $\{\mathbf{Q}_i \mathbf{q}_i\}_{i=2}^n$  are minimal connecting

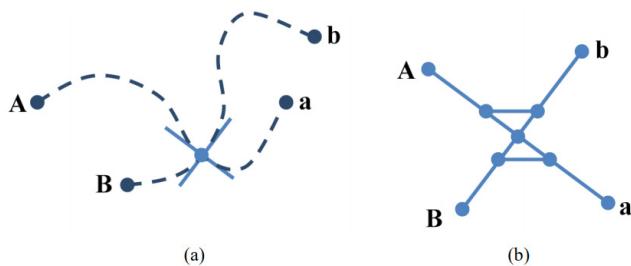
line segments in  $C(n \geq 1, m - 1)$ .

The key equation can be invoked in a sequence that begins with  $n$  connecting line segments, and ends with any two. Furthermore,  $n$  line segments do not cross if any combination of two does not cross. Thus, it remains to prove the theorem for  $n = 2$  connecting line segments. The case of  $n = 1$  line segment should be proven, too; in this case the crossing of the curved line is with itself. Now, consider the case of  $n = 2$ .

Two crossing line segments are shown in Figure A3(a) and a view of the infinitesimal region surrounding the crossing point is shown in Figure A3(b) along with two new short-circuiting line segments. Say that the two straight line segments  $Aa$  and  $Bb$  shown in Figure A3(b) contribute to the total length  $L$  by an infinitesimal amount  $dL$ . As shown, by result (1) in the Lemma, the total length of the short-circuiting line segments can be used to eliminate the crossing point and switch the assignment from  $Aa$  and  $Bb$  to the assignment of  $Ab$  and  $Ba$  while shortening  $dL$  and therefore shortening  $L$ . But  $L$  was the smallest total length, which is contradictory. It follows that the two line segments could not have crossed. The two line segments in Figure A3(a) can be regarded as one line segment, so one line segment could not have crossed on itself, either.



**Figure A2.** Illustration ( $n = m = 19$  case).



**Figure A3.** Crossing line segments.

## REFERENCES

1. Bellingham, J., Tillerson, M., Richards, A. and How, J.P. (2003) Multi-Task Allocation and Path Planning for Cooperative UAVs. In: Butenko, S., Murphey, R. and Pardalos, P.M., Eds., Cooperative Control: Models, Applications and Algorithms, Kluwer Academic Publishers, Boston, 23-39.
2. Cummings, M.L., Bruni, S., Mercier, S. and Mitchell, P.J. (2007) Automation Architecture for Single Operator, Multiple UAV Command and Control. *The International Command and Control Journal*, 1, 1-24.
3. Shima, T. and Rasmussen, S. (Eds.) (2009) UAV Cooperative Decision and Control. Society for Industrial and Applied Mathematics, Philadelphia. <http://dx.doi.org/10.1137/1.9780898718584>
4. Adams, J.A., Humphrey, C.M., Goodrich, M.A., Cooper, J.L., Morse, B.S., Engh, C. and Rasmussen, N. (2009) Cognitive Task Analysis for Developing Unmanned Aerial Vehicle Wilderness Search Support. *Journal of Cognitive Engineering and Decision Making*, 3, 1-26. <http://dx.doi.org/10.1518/155534309X431926>
5. Donmez, B.D., Nehme, C. and Cummings, M.L. (2010) Modeling Workload Impact in Multiple Unmanned Vehicle Supervisory Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 40, 11801190. <http://dx.doi.org/10.1109/TSMCA.2010.2046731>
6. Gardiner, B., Ahmad, W., Cooper, T., Haveard, M., Holt, J. and Biaz, S. (2011) Collision Avoidance Techniques for Unmanned Aerial Vehicles. Auburn University, Auburn.
7. How, J., Ellis King, E. and Kuwata, Y. (2004) Flight Demonstrations of Cooperative Control for UAV Teams. AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit, Chicago, 20-23 September 2004.
8. Edwards, D. and Silverberg, L.M. (2010) Autonomous Soaring: The Montague Cross-Country Challenge. *Journal of Aircraft*, 47, 1763-1769. <http://dx.doi.org/10.2514/1.C000287>
9. Levedahl, B. and Silverberg, L.M. (2005) Autonomous Coordination of Aircraft Formations Using Direct and NearestNeighbor Approaches. *Journal of Aircraft*, 42, 469-477. <http://dx.doi.org/10.2514/1.6868>
10. Tsourdos, A., White, B. and Shanmugavel, M. (2011) Cooperative Path Planning of Unmanned Aerial Vehicles. Wiley, New York.

11. Lozano-Perez, T. and Wesley, M.A. (1979) An Algorithm for Planning Collision Free Paths among Polyhedral Obstacles. *Communications of the ACM*, 22, 560-570. <http://dx.doi.org/10.1145/359156.359164>
12. Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4, 100-107. <http://dx.doi.org/10.1109/TSSC.1968.300136>
13. Kuhn, H.W. (1955) The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2, 8397.
14. Dubins, L.E. (1957) On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79, 497-516. <http://dx.doi.org/10.2307/2372560>
15. Levedahl, B. and Silverberg, L.M. (2009) Control of Underwater Vehicles in Full Unsteady Flow. *IEEE Journal of Oceanic Engineering*, 34, 656-668.



# **CHAPTER**

# **4**

## **Quadcopter Design for Payload Delivery**

---

**Gordon Ononiwu, Ondoma Onojo, Oliver Ozioko, Onyebuchi Nosiri**

Department of Electrical/Electronic Engineering, Federal University of Technology, Owerri, Nigeria

### **ABSTRACT**

This paper presents the design and implementation of a quadcopter capable of payload delivery. A quadcopter is a unique unmanned aerial vehicle which has the capability of vertical take-off and landing. In this design, the quadcopter was controlled wirelessly from a ground control station using

---

**Citation:** Ononiwu, G. , Onojo, O. , Ozioko, O. and Nosiri, O. (2016), "Quadcopter Design for Payload Delivery". *Journal of Computer and Communications*, **4**, 1-12. doi: 10.4236/jcc.2016.410001.

**Copyright:** © 2016 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

radio frequency. It was modeled mathematically considering its attitude and altitude, and a simulation carried out in MATLAB by designing a proportional Integral Derivative (PID) controller was applied to a mathematical model. The PID controller parameters were then applied to the real system. Finally, the output of the simulation and the prototype were compared both in the presence and absence of disturbances. The results showed that the quadcopter was stable and able to compensate for the external disturbances.

**Keywords:** Quadcopter, PID, Controller, Payload, Delivery, Unmanned, Aerial, Vehicle

## INTRODUCTION

Quadcopters are unique unmanned aerial vehicles which have a vertical take-off and landing ability [1]. In the past, it was primarily used in the military, where it was deployed in hostile territory, to reduce pilot losses. The recent reduction in cost and feature size of semi-conductor logic has led to the application of quadcopters in many other areas like weather monitoring, forest fire detection, traffic control, cargo transport, emergency search and rescue, communication relaying, etc. Quadcopters operate in fixed rotor propulsion mode, an advantage over the traditional helicopter. They have four rotors which are arranged in such a way that rotors on transverse ends rotate in the same direction, while the other two operate in the opposite direction [1]. Changing the parameters of the attitude (roll, pitch and yaw), and altitude determines the behavior and position of the quadcopter. The throttle on each rotor changes its attitude. The UAVs can be broadly classified into two categories: fixed wing versus rotary wing, each with their own strengths and weaknesses. For instance, the fixed-wing UAVs usually have high speed, and heavy payload, but they must maintain a continuous forward motion to remain in the air [2].

The applications for quadcopters have recently grown significantly, and related works are presented here. [3] presented an altitude controller design for UAVs in real time applications. The author described a controller design method for the hovering control of UAVs and automatic vertical take-off systems. In 2012, the attitude and altitude dynamics of an outdoor quadrotor were controlled using two different structures of proportional integral derivative (PID) controllers [4]. The tests showed that the two structures were successful. [5] presents a Quadcopter for surveillance monitoring. In [6], the design and control of VertiKUL, a Vertical Take-Off and Landing (VTOL) transitioning tail-sitter Unmanned Aerial Vehicle (UAV) was

presented. The UAV is capable of hover flight and forward flight, and its main application is for parcel delivery.

## METHODOLOGY

In this work, aquadcopter was first modeled mathematically, after which a simulation was carried out in MATLAB by designing a PID controller, which was applied to the mathematical model. The PID controller parameters were then applied to the real system. Finally, the output of the simulation and the real system were compared. In this paper, attitude and altitude dynamics of a quadcopter were considered. Therefore, roll, pitch, yaw, and altitude dynamics were modelled.

### Mathematical Model of Attitude Dynamics

One of the aims of the work is to stabilize the quadcopter in the hovering condition. The following assumptions were made [4]-[7]: i) System structure is supposed to be rigid and symmetrical. ii) Earth fixed reference frame is assumed to be inertial. iii)The rotors are rigid, i.e. no blade flapping occurs.

Attitude of the quadcopter is defined by roll, pitch and yaw angles; namely  $\phi$ ,  $\theta$ , and  $\psi$ , respectively. Angular velocity components in body reference frame are  $p$ ,  $q$ , and  $r$ , respectively.  $F_1$ ,  $F_2$ ,  $F_3$ , and  $F_4$  are the thrust forces from the four motors, which are generated by propellers. Axes and states are represented in Figure 1. It is possible to obtain a rotation matrix with the following equation:

$$R = R_x R_y R_z \quad (2.1)$$

where  $R$  is the matrix transformation dynamics of the model quadcopter moving from landing position to a fixed position in space.

Therefore, substituting  $R_x$ ,  $R_y$ , and  $R_z$  into Equation (2.1), we get the system matrix:

$$R_{xyz} = \begin{bmatrix} \cos\psi \cos\theta & \cos\psi \sin\theta \sin\phi - \sin\psi \sin\phi & \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \\ \sin\psi \sin\theta & \sin\psi \sin\theta \sin\phi + \cos\psi \cos\phi & \sin\psi \sin\theta \cos\phi - \sin\psi \cos\phi \\ -\sin\theta & \sin\phi \cos\theta & \cos\theta \cos\phi \end{bmatrix}. \quad (2.2)$$

The angular velocity of the quadcopter can be written as:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (2.3)$$

It is possible to write state equations of the Euler angles in terms of angular velocity components, p, q, and r by using rotation matrices.

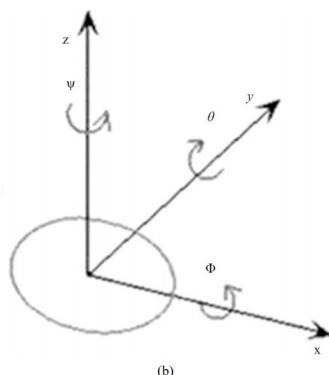
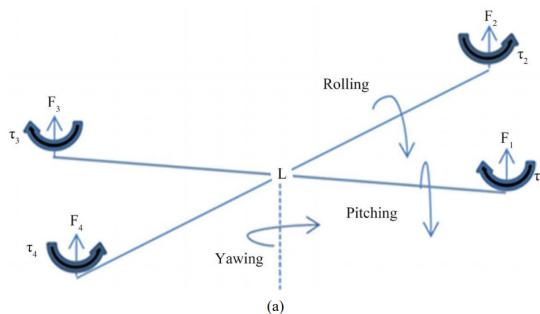
$$\psi = p + q \tan \theta \sin \varphi + r \tan \theta \cos \varphi \quad (2.4)$$

$$\theta = q \cos \varphi - r \sin \varphi \quad (2.5)$$

$$\psi = p + q \sec \varphi + r \sec \theta \cos \varphi. \quad (2.6)$$

Inertia matrix can be written below as:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}. \quad (2.7)$$



**Figure 1.** Schematic representation of the system.

Inertia matrix,  $I$  is assumed to be a diagonal matrix (i.e. the mass products of inertia terms are assumed to be 0). Thrust forces generated by each motor-propeller pair are denoted by  $F_1$ ,  $F_2$ ,  $F_3$  and  $F_4$ . Therefore, resultant moments about roll, pitch and yaw axes are given below:

$$\sum M_x = (F_3 - F_1) \frac{L}{2} \quad (2.8)$$

$$\sum M_y = (F_2 - F_4) \frac{L}{2} \quad (2.9)$$

$$\sum M_z = (M_1 + M_3) - (M_2 + M_4). \quad (2.10)$$

Therefore,

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{qr}{I_x}(I_y - I_z) - \frac{L}{2I_x}(F_1 - F_3) \\ \frac{pr}{I_y}(I_y - I_z) - \frac{L}{2I_z}(F_2 - F_4) \\ \frac{pq}{I_z}(I_x - I_y) \frac{((F_2 + F_4) - (F_1 + F_3))d}{I_z b} \end{bmatrix}. \quad (2.11)$$

Hence, state vector and state equations considering the yaw, roll and the pitch dynamics of the platform are expressed below:

$$x = \begin{bmatrix} \varphi \\ \theta \\ \psi \\ p \\ q \\ r \end{bmatrix} \quad (2.12)$$

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} p + q \tan \theta \sin \varphi + r \tan \theta \cos \varphi \\ q \cos \varphi - r \sin \varphi \\ p + q \sec \varphi + r \sec \theta \cos \varphi \\ \frac{qr}{I_x}(I_y - I_z) - \frac{L}{2I_x}(F_1 - F_3) \\ \frac{pr}{I_y}(I_y - I_z) - \frac{L}{2I_z}(F_2 - F_4) \\ \frac{pq}{I_z}(I_x - I_y) \frac{((F_2 + F_4) - (F_1 + F_3))d}{I_z b} \end{bmatrix}. \quad (2.13)$$

Also, by applying the force and moment balance laws, we have the following quadcopter motion formulation:

$$\ddot{x} = u_1 (\cos \phi \sin \theta \cos \psi + \sin \iota \sin \psi) - \frac{K_1 \dot{x}}{m} \quad (2.14)$$

$$\ddot{y} = u_1 (\sin \phi \sin \theta \cos \psi + \cos \phi \sin \psi) - \frac{K_2 \dot{y}}{m} \quad (2.15)$$

$$\ddot{z} = u_1 (\cos f \cos \psi) - g - \frac{K_3 \dot{z}}{m} \quad (2.16)$$

where  $K_i$  is the drag coefficient (assume zero since drag is negligible at low speed).

The angle movement of Quadcopter is given by

$$\phi_d = \tan^{-1} \frac{y_d - y}{x_d - x} \quad (2.17)$$

$$\psi_d = \tan^{-1} \frac{z_d - z}{\sqrt{(x_d - x)^2 + (y_d - y)^2}}. \quad (2.18)$$

## Controller Input

Quadcopter has four controller input  $U_1$ ,  $U_2$ ,  $U_3$ , and  $U_4$ .

$$U_1 = (Th_1 + Th_2 + Th_3 + Th_4)/m \quad (2.19)$$

$$U_2 = (-Th_1 - Th_2 + Th_3 + Th_4)/I_1 \quad (2.20)$$

$$U_3 = l(-Th_1 + Th_2 + Th_3 - Th_4)/I_2 \quad (2.21)$$

$$U_4 = l(Th_1 + Th_2 + Th_3 + Th_4)/I_3 \quad (2.22)$$

where,

$Th_i$  = thrust generated by each motor

$l$  = force to moment scaling factor

$I_i$  = moment of inertia with respect to the axes.

The second derivative of Euler angles are:

$$\ddot{\theta} = U_2 - lk_4 \dot{\theta}/I_1 \quad (2.23)$$

$$\ddot{\psi} = U_3 - lk_5 \dot{\theta}/I_2 \quad (2.24)$$

$$\ddot{\Phi} = U_1 - lk_6 \dot{\theta}/I_3. \quad (2.25)$$

## **Motor and Propeller Model**

Brushless motors are used as the actuators of propellers of the quadcopter. Four brushless motors in the system are driven via Electronic Speed Controllers (ESC). ESC's converts PWM signals into a three-phased signal, which rotates the motor continuously. Motor and propeller unit models are identified experimentally. These models are algebraic, steady state ones. Thrust force generated due to the rotation of propellers can be calculated with the equation given below:

$$F_z = b\Omega^2 \quad (2.26)$$

where b is thrust factor which is determined experimentally.

## **Mathematical Model of Altitude Dynamics**

Total thrust force and weights are the acting forces in z direction. Translational acceleration of the quadcopter in z-axis can be obtained, using the following formula;

$$\ddot{z} = \frac{\sum_{i=1}^4 F_i}{m} \cos \theta \cos \varphi - g . \quad (2.27)$$

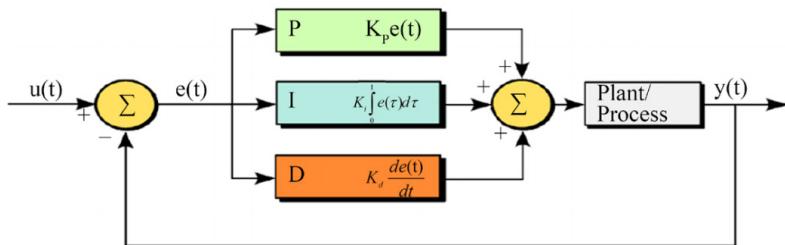
This equation reveals position in z-axis and velocity in z-axis as two states.

## **The PID Controller Design**

The PID is a popular way of controlling the quadcopter, and one widely used strategy to control a quadcopter with a PID is to create an individual closed-loop system for each independent axis of each sensor. That is, we would have the same structure depicted in Figure 2 for each sensor axis where  $y(t)$  would be the sensor output (angular velocity or acceleration of each axis) and  $u(t)$  the desired value to that axis. This way we would have three different PID controllers for the gyroscope (one for each axis) and the input to each control loop would be an angular velocity reference. That is, making all references equal to zero would make the quadcopter preserve all its rotation angles. Other three distinct PIDs would then be used to control the accelerometer axes. Therefore, setting zero to x and z axes would make the quadcopter hover (all weight force would be concentrated in y axis).

Another widely used strategy is to mix the accelerometer and gyroscope signals to find a single angle value for each axis. That is, combining the gyroscope information with the accelerometer information, one can find a

unique angle value for each axis that can represent an absolute angle of rotation or a variation with respect to the previous time step. Therefore, one can use a total of only three closed-loop systems instead of six. Also, there are three multiple ways of combining those signals, and the control system efficiency will directly depend on which way is chosen. Since the sensors can have errors (e.g. noise and drift), a method that minimizes the main problems of each sensor is chosen and the best features of each can be combined.



**Figure 2.** A typical PID controller internal blocks.

Therefore, the control signals generated by each closed-loop structure must be combined before sending to the motors. A commonly adopted way of combining those signals is given below:

$$m_1 = T + u_p + u_y \quad (2.28)$$

$$m_2 = T + u_r - u_y \quad (2.29)$$

$$m_3 = T - u_p + u_y \quad (2.30)$$

$$m_4 = T - u_r - u_y \quad (2.31)$$

where,  $m_i$  = signal sent to the  $i$ th motor,  $T$  = throttle signal and  $u_p, u_r, u_y$  = control signals (for pitch, roll and yaw axes respectively)

It can be seen in the equations above that those signals are used in a differential mode due to the plant symmetry. Thus, another important detail is that the signals of each term inside each equation depends on each axis reference (e.g. whether positive pitch rotation is clockwise or counter-clockwise) as well as where each motor is positioned. So one must know the motor configuration (whether plus + or cross ×) before writing the equations. Finally, each motor will be individually controlled in an open-loop configuration which can already lead us to a flyable quadcopter.

## SIMULATION

The quadcopter design details were used to produce a model to be constructed in the simulation window. Some effects (like aerodynamic effects such as blade flapping) were ignored or dramatically simplified, thus, a limitation to the model. The system model contains five major blocks (systems) each containing its own subsystem or unit blocks that describe the overall system dynamics of our quadcopter. The system blocks include: i) Path Command block; ii) Position Controller block; iii) Attitude Controller block; iv) Quadcopter Control mixing block; v) Quadcopter Dynamics block.

### Path Command and Position Controller

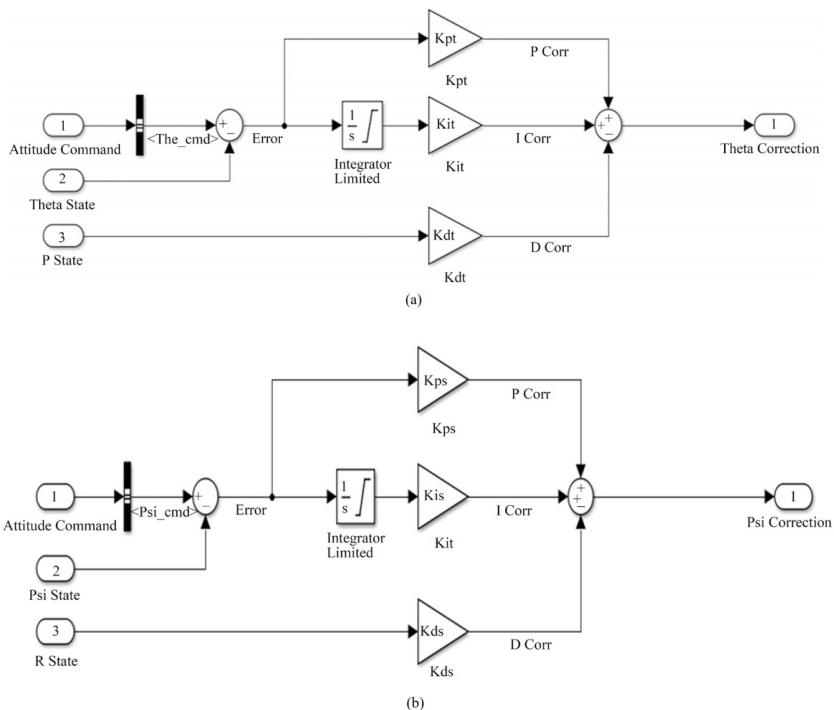
The path command block is used to set the desired position of quadcopter in terms of x, y and z coordinates as well as the yaw (for directioning on the x-y plane). These parameters serve as the set points for the position controller. These variables are loaded as Matlab codes into the simulation through the MATLAB workspace. The position controller takes inputs from the Path command as its set point, and also receives the current position (state) of the quadcopter as feedback. It then compares this feedback with the set point to give the error.

The position error in the body frame is computed, and maps the body frame desired velocity. This desired velocity is then mapped to a desired attitude. The essence of this rotation is because the Inertial-frame velocity is used as the state to be controlled (Path command input).

### The Attitude Controller

The attitude controller, like the position controller, takes feedback from the output of the system as one of its inputs, but takes its second input from the position controller. This block contains the full proportional integral derivative (PID) controller which does the final attitude correction. The feedback to this block contains the current attitude, and angular velocity of the quadcopter in x, y and z axes. The controller essentially compares the current state with the set point, and attempts to bring these various parameters to the set point. The integral part of the PID ensures that the parameters remain at set point once achieved.

Figure 3(a) and Figure 3(b) are the Simulink blocks for Pitch and Yaw, including their corresponding attitude correction outputs.



**Figure 3.** (a) Simulink block for the pitch (Theta) correction; (b) Simulink block for the yaw (Psi) correction.

## Quadcopter Control Mixing

There are two quadcopter configurations available, the quad + configuration and the quad  $\times$  configuration. The quadcopter control mixing block does the control signal mixing depending on the quadcopter configuration used (plus + or cross  $\times$ ). It gets the corrected parameters for Pitch ( $\theta$ ), Roll ( $\psi$ ), Yaw ( $\phi$ ) and Altitude ( $Z$ ) from the attitude controller and generates the necessary throttle commands to actuate each motor ( $m_1, m_2, m_3, m_4$ ).

$$mc_1 = z_c - \theta_c - \phi_c - \psi_c \quad (3.1)$$

$$mc_2 = z_c - \theta_c + \phi_c + \psi_c \quad (3.2)$$

$$mc_3 = z_c + \theta_c + \phi_c - \psi_c \quad (3.3)$$

$$mc_4 = z_c + \theta_c - \phi_c + \psi_c \quad (3.4)$$

## Quadcopter Dynamics

The quadcopter dynamics block contains two major subsystem blocks: The motor dynamics block, and The State Equations block. The Primary function of the Motor dynamics block is to convert the percentage throttle command to the motor RPM.

The motor dynamics block contains the state space model;

$$\dot{x} = Ax + Bu \quad (3.5)$$

$$y = Cx + Du \quad (3.6)$$

where,  $\dot{x} = dx/dt$  (velocity),  $A = -1/\text{quadmodel.T}$

$$B = 1, \quad C = 1/\text{quadmodel.T}$$

$u$  = control input.

The four RPM values generated are fed into the State equations block which contains a level two S-function block, where the flight dynamic state equations are loaded. This S-function is written in MATLAB. The S-function block holds the initial conditions (Table 1) used for the simulation. Its parameters are the quadcopter model functions and the input initial conditions. The outputs,  $(p, q, r, \phi, \theta, \psi, u, v, w, x, y, z)$  describes the physical behavior of the quadcopter.

The main function of the S-function block is to use various parameters fed into it, to calculate the quadcopter outputs (Angular velocities, linear velocities, position (along the x, y and z axes) and attitude (yaw, pitch and roll)) which are then used for the simulation plots and 3D animation. Also, a disturbance block was added as input to the S-function block to simulate for response to external forces which might act on the quadcopter body during flight and how it can compensate and recover from these disturbances whilst still trying to maintain flight stability (set point).

The data used for the quadcopter simulation was based on the design model presented. The individual parameters of the components that make up the quadcopter were carefully calculated. Some parameters such as Torque, Thrust and Time Constants where estimated from already measured data of a similar system. This estimation is due to the unavailability of necessary test bench (and instruments) required to carefully monitor and measure these parameters. Since our design model is a quadcopter for payload delivery, our simulation focused on the following: 1) Load test (maximum allowable load on-board); 2) Hover test (with load and without load); 3) Disturbance test (with load and without load).

The PID Control gains were carefully tuned and the appropriate gain value was chosen based on the simulation results obtained. The PID gains used in the Attitude controller are presented in Table 1 while the PD gains of the position controller are presented in Table 2.

## SIMULATION RESULTS AND DISCUSSION

The simulation was run for 60 seconds and several graphs were generated. Figure 4 shows the result of the simulation with no external disturbance added. The Path command was such that the quadcopter will move a diamond-shaped path. P, Q and R are angular velocities about the X, Y and Z axes respectively; Phi, theta and Psi are the Roll, Pitch and Yaw angles;

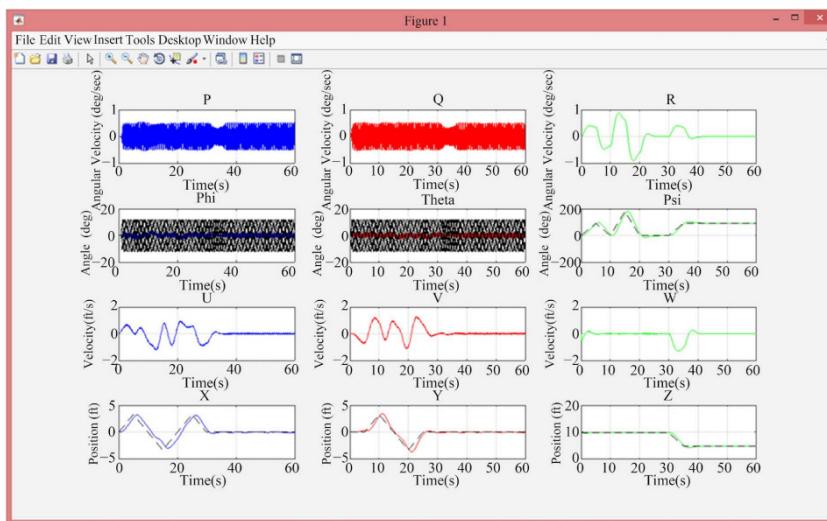
U, V and W are the translational velocities along the X, Y and Z axes respectively; X, Y, and Z define the position of the quadcopter along the X, Y and Z axes respectively.

**Table 1.** Attitude controller PID gains.

| PARAMETERS   | VALUE                                       |
|--|---|
| Gross mass, $m$  | 1.5 kg                                      |
| Time constant  | 0.076 s                                     |
| Minimum throttle   | 5%  |
| Thrust coefficient, $c_t$                                | $1.4865 \times 10^{-7}$ N/RPM <sup>2</sup>  |
| Torque coefficient, $c_q$                                | $2.925 \times 10^{-9}$ N·m/RPM <sup>2</sup> |
| Percentage throttle to RPM conversion coefficient, $c_r$ | 80.584 RPM/%                                |
| Mass Moment of inertia across the x-axis, $J_x$          | 0.0085136 kg·m <sup>2</sup>                 |
| Mass Moment of inertia across the y-axis, $J_y$          | 0.0085136 kg·m <sup>2</sup>                 |
| Mass Moment of inertia across the z-axis, $J_z$          | 0.016334 kg·m <sup>2</sup>                  |

**Table 2.** The position controller PD gains.

| Parameter                  | Proportional gain, $k_p$ | Differential gain, $k_d$ |
|----------------------------|--------------------------|--------------------------|
| Pitch ( $\theta$ ) command | 3.5                      | 2.9                      |
| Roll ( $\phi$ ) command    | 3.5                      | 2.9                      |



**Figure 4.** Result of the simulation without disturbance.

## Hover Test

The hover simulation of a quadcopter in flight describes its ability to float at/about a fixed point in air. Hover test was done at 10 feet and the results are given below.

The graphs shown in Figure 4 represent a stable system. At first, there was an overshoot, but with time the controller sets each of these parameters to set point.

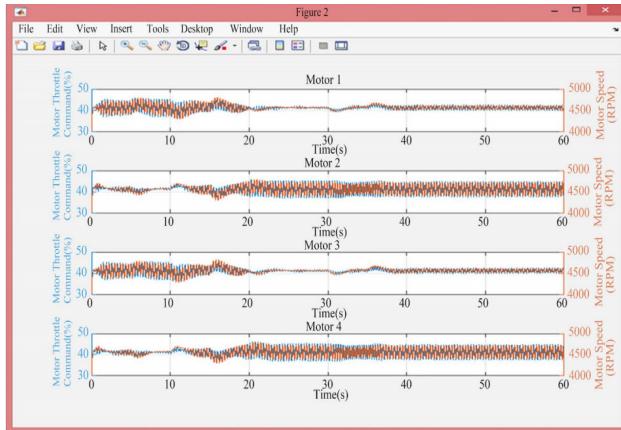
## Disturbance Test

Disturbance was introduced to the quadcopter flight simulation which could represent the action of external force like wind on the body of the quadcopter while in flight. The graph in Figure 5 represents the system without any disturbance. However, as can be seen from the graph in Figure 6, when a disturbance was introduced, the quadcopter was able to compensate properly. Therefore, the PID controller logic is functional and it can be implemented for the actual system.

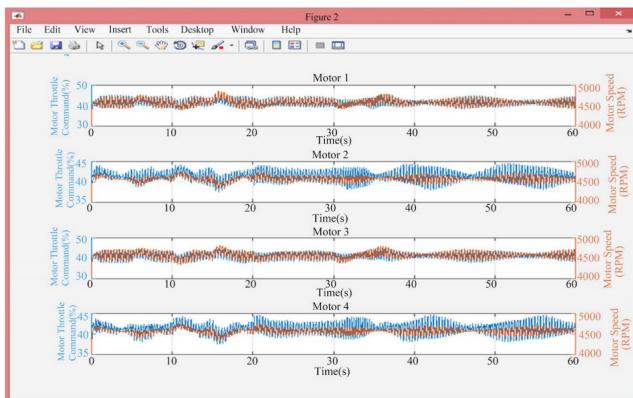
The effect of a disturbance to the flight path was further tested with regards to linear and angular velocity is presented in Figure 7. The system acknowledged the disturbance but was able to compensate adequately, and return to a stable flight.

## Load Test

Since the quadcopter design model is for payload delivery, the simulation shows how an increase in the gross mass of the quadcopter affects its motor speed. The result is presented in Figure 8 and Figure 9. Again, the system was able to maintain a stable flight.



**Figure 5.** Motor throttle vs motor speed.

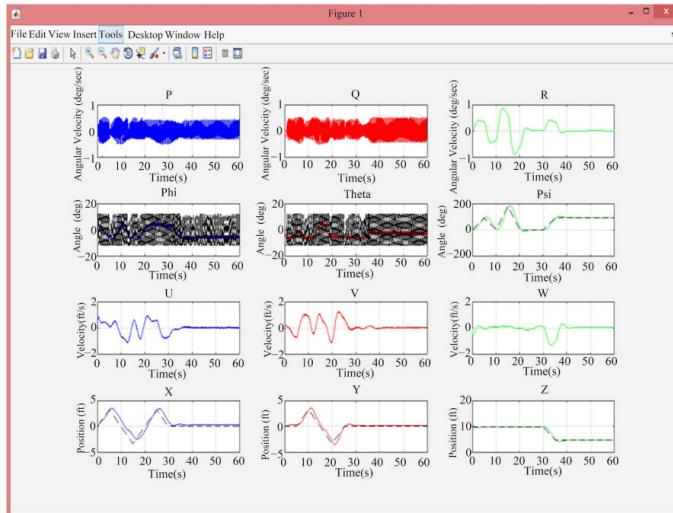


**Figure 6.** Motor throttle vs motor speed (disturbance test).

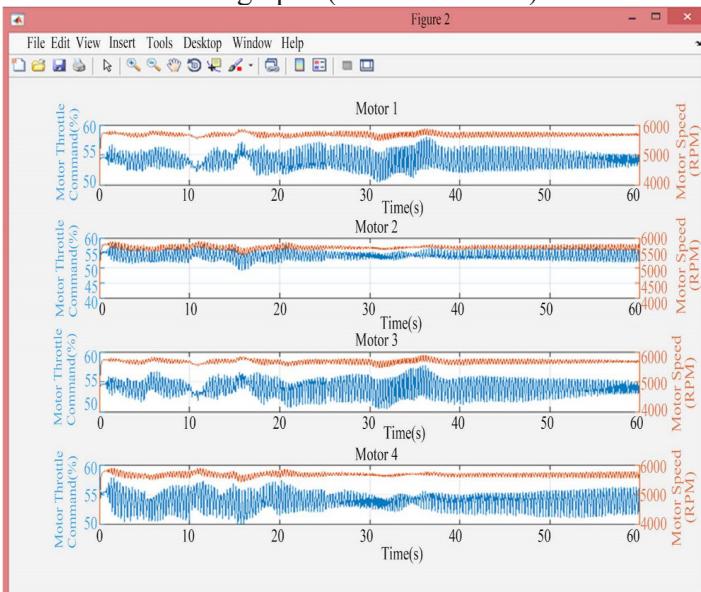
## CONCLUSION AND FUTURE WORK

A stable 1.26 kg quadcopter for payload delivery was designed, simulated and implemented. The design and simulation, which utilized a PID controller, was able to yield results that showed good compensation in the presence of

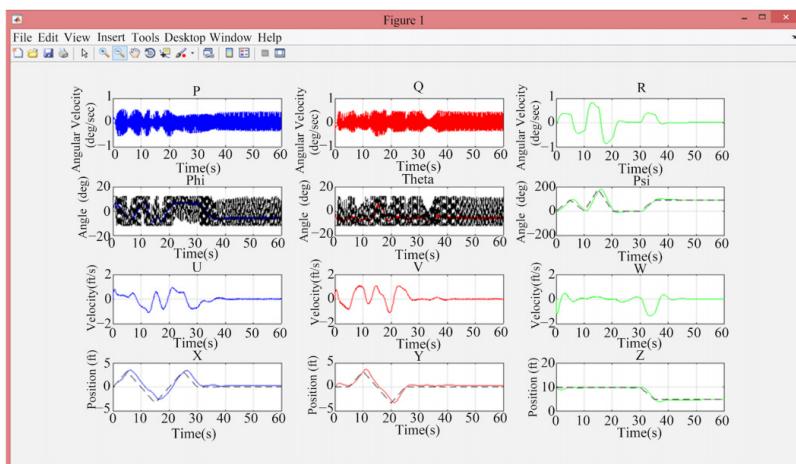
external disturbances. The Quad X structural model was used to enable the Quadcopter to gain more speed over a short period of operation, as timely delivery of payload is an important factor. Future study would be to equip the quadcopter with a solar powered system, and to encrypt the communication between the quadcopter and the ground station.



**Figure 7.** Simulation result graphs (disturbance test).



**Figure 8.** Motor throttle vs motor speed (load test).



**Figure 9.** Simulation result graphs (load test).

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

## REFERENCES

1. Gupte, S., Mohandas, P.I.T. and Conrad, J.M. (2012) A Survey of Quadrotor Unmanned Aerial Vehicles. 2012 Proceedings of IEEE, Orlando, FL, 15-18 March 2012, 1-6. <http://dx.doi.org/10.1109/secon.2012.6196930>
2. Zeng, Y., Zhang, R. and Lim, T.J. (2016) Wireless Communications with Unmanned Aerial Vehicles: Opportunities and Challenges.
3. Mustapa, M.Z. (2015) Altitude Controller Design for Quadcopter UAV. Jurnal Teknologi.
4. Guclu, A. (2012) Attitude and Altitude Control of an Outdoor Quadrotor. The Graduate School of Natural and Applied Science, Atilim University, 76.
5. Ononiwu, G.C., Onojo, O.J., Chukwuchekwa, N. and Isu, G.O. (2015) UAV Design for Security Monitoring. International Journal of Emerging Technology & Research, 2, 16-24.
6. Hochstenbach, M. and Notteboom, C. (2015) Design and Control of an Unmanned Aerial Vehicle for Autonomous Parcel Delivery with Transition from Vertical Take-off to Forward Flight. International Journal of Micro Air Vehicles, 7, 395-405. <http://dx.doi.org/10.1260/1756-8293.7.4.395>
7. Min, B.-C., Hong, J.-H. and Matson, E.T. (2011) Adaptive Robust Control (ARC) for an Altitude Control of a Quadrotor Type UAV Carrying an Unknown Payloads. 2011 11th International Conference on Control, Automation and Systems, Korea, 26-29 October 2011, 1147-1151.



# **SECTION 2**

# **TRAJECTORY CONTROL TECHNIQUES**



# **CHAPTER**

# **5**

## **Advanced UAV Trajectory Generation: Planning and Guidance**

---

**Antonio Barrientos, Pedro Gutiérrez and Julián Colorado**

Universidad Politécnica de Madrid – (Robotics and Cybernetics Group),  
Spain

### **INTRODUCTION**

As technology and legislation move forward (JAA & Eurocontrol, 2004) remotely controlled, semi-autonomous or autonomous Unmanned Aerial Systems (UAS) will play a significant role in providing services and enhancing safety and security of the military and civilian community at large (e.g. surveillance and monitoring) (Coifman et al., 2004). The potential

---

**Citation:** Antonio Barrientos, Pedro Gutierrez and Julian Colorado (January 1st 2009). “Advanced UAV Trajectory Generation: Planning and Guidance”, Aerial Vehicles, Thanh Mung Lam, IntechOpen, DOI: 10.5772/6467.

**Copyright:** © 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License

market for UAVs is, however, much bigger than just surveillance. UAVs are ideal for risk assessment and neutralization in dangerous areas such as war zones and regions stricken by disaster, including volcanic eruptions, wildfires, floods, and even terrorist acts. As they become more autonomous, UAVs will take on additional roles, such as air-to-air combat and even planetary science exploration (Held et al., 2005).

As the operational capabilities of UAVs are developed there is a perceived need for a significant increase in their level of autonomy, performance, reliability and integration with a controlled airspace full of manned vehicles (military and civilian). As a consequence researchers working with advanced UAVs have moved their focus from system modeling and low-level control to mission planning, supervision and collision avoidance, going from vehicle constraints to mission constraints (Barrientos et al., 2006). This mission-based approach is most useful for commercial applications where the vehicle must accomplish tasks with a high level of performance and maneuverability. These tasks require flexible and powerful trajectory-generation and guidance capabilities, features lacking in many of the current commercial UAS. For this reason, the purpose of this work is to extend the capabilities of commercially available autopilots for UAVs. Civil systems typically use basic trajectory-generation algorithms, capable only of linear waypoint navigation (Rysdyk, 2003), with a minimum or non-existent control over the trajectory. These systems are highly constrained when maneuverability is a mission requirement. On the other hand, military researchers have developed algorithms for high-performance 3D path planning and obstacle avoidance (Price, 2006), but these are highly proprietary technologies that operate with different mission constraints (target acquisition, threat avoidance and situational awareness) so they cannot be used in civil scenarios.

This chapter presents a robust Trajectory Generation and Guidance Module ( $TG^2M$ ), a software tool capable of generating complex six-degrees-of-freedom trajectories in 3D space with velocity, acceleration, orientation and time constraints. The  $TG^2M$  is an extension module to the Aerial Vehicle Control Language (AVCL), a software architecture and interpreted language specification that address the issues of mission definition, testing, validation and supervision for UAVs (Barrientos et al., 2006). The AVCL platform incorporates a 3D visual simulator environment that uses a Geographic Information System (GIS) as the world's data model. The GIS backend means all objects are geo-referenced and that several official and commercial databases may be used for mission planning (roads, airports,

power lines, crop fields, etc.). The language specification contains a wide set of instructions that allow the off/on-line supervision and the creation of vehicle-independent missions.

The TG<sup>2</sup>M is designed to model 3D cartesian paths with parametric constraints. The system uses two types of mathematical descriptors for trajectories: analytical functions and polynomial interpolation. The two main contributions of the module are its geometrical representation of the trajectory and its parametric definition. Simple maneuvers like lines and circumference arcs are created with analytical functions that constrain the geometry of the desired path; then the parametric constraints are applied. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. More complex maneuvers are described with polynomial interpolation and fitted to the critical control path points, meeting desired position, time and velocity constraints. These polynomial functions are based on third and fourth order splines with fixed boundary conditions (for example initial and final velocities), which join all control points with a continuous and smooth path (Jaramillo-Botero et al., 2004).

The section 2 of this chapter presents some of the current techniques extracted from the military aerial survey applications, showing complex mission-planning tools capable of addressing the mission-specific constraints required. Within those approaches, this section also introduces a brief description of the AVCL architecture, describing its components, modules, and the main features provisioned, addressing a novel way of human-mission planning definition and testing. The section 3 introduces the AVCL built-in TG<sup>2</sup>M framework, describing the different techniques used for the trajectory planning and guidance of UAVs, as well as the mathematical treatment of these methods (analytical functions and polynomial interpolation). On the other hand, simulation-based results (see section 4) using a mini-helicopter simulator embedded into the AVCL environment will show the capabilities of the TG<sup>2</sup>M while flying aggressive and simple maneuvers. Last but not least, “final observations” in section 5 includes comments about the TG<sup>2</sup>M framework and the upcoming additions to the TG<sup>2</sup>M under current development.

## MOTION-PLANNING METHODOLOGIES

A planning algorithm should provide feasible and flyable optimal trajectories that connect starting with target points, which should be compared and

valued using specific criteria. These criteria are generally connected to the following major concerns, which arise during a plan generation procedure: feasibility and optimality. The first concern asks for the production of a plan to safely “move” the UAV to its target state, without taking into account the quality of the produced plan. The second concern asks for the production of optimal, yet feasible, paths, with optimality defined in various ways according to the problem under consideration (LaValle, 2006). Even in simple problems searching for optimality is not a trivial task and in most cases results in excessive computation time, not always available in real-world applications. Therefore, in most cases we search for suboptimal or just feasible solutions. The simplest way to model an UAV path is by using straight-line segments that connect a number of waypoints, either in 2D or 3D space (Moitra et al., 2003, Zheng et al., 2005). This approach takes into account the fact that in typical UAV missions the shortest paths tend to resemble straight lines that connect waypoints with starting and target points and the vertices of obstacle polygons. Although waypoints can be efficiently used for navigating a flying vehicle, straight-line segments connecting the corresponding waypoints cannot efficiently represent the real path that will be followed by the vehicle due to the own kinematics of the traced path. As a result, these simplified paths cannot be used for an accurate simulation of the movement of the UAV in an optimization procedure, unless a large number of waypoints is adopted. In that case the number of design variables in the optimization procedure explodes, along with the computation time. This is why this section presents some background based on the state-of-the-art mission/path planning for UAVs. The purpose (apart from the state-of-the-art survey) is to compare to the AVCL architecture in order to observe how its TG2M embedded framework (see section 3 for details) is a complement of some lacking approaches found in this specialized literature.

## Background

Researchers at top research centers and universities (e.g. JPL at Caltech, Carnegie Mellon, NASA, ESA, among others) are dealing with the development of new strategies that allow high-level mission UAV planning within the desired flight performance. As examples of these approaches we present the NASA Ames Research Center, the Georgia Institute of Technology, and the University of Illinois.

For the past decade NASA has focused on developing an efficient and robust solution to Obstacle Field Navigation (OFN), allowing a fast planning of smooth and flyable paths around both known and unknown obstacles

(Herwitz, 2007). Evolutionary algorithms have been used as a viable candidate to solve path-planning problems effectively, providing feasible solutions within a short time. Given a number of UAVs that are launched from different and known initial locations, the algorithm must create 2-D trajectories with a smooth velocity distribution along each trajectory and with the goal of reaching a predetermined target location, while ensuring collision avoidance and satisfying specific route and coordination constraints and objectives. B-Splines curves are used in order to model both the 2-D trajectories and the velocity distribution along each flight path. A flying laboratory for autonomous systems, based on the Yamaha RMAX helicopter is being developed, which it incorporates the OFN planner and one all-digital camera system with a state-of-the-art tracking and passive ranging capabilities. Machine stereo-vision is being used to determine safe landing areas and monocular vision is used to track the landing location without access to GPS. A Ground Control Station (GCS) is being integrated into the simulation environment to investigate the issues related to high altitude and long endurance flights.

As an alternative approach other researchers do not focus on generating complex trajectories profiles; their aim is to develop robust mission management, capable of successfully integrating the available onboard hardware (e.g. cameras, sensors, etc). The Georgia Tech's UAV Lab developed the GTMax architecture (Alison et al., 2003) based on a Yamaha R-Max mini-helicopter, is an example of this alternative approach. The GTMax system is capable of fully autonomous flight with decentralized software modules for trajectory generation, offline simulation, supervision, guidance and control. The trajectory module generates lineal waypoints between targets with initial and final velocity parameterization. Commands to the helicopter take the form of different types of waypoints. All trajectories generated are assumed to be physically feasible by the helicopter, the kinematic model used for the trajectory generation uses specifiable limits on the maximum speed and acceleration the aircraft may have during a maneuver. From a high-level mission management perspective the GTMax architecture allows the UAV to autonomously locate targets with an identifying symbol within a designated search area. Once the correct target is identified the mission coordination and flight path generation is done at a centralized location on the ground, and the commands are transmitted to the UAV via a wireless datalink. The GTMax GCS interfaces with the primary flight computer and displays vehicle status, the object tracker information and the flight plans generated by the mission planner. The Vision Monitoring Station receives

streaming video from the camera and the output of the image processor. This allows the operator to monitor the efficiency of the image processing as well as visually document the results of the search in the final phase of the mission.

For other projects the main goal is the unification of high-level mission commands and development of generic languages that support online UAV mission planning without constraining the system to a single vehicle. The University of Illinois at Urbana-Champaign (Frazzoli, 2002) presents a new framework for UAV motion planning and coordination. This framework is based on the definition of a modelling language for UAV trajectories, based on the interconnection of a finite number of suitably defined motion primitives, or maneuvers. Once the language is defined, algorithms for the efficient computation of feasible and optimal motion plans are established, allowing the UAV to fulfill the desired path.

If we analyze the UAS described above, almost all of them share one important limitation: their software architecture is tightly coupled to one vehicle and the capabilities of its lowlevel controller. Civil applications require open and extendable software architectures capable of talking to vehicles from different suppliers. The AVCL addresses those limitations, allowing to model different vehicles into a single common language (vehicleindependent missions). In the same fashion the described vehicles show that complex and simple maneuvers could be a suitable solution depending on the kind of mission to fulfill. For this reason the AVCL is extended with the TG2M framework, capable of generating simple and complex 3D paths with the necessary vehicle constrains. The next sub-section introduces the AVCL architecture and some of the features provisioned.

## The Aerial Vehicle Control Language - AVCL

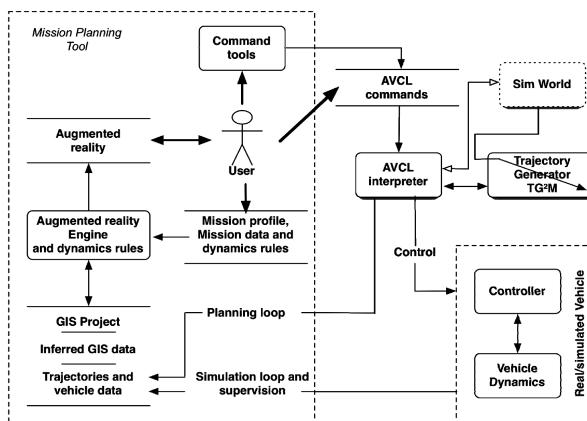
The AVCL is not just a language capable of describing the missions and capabilities of an heterogeneous group of vehicles, it is part of a bigger framework that includes its interpreter, a definition of a base-vehicle, and a Mission Planner that uses GIS as the datamodel for the world (Barrientos et al., 2006). The Mission Planner (MP) is not tied to a particular set of vehicles, sensors or commands. At any given time new functionality can be loaded and displayed to the human operator as new options and commands. This means that the MP tool is to be extended through Vehicle and Command Libraries without recompiling, and those new capabilities and better vehicles can be added easily. The Mission Planner is a great tool for simulation and direct

comparison of various trajectory trackers, UAV models and controllers, because it can display N missions at the same time.

When considered just as a language the AVCL concept is the abstraction layer that allows the human supervisor to create missions that are vehicle and payload independent, promoting code reuse. At the same time the AVCL statements and commands hide device specific software/hardware, and serve as mission definition and storage. As an example of the code used to define operations within a mission:

```
uav.Sensors(0) = parser.loadObject ('camera.lib')
uav.Sensors(0).LookAt (p1)
uav.Sensors(1) = parser.loadObject ('laser.lib')
uav.Sensors(1).TurnOn()
uav.doLine (way_points = {p1, p2, p3, p4}, vel = 0.9 m_s)
```

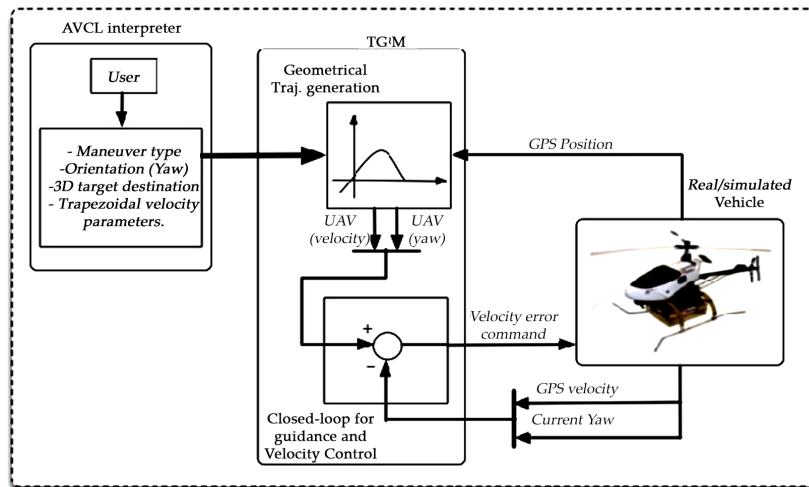
Compared to previous vehicle programming languages the AVCL and its interpreter provide several advantages: intuitive handling of different systems of units; its use of the object-oriented-programming paradigm; facilities for inter-vehicle communications; run-time definition of relations between vehicles, sensors and other equipment; it may be extended easily through C, C++ or C# code; the interpreter is a light-weight application written in C++, therefore it may be deployed in many SW/HW architectures. Before the development of the TG2M module the AVCL framework relied on a simpler guidance module to connect waypoints with straight-line segments, and while the language could describe complex maneuvers and mission constraints the framework lacked the capacity to fly a vehicle through complex paths.



**Figure 1.** The AVCL simplified diagram for mission planning

## TRAJECTORY GENERATION AND GUIDANCE MODULE - TG<sup>2</sup>M

The TG<sup>2</sup>M is designed to model 3D cartesian paths with parametric constraints. The system uses two types of mathematical descriptors for trajectories: polynomial interpolation and analytical functions. Complex maneuvers are described with polynomial interpolation based on third and fourth order splines with fixed boundary conditions (e.g. initial and final velocities user definition), which join all control points with a continuous and smooth path. Likewise, simple maneuvers like lines and circumference are created with analytical functions that constrain the geometry of the desired path. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc.



**Figure 2.** The TG2M framework

As shown in Fig. 2, two main modules compose the TG2M framework: the geometrical trajectory generation and the online closed-loop guidance. All the parameters and fundamental data (e.g. the desired maximum speed, etc) are provided by the user via the AVCL interpreter.

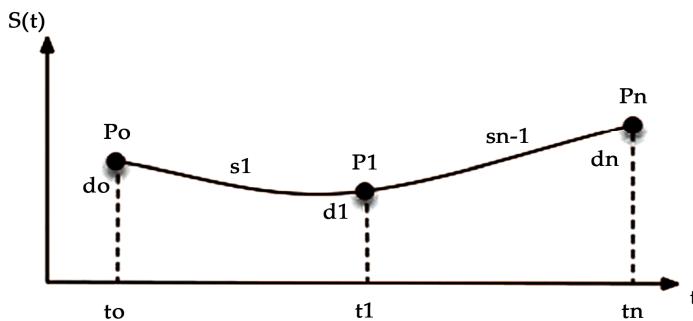
### Polynomial Interpolation Using Splines

The fundamental idea behind the spline interpolation is based on the definition of smooth curves through a number of points. These curves are represented by a polynomial function (normally of third-grade) defined by

some coefficients that determine the spline used to interpolate the numerical data points. These coefficients bend the line so that it passes through each of the data points without any erratic behavior or breaks in continuity. The essential idea is to define a third-degree polynomial function  $S(t)$  of the form:

$$S(t) = a_i t^3 + b_i t^2 + c_i t + d_i \quad (1)$$

This third-degree polynomial needs to conform to the following conditions in order to interpolate the desired knot-points as depicted in Fig. 3:



**Figure 3.** Knot-points to interpolate using 3D splines (with free boundary conditions)

- $S(t)$ ,  $S'(t)$  and  $S''(t)$  will be continuous on the interval  $[t_0, t_n]$ .
- Since the curve  $S(t)$  must be continuous across its entire interval, it can be concluded that each sub-function must joint at the data points, so:  $s_i(t_i) = s_{i-1}(t_i)$ .

Taking into account the set of conditions previously described, for each  $i = 1, 2, \dots, n-1$ ,  $t_i \in [t_i, t_{i+1}]$   $s_i(t_i) = s_{i-1}(t_i)$ , and letting  $h_i = \sum_{i=1}^{n-1} t_{i+1} - t_i$ , Eq. (1) is re-writing as:

$$s_i(t_i) = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i \quad (2)$$

Also, to make the curve smooth across the interval, the first derivative of the  $S(t)$  function must be equal to the derivative of the position reference points; this yields:

$$\begin{aligned}
 s_i'(t_i) &= 3a_i h_i^2 + 2b_i h_i + c_i \\
 s_i''(t_i) &= s_{i-1}'(t_i) \\
 s_i''(t_i) &= c_i = 3a_{i-1} h_i^2 + 2b_{i-1} h_i + c_{i-1}
 \end{aligned} \tag{3}$$

Applying the same approach for the second derivative:

$$\begin{aligned}
 s_i'''(t_i) &= 6a_i h_i + 2b_i \\
 s_i'''(t_i) &= s_{i+1}''(t_i) \\
 s_i'''(t_i) &= 2b_{i+1} = 6a_i h_i + 2b_i
 \end{aligned} \tag{4}$$

For the solution of the polynomial coefficients in Eq. (1), the system in Eq. (5) must be solved as:

$$A \cdot Y = f \tag{5}$$

where the matrix A R<sup>nxn</sup> (n is the number of knot-points to interpolate) corresponds to:

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_0 & 2(h_1 + h_2) & h_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \dots & \dots & \dots & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The term h R<sup>n-1</sup> in Eq. (6) is the time vector defined for each point (P<sub>0</sub>, P<sub>1</sub>, ..., P<sub>n</sub>). The bi coefficients in Eq. (1) are stacked in Y R<sup>nx3</sup>, which yields the term f R<sup>nx3</sup> as:

$$f = \begin{bmatrix} 0 \\ \frac{3}{h_1}(d_2 - d_1) - \frac{3}{h_0}(d_1 - d_0) \\ \vdots \\ \frac{3}{h_{n-1}}(d_n - d_{n-1}) - \frac{3}{h_{n-2}}(d_{n-1} - d_{n-2}) \\ 0 \end{bmatrix} \tag{7}$$

From Eq. (2) and (3), the a<sub>i</sub> and c<sub>i</sub> coefficients are respectively obtained as:

$$\begin{aligned}
 a_i &= \frac{b_{i+1} - b_i}{3h_i} \\
 c_i &= \frac{1}{h_i} (d_{i+1} - d_i) - \frac{h_i}{3} (2b_i - b_{i+1}) \\
 d_i &= S(t_i)
 \end{aligned} \tag{8}$$

*Example 1.* Simple UAV trajectory planning using 3D splines with fixed boundary condition. Let's define three knot-points as:  $P_0 = [0,0,0]$ ,  $P_1 = [5,10,10]$ ,  $P_2 = [0,10, 20]$  with the following time condition for each point:  $t = [0,10, 20]$  (note that the time vector components are given in seconds). Calculate a 3D spline considering zero initial and final velocities.

The natural 3D splines with free boundary conditions may generate smooth paths but without control over the velocities over the knot-points. Because of this, Eq. (6) and (7) must be complemented in order to generate a 3D spline with the fixed boundary conditions, in this case, with zero initial and final velocities. Re-writing those equations yield:

$$A = \begin{bmatrix} 2h_0 & h_0 & 0 & \dots & \dots & \dots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_0 & 2(h_1 + h_2) & h_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \quad (9)$$

The same system  $A \cdot Y = f$  must be solved with the new  $A \in R^{nxn}$  from Eq. (9) and  $f \in R^{nx3}$  defined in Eq. (10). Note that the first derivative of the  $S(t)$  function has been added in the first and last position of the  $f$  vector, allowing the control of the initial and final velocities of the curve.

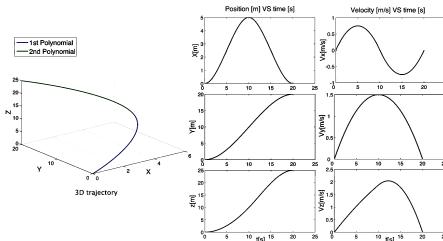
$$f = \begin{bmatrix} \frac{3}{h_0}(d_1 - d_0) - 3S'(t_0) \\ \frac{3}{h_1}(d_2 - d_1) - \frac{3}{h_0}(d_1 - d_0) \\ \vdots \\ \frac{3}{h_{n-1}}(d_n - d_{n-1}) - \frac{3}{h_{n-2}}(d_{n-1} - d_{n-2}) \\ 3S'(t_n) - \frac{3}{h_{n-1}}(d_n - d_{n-1}) \end{bmatrix} \quad (10)$$

The first procedure is to obtain the time vector as follows:  
 $h = \sum_{i=1}^{n-1} t_{i+1} - t_i = [10 \ 10]$ , then the system:  $Y = A^{-1} f$  must be solved to obtain the polynomial coefficients as:

$$A = \begin{bmatrix} 20 & 10 & 0 \\ 10 & 40 & 10 \\ 0 & 10 & 20 \end{bmatrix}, \quad Y_x = \begin{bmatrix} b_{0x} = 0.15 \\ b_{1x} = -0.15 \\ b_{2x} = 0.15 \end{bmatrix}, \quad Y_y = \begin{bmatrix} b_{0y} = 0.15 \\ b_{1y} = 0 \\ b_{2y} = -0.15 \end{bmatrix}, \quad Y_z = \begin{bmatrix} b_{0z} = 0.1125 \\ b_{1z} = 0.0750 \\ b_{2z} = -0.2625 \end{bmatrix}$$

$$f_x = \begin{bmatrix} 1.5 \\ -3 \\ 1.5 \end{bmatrix}, \quad f_y = \begin{bmatrix} 3 \\ 0 \\ -3 \end{bmatrix}, \quad f_z = \begin{bmatrix} 3 \\ 1.5 \\ -4.5 \end{bmatrix}$$

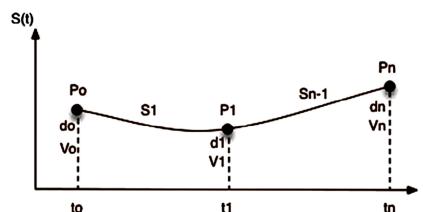
Finally, the two polynomials for each (x, y, z) component are evaluated from the time [0, 10] to [10, 20]. Figure 4 shows the results.



**Figure 4.** 3D spline with fixed boundary condition

The end of Example 1: Simple UAV trajectory planning using 3D splines.

In the previous example, the 3D splines with fixed boundary conditions allowed to define a smooth curve across the knot-points. Nevertheless, two basic problems must be taken into account: the 3D spline only allows the user to establish the initial and the final velocities of the whole trajectory, limiting the user to have total control over the other points. The second problem is the smoothness of the acceleration curves (linear). To solve these problems 4D splines must be used. These address the user total control over the velocity profile across the whole trajectory and its results in additional smoothness for position, velocity and even acceleration curves. This could be more effective for UAVs tasks where the mission requires a strong control of the vehicle acceleration and velocities at each defined knot-point.



**Figure 5.** Knot-points to interpolate with 4D splines

The fourth-degree polynomial is defined by:

$$s(t) = e_i t^4 + a_i t^3 + b_i t^2 + c_i t + d_i \quad (11)$$

The same 3D-spline conditions previously described also apply to the 4D-spline. To obtain a generalized solution of the system to solve ( $A \cdot Y = f$ ), we start from the three-point case as depicted in Fig. 5. The polynomials for each trajectory segment as a function of time  $t$  are:

$$\begin{aligned} s_n(t_0) &= e_0 t_0^4 + a_0 t_0^3 + b_0 t_0^2 + c_0 t_0 + d_0 = f(t_0) \\ s_n(t_1) &= e_0 t_1^4 + a_0 t_1^3 + b_0 t_1^2 + c_0 t_1 + d_0 = f(t_1) \\ s_{n-1}(t_1) &= e_1 t_1^4 + a_1 t_1^3 + b_1 t_1^2 + c_1 t_1 + d_1 = f(t_1) \\ s_{n-1}(t_n) &= e_1 t_n^4 + a_1 t_n^3 + b_1 t_n^2 + c_1 t_n + d_1 = f(t_n) \end{aligned} \quad (12)$$

Taking the first and second derivatives (velocities and accelerations), we obtain:

$$\begin{aligned} s'_1(t_0) &= 4e_0 t_0^3 + 3a_0 t_0^2 + 2b_0 t_0 + c_0 = V_0 \\ s'_1(t_1) &= 4e_0 t_1^3 + 3a_0 t_1^2 + 2b_0 t_1 + c_0 = V_1 \\ s'_{n-1}(t_1) &= 4e_1 t_1^3 + 3a_1 t_1^2 + 2b_1 t_1 + c_1 = V_1 \\ s'_{n-1}(t_n) &= 4e_1 t_n^3 + 3a_1 t_n^2 + 2b_1 t_n + c_1 = V_n \end{aligned} \quad (13)$$

The second derivatives of Eq. (12) yield a set of accelerations. Equaling the acceleration functions for the intermediate points ( $t_1$  for each case) and setting to zero the initial and final acceleration of the path segment yields:

$$\begin{aligned} s''_0(t_0) &= 12e_0 t_0^2 + 6a_0 t_0 + 2b_0 = s'_{n-1}(t_1) = 12e_1 t_1^2 + 6a_1 t_1 + 2b_1 \\ s''_0(t_1) &= 12e_0 t_1^3 + 6a_0 t_1 + 2b_0 = s''_{n-1}(t_n) = 12e_1 t_n^2 + 6a_1 t_n + 2b_1 \end{aligned} \quad (14)$$

Equations (12), (13) and (14) conform the complete system in order to obtain the ten polynomial coefficients. Solving  $A \cdot Y = f$ , the matrix  $A \in \mathbb{R}^{5(n-1) \times 5(n-1)}$  corresponds to:

$$A = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_1 & t_1^2 & t_1^3 & t_1^4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 & 4t_2^3 \\ 0 & 0 & 2 & 6t_1 & 12t_1^2 & 0 & 0 & -2 & -6t_1 & -12t_1^2 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 0 & 0 & -2 & -6t_2 & -12t_2^2 \end{bmatrix} \quad (15)$$

The polynomial coefficients are stacked into the  $Y \in R^{5(n-1)}$  vector, and with the  $f \in R^{5(n-1)}$  term, the total system is defined from Eq. (14) and (15):

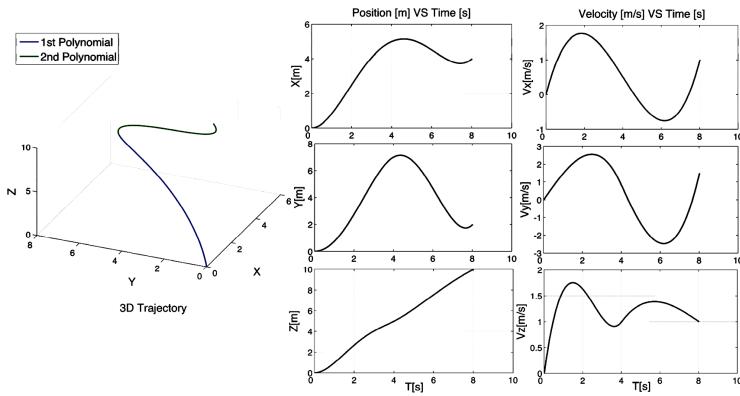
$$\begin{aligned} Y &= [d_0 \ c_0 \ b_0 \ a_0 \ e_0 \ d_1 \ c_1 \ b_1 \ a_1 \ e_1]^T \\ f &= [f(t_0) \ f(t_1) \ f(t_2) \ f(t_n) \ V_0 \ V_1 \ V_2 \ V_n \ 0 \ 0]^T \end{aligned} \quad (16)$$

*Example 2.* UAV trajectory planning using 4D splines with total velocity control Lets define three knot-points as:  $P_0 = [0,0,0]$ ,  $P_1 = [5,7,5]$ ,  $P_2 = [4,2,10]$  in the time  $t = [0,4,8]$  (note that the time vector components are given in seconds). Calculate the 4D spline considering the following velocity profile:  $V_0 = [0,0,0]$ ,  $V_1 = [0.5,0.8,1]$ ,  $V_2 = [1,1.5,1]$ .

Solving the system:  $Y = A^{-1} f$ , we obtain the following numerical data:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 16 & 64 & 256 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 16 & 64 & 256 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 64 & 512 & 4096 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8 & 48 & 256 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 48 & 256 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 16 & 192 & 2048 \\ 0 & 0 & 2 & 24 & 192 & 0 & 0 & -2 & -24 & -192 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \end{bmatrix}, \quad Y_{x,y,z} = \begin{bmatrix} d_{0x} & d_{0y} & d_{0z} \\ c_{0x} & c_{0y} & c_{0z} \\ b_{0x} & b_{0y} & b_{0z} \\ a_{0x} & a_{0y} & a_{0z} \\ e_{0x} & e_{0y} & e_{0z} \\ d_{1x} & d_{1y} & d_{1z} \\ c_{1x} & c_{1y} & c_{1z} \\ b_{1x} & b_{1y} & b_{1z} \\ a_{1x} & a_{1y} & a_{1z} \\ e_{1x} & e_{1y} & e_{1z} \end{bmatrix}, \quad f_{x,y,z} = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 7 & 5 \\ 5 & 7 & 5 \\ 4 & 2 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad f_{x,y,z} = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 7 & 5 \\ 5 & 7 & 5 \\ 4 & 2 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, the two polynomials for each ( $x, y, z$ ) component are evaluated from the time  $[0, 4]$  to  $[4, 8]$  seconds. Figure 6 shows the results:

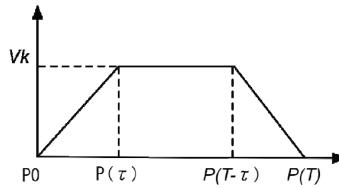


**Figure 6.** 4D spline with total control of the knot-points velocities

The end of Example 2: UAV trajectory planning using 4D splines with total velocity control

### Simple Maneuvers with Analytical Functions

For simple maneuvers the TG2M framework also supports the definition of straight-lines and circumferences via analytical functions that constrain the geometry of the desired path. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. This kind of parameterization is useful when the mission requires the UAV stops at the desired end-point of the trajectory effectively, due to the usercontrol of the acceleration slope tilt level. For both (lines and circumferences) the desired set of velocities must fulfill the following trapezoidal velocity profile:



**Figure 7.** Trapezoidal velocity profile used for simple UAV maneuvers

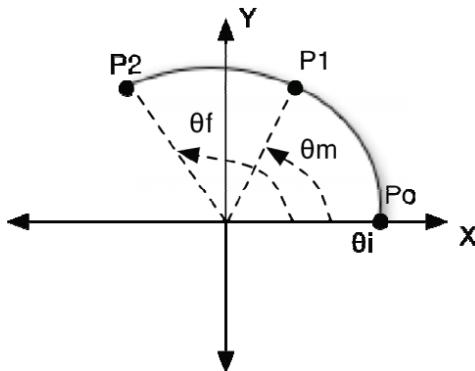
From Fig. 7, three fundamental segments compose the total function to define the straightline. The two intermediate points of the trapezoidal curve:  $P(\tau)$ ,  $P(T - \tau)$  are calculated as:

$$\begin{aligned}
 P(\tau)_{x,y,z} &= \frac{1}{2\|P(T)_{x,y,z} - P_{0x,y,z}\|} \frac{V_k}{t} t^2 \left( P(T)_{x,y,z} - P_{0x,y,z} \right) + P_{0x,y,z} \\
 P(T-\tau)_{x,y,z} &= \frac{V_k(T-2t)}{\|P(T)_{x,y,z} - P_{0x,y,z}\|} \left( P(T)_{x,y,z} - P_{0x,y,z} \right) + P(\tau)_{x,y,z}
 \end{aligned} \tag{17}$$

Once that the intermediate points have been defined, the three segments that compose the total function are defined by Eq. (18). In the first section (from  $P_0$  to  $P(\tau)$ ) the initial velocity is set  $V_i = 0$  and progresses toward a final velocity  $V_k$ . The second segment (from  $P(\tau)$  to  $P(T-\tau)$ ) is traced at constant maximum velocity  $V_k$ . Finally the last segment (from  $P(T-\tau)$  to  $P(T)$ ) drives the vehicle from  $V_k$  to zero velocity.

$$f(t) = \begin{cases} \frac{V_k}{2\tau(P(\tau)-P_0)} t^2 (P(\tau) - P_0) + P_0 & 0 \leq t \leq \tau \\ \frac{V_k}{(P(T-\tau)-P(\tau))} (t-\tau)(P(T-\tau) - P(\tau)) + P(\tau) & \tau < t \leq T-\tau \\ \frac{(t-T+\tau)}{(P(T)-P(T-\tau))} \left( -\frac{V_k}{2\tau}(t-T+\tau) + V_k \right) (P(T) - P(T-\tau)) + P(T-\tau) & T-\tau < t \leq T \end{cases} \tag{18}$$

The same approach is applied to generate circumferences with the trapezoidal profile used for the straight-lines. Three knot-points define the circumference path and the objective is to find the trace angle across the trajectory.



**Figure 8.** X-Y plane for circumference maneuver

The first step is to determine the center of the circle that joins the three knot-points. The equation of the trajectory plane is defined by the cross product between the vectors formed by points  $P_0$ ,  $P_1$  and  $P_2$ , as shown in Eq. (19).

$$\begin{aligned} P_0 \vec{P}_1 \times P_0 \vec{P}_2 &= N(A, B, C) \\ Ax + By + Cz &= cte \end{aligned} \quad (19)$$

The center of the circle  $c(x_c, y_c, z_c)$  is always equidistant to any point over the circle, then:

$$\begin{aligned} (x_{0,1,2} - x_c)^2 + (y_{0,1,2} - y_c)^2 + (z_{0,1,2} - z_c)^2 &= K^2 \\ Ax_c + By_c + Cz_c &= cte \end{aligned} \quad (20)$$

To obtain the relation between the angle motion ( $\theta$ ), the angular velocity ( $\omega$ ) and the tangential velocity ( $v_t$ ), constant velocity is assumed across the path, yielding:

$$\begin{aligned} \omega &= \frac{d\theta}{dt} = \frac{v_t}{r} \\ d\theta &= \frac{v_t}{r} dt \\ \int d\theta &= \int \frac{v_t}{r} dt \\ \theta(t) &= \frac{v_t}{r} t + \theta_i \quad \text{where } t \in [0, T] \end{aligned} \quad (21)$$

Likewise, the relation between the angle motion ( $\theta$ ) and the angular acceleration of the curve is given by:

$$\begin{aligned} at + v_0 &= r \frac{d\theta}{dt} \\ d\theta &= \frac{at + v_0}{r} dt \\ \int d\theta &= \int \frac{at + v_0}{r} dt \\ \theta(t) &= \frac{a}{2r} t^2 + \frac{v_0}{r} t + \theta_i \quad \text{where } t \in [0, T] \end{aligned} \quad (22)$$

If the known parameter is the total motion time ( $T$ ) of the trajectory, we set the acceleration term to the left-side of the equation, yielding:

$$\theta(t) = \theta_f = \frac{a}{2r} t^2 + \frac{v_0}{r} t + \theta_i$$

$$a = \frac{2[(\theta_f - \theta_i) - v_0 T]}{T^2} \quad (23)$$

Otherwise, if the constrained parameters are the initial and final velocity, the acceleration function is given by:

$$a = \frac{v_{ft} - v_{0t}}{T}$$

$$T = \frac{2(\theta_f - \theta_i)}{v_{0t} + v_{ft}} \quad (24)$$

*Example 3.* UAV straight-line trajectory with trapezoidal velocity profile  
Let's defined a straight-line trajectory starting from  $P_0 = [0,0, 20]$  to  $P(T) = [10,0, 20]$  with 10 meters of displacement in the X coordinate at a constant altitude of 20 meters. Define the 3D cartesian trajectory with a maximum velocity of 1m/s taking into account a trapezoidal velocity profile with 30% of acceleration time.

The distance vector to trace is:  $\text{dist} = P(T) - P_0 = [10,0,0]$ .

The total velocity is obtained from the norm of the vector dist:  $V_T = \text{dist} = 10$ .

The total motion time is calculated as:  $T = \frac{V_T}{V_k(1-pt)} = 7$ , where  $V_k$  is the maximum velocity at 1m/s and  $pt$  is the percentage of acceleration (30% = 0.3). The acceleration motion is also obtained as:  $a = \frac{V_k}{T * pt} = 0.2381$ . The number of the total points to generate is:  $n = 28$  (the user defines this parameter depending on the UAV controller data rate). In the same way, the number of points in the first and last segment of the trapezoidal profile is calculated as:  $k = 2(pt \cdot n) = 16$ , and for the constant velocity segment:  $k_c = n - 2k = 12$ . Once that the preliminary data have been calculated, Eq. (16) is used to obtain the intermediate points (see Fig. 7):

$$\begin{aligned}
 Px(\tau) &= \frac{1}{2V_t} at^2 (dist)_x + P_{0x} = 2.10 & Px(T-\tau) &= \frac{V_k(T-2t)}{V_T} (dist)_x + P(\tau)_x = 7.7 \\
 Py(\tau) &= \frac{1}{2V_t} at^2 (dist)_y + P_{0y} = 0 & Py(T-\tau) &= \frac{V_k(T-2t)}{V_T} (dist)_y + P(\tau)_y = 0 \\
 Pz(\tau) &= \frac{1}{2V_t} at^2 (dist)_z + P_{0z} = 20 & Pz(T-\tau) &= \frac{V_k(T-2t)}{V_T} (dist)_z + P(\tau)_z = 20
 \end{aligned}$$

The first segment of the path (from  $P_0$  to  $P(\tau)$ ) is calculated using Eq. (17):

$$\begin{aligned}
 fx(t) &= \sum_{i=1}^{k/2} \frac{V_k}{2t \|P(t) - P_0\|} t_i^2 (P(t)_x - P_{0x}) + P_{0x} \\
 fy(t) &= \sum_{i=1}^{k/2} \frac{V_k}{2t \|P(t) - P_0\|} t_i^2 (P(t)_y - P_{0y}) + P_{0y} \quad 0 \leq t \leq \tau \\
 fz(t) &= \sum_{i=1}^{k/2} \frac{V_k}{2t \|P(t) - P_0\|} t_i^2 (P(t)_z - P_{0z}) + P_{0z} \\
 t &= \left[ 0 : \frac{1}{V_k} \frac{2\|P(\tau) - P_0\|}{k} : \frac{1}{V_k} 2\|P(\tau) - P_0\| \right]
 \end{aligned}$$

The time vector  $t$  is obtained as:  $t = [0 \ 0.5250 \ 1.0500 \ 1.5750 \ 2.1000 \ 2.6250 \ 3.1500 \ 3.6750 \ 4.2000]$ , which is:

For the second segment (at constant velocity  $V_k$ ), the time vector is calculated as:  $t = \left[ 0 : \frac{1}{n-2k} : 1 \right] = [0 \ 0.0833 \ 0.1667 \ 0.25 \ 0.3333 \ 0.4167 \ 0.5 \ 0.5833 \ 0.6667 \ 0.75 \ 0.83 \ 0.9167]$

$$\begin{aligned}
 fx(t) &= \sum_{i=1}^{kc} t_i (P(T-\tau)_x - P(\tau)_x) + P(\tau)_x \\
 fy(t) &= \sum_{i=1}^{kc} t_i (P(T-\tau)_y - P(\tau)_y) + P(\tau)_y \quad \tau < t \leq T - \tau \\
 fz(t) &= \sum_{i=1}^{kc} t_i (P(T-\tau)_z - P(\tau)_z) + P(\tau)_z
 \end{aligned}$$

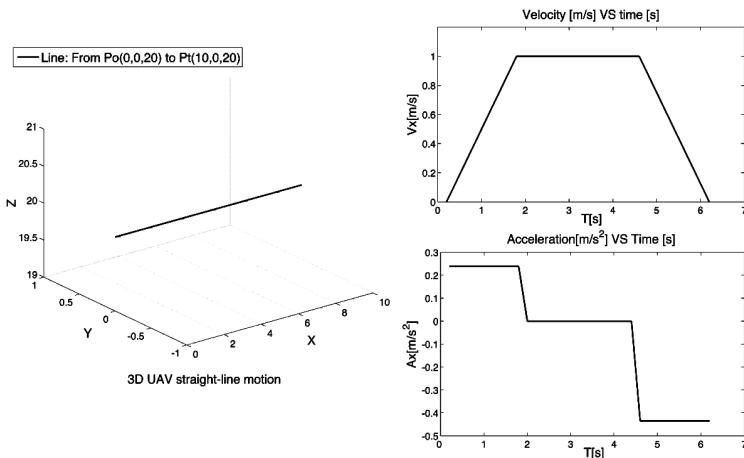
Finally the last segment drives the vehicle from  $V_k$  to zero end velocity. The time vector is:

$$t = \left[ 0 : \frac{1}{V_k} \frac{2\|P(T-\tau) - P(\tau)\|}{k} : \frac{1}{V_k} 2\|P(T-\tau) - P(\tau)\| \right] = \left[ 0 \ 0.287 \ 0.575 \ 0.862 \ 1.15 \ 1.43 \right. \left. \begin{matrix} \\ 1.72 \ 2.01 \ 2.3 \end{matrix} \right]$$

yielding the following function for each coordinate of motion:

$$\begin{aligned}
 f(t)_x &= \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2 \|P(T) - P(T-t)\|_x} \left( P(T)_x - P(T-t)_x \right) + P(T-t)_x \\
 f(t)_y &= \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2 \|P(T) - P(T-t)\|_y} \left( P(T)_y - P(T-t)_y \right) + P(T-t)_y \quad T-t < t \leq T \\
 f(t)_z &= \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2 \|P(T) - P(T-t)\|_z} \left( P(T)_z - P(T-t)_z \right) + P(T-t)_z
 \end{aligned}$$

The results are presented in Fig. 9:



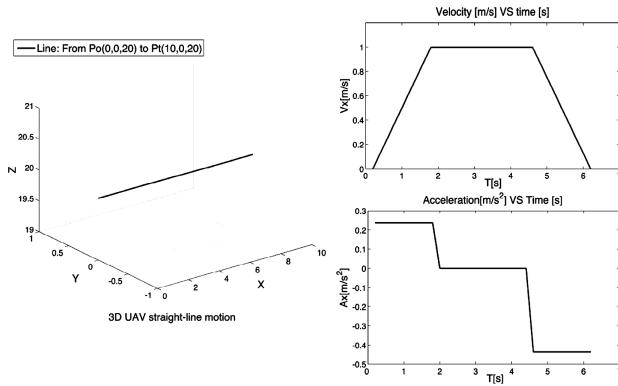
**Figure 9.** UAV straight-line motion with trapezoidal velocity profile

The end of Example 3: UAV straight-line trajectory with trapezoidal velocity profile

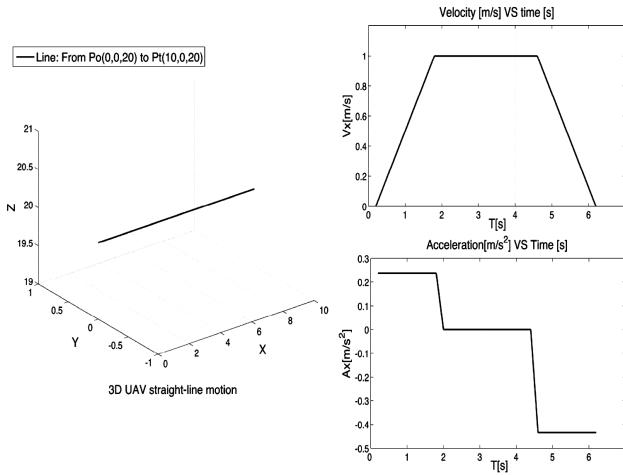
## UAV Guidance

The geometrical representation of a UAV trajectory has been presented in the previous sub-section. Complex trajectories may be described using third-fourth order degree splines, or simple maneuvers may be generated using common lines and circumferences functions with some parametric features defined by the end-user.

But in order to complete the generation of trajectories for a single UAV a guidance module is required.



**Figure 10.** TG2M guidance scheme

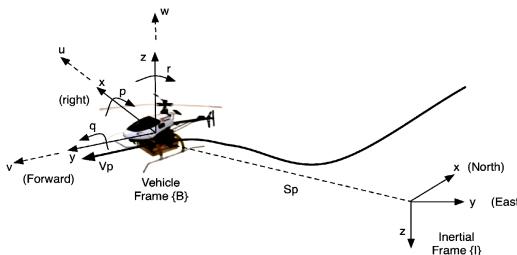


**Figure 11.** Velocity error command

The AVCL simulation module contains a dynamic model of a mini-helicopter (Valero, 2005) and its associated controllers: attitude, velocity and position. The velocity controller is based on a Proportional-Integrative “PI” design; it receives the velocity error command generated by the guidance module shown in Fig. 10. These velocity error references are named V<sub>xref</sub> and V<sub>yref</sub>, with respect to either the world or vehicle’s frame, and are calculated with the UAV’s error position. In Fig. 11 the theoretical (or ideal) UAV velocity vector is represented by the V<sub>t</sub> term. Due to wind and other perturbations during flight, a velocity error vector V<sub>e</sub> must be considered in order to set the final velocity references to send to the vehicle’s controller.

This vector is derived from the UAV position error between the current and desired positions. Finally, the velocity references  $Vx_{ref}$  and  $Vy_{ref}$  are the components of the vector  $V_t + V_e$ .

The guidance module must take into account that the helicopter's orientation (yaw) is independent from the direction of travel. The high-level modules must generate a set of orientations across the vehicle's trajectory. The built-in AVCL control module (see Fig. 10) is capable of receiving velocity and yaw angle orientation commands from the  $TG^2M$  module and generating the needed commands for the attitude controller that governs the vehicle's roll and pitch. The  $TG^2M$ 's guidance module focuses on the generation of yaw references, and use a simple Proportional "P" controller for smooth yaw angle transition during the flight. Two methods are used to generate the yaw angle references: for simple maneuvers the yaw angle is calculated using simple trigonometric relations due to the path displacement. For complex trajectories using splines we introduce a feasible method to calculate a smooth yaw angle evolution. This method also shows how to calculate roll and pitch references due to the vehicle trajectory and its velocity. For roll, pitch and yaw angles calculation, the following frame of reference is used:



**Figure 12.** Frames of reference for UAV guidance and control

From Fig. 12, the following vectors can describe the motion of the UAV vehicle in 6 DOF:

$$\begin{aligned}\eta &= [\eta_1, \eta_2]^T = [x, y, z, \phi, \theta, \psi]^T \\ v &= [v_1, v_2]^T = [u, v, w, p, q, r]^T\end{aligned}\quad (25)$$

In Eq. (25),  $\eta_1$  denotes the position of the center of mass CM of the vehicle and  $\eta_2$  its orientation described by the Euler angles with respect to the inertial frame  $\{I\}$ . The vector  $v_1$  refers to the linear velocity and  $v_2$  to the

angular velocity of vehicle frame {B} with respect to inertial frame {I}. In order to express the velocity quantities between both frames of references (from {B} to {I} and vice versa), the following transformation matrix is used:

$$\eta'_1 = R_B^I v_1$$

$$\eta'_1 = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\theta s\phi \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} v_1 \quad (26)$$

The body-fixed angular velocities and the rate of the Euler angles are related through:

$$\eta'_2 = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{bmatrix} v_2 \quad (27)$$

The position and the magnitude of the velocity vector at a point P on the trajectory are given by :

$$S_p = [x, y, z]^T$$

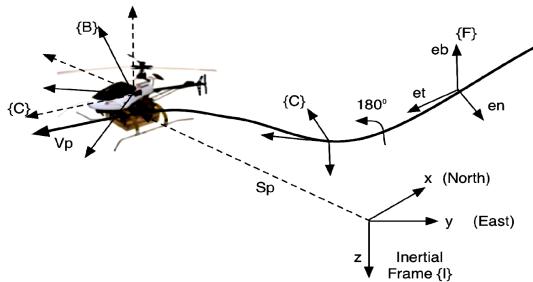
$$V_p = \|S'_p\| = \sqrt{x'^2 + y'^2 + z'^2} \quad (28)$$

The method to define the Euler angles is based on the Frenet-Serret theory (Angeles, 1997). To every point of the curve we can associate an orthonormal triad of vectors (a set of unit vectors that are mutually orthogonal) namely the tangent  $e_t$ , the normal  $e_n$  and the binormal  $e_b$  (see Fig. 13). The Frenet-Serret theory says that by properly arranging these vectors in a matrix  $\in \mathbb{R}^{3 \times 3}$ , we obtain a description of the curve orientation due to the position, velocity and acceleration of the UAV while tracing out the path. The unit vectors are then defined as:

$$e_t = \frac{S'_p}{V_p}, \quad e_b = \frac{(S'_p \times S''_p)}{\|S'_p \times S''_p\|}, \quad e_n = e_b \times e_t \quad (29)$$

In the definition of a frame associated with the curve the original definition of the Frenet frame for counterclockwise rotating curves is used; in the case of a clockwise rotating curve, the z-axis of the Frenet frame points in the opposite direction upwards than the inertial {I} frame. So in order to define small relative rotation angles for the orientation of a vehicle

rotating clockwise and having its  $z_b$  axis pointing downwards, we define a reference frame associated with the curve as previously, but rotated with respect to the Frenet by an angle of 180 degrees about the x-axis of the Frenet frame (see Fig. 13).



**Figure 13.** The inertial, the Frenet, the vehicle and the curve Frames

Collectively we denote the Frenet and the rotated frame as the “curve” frame  $\{C\}$ . According to the notation of rotational transformations used in robotics literature, we can express the coordinates of a vector given in the curve frame  $\{C\}$  to the  $\{I\}$  frame with the matrix:

$$R_C^I = \begin{bmatrix} e_t & e_n & e_b \end{bmatrix}$$

$$R_I^C = R_C^{I^T} \quad (30)$$

For a counterclockwise rotation:  $R_C^I = R_x(180^\circ) \begin{bmatrix} e_t & e_n & e_b \end{bmatrix}$ . Likewise, the rotation of the  $\{B\}$  frame from the  $\{C\}$  frame to the reference  $\{R\}$  frame can be expressed using customary aeronautical notation by considering the sideslip angle  $\beta$  and angle of attack  $\alpha$ , (Siouris, 2004):

$$\beta = \sin^{-1} \left( \frac{v_R}{V_p} \right)$$

$$\alpha = \tan^{-1} \left( \frac{w_R}{u_R} \right) \quad (31)$$

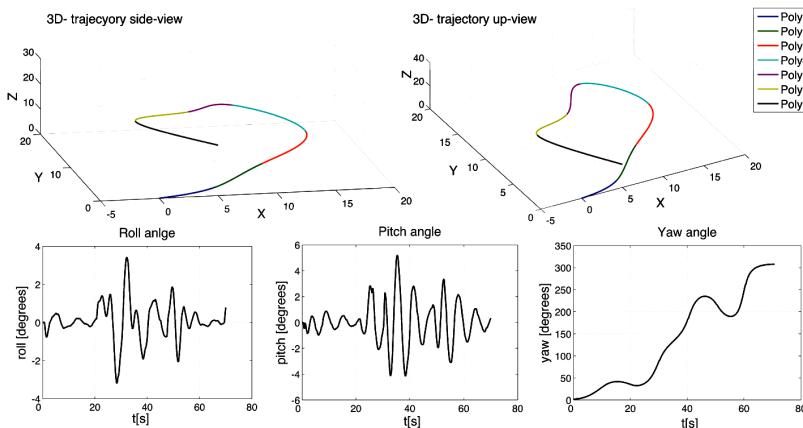
The vector  $v_R$  refers to the y-axis velocity component in the reference frame and  $w_R, u_R$  to the z and x - axis respectively. The overall rotation is composed by a rotation about body  $z_B$  axis through the angle  $\beta$ , followed by a rotation about the body  $y_B$  through the angle  $\alpha$ , which is expressed as:

$$R_c^R = R_y^T(\alpha) R_z^T(-\beta) \quad (32)$$

Finally, the roll, pitch and yaw angles can be deduced as follows:

$$\begin{aligned} R_i^R &= R_c^R R_i^C \\ \phi &= \text{atan2}(r_{23}, r_{33}) \\ \theta &= \text{atan2}(-r_{13}, \sqrt{r_{23}^2 + r_{33}^2}) \\ \psi &= \text{atan2}(r_{12}, r_{11}) \end{aligned} \quad (33)$$

Where  $r_{i,j}$  represent the components of the rotation matrix  $R_i^R \in \mathbb{R}^{3 \times 3}$ . Computing the previous methodology, we use 3D splines with fixed boundary conditions in order to generate a complicated path as shown in Fig. 11. Seven knot-points have been used (distance are in meters):  $P = [0 \ 0 \ 0; 5 \ 1 \ 2; 10 \ 5 \ 5; 15 \ 10 \ 10; 10 \ 15 \ 15; 5 \ 10 \ 20; 0 \ 8 \ 20; 5 \ 0 \ 20]$ . Eq. (33) has been used to obtain the UAV orientation with respect to the Inertial Frame as:



**Figure 14.** Roll, Pitch and Yaw angle references for UAV guidance

For complex maneuvers the Frenet theory allowed to define smooth yaw references as well as roll and pitch angles if it is required. Nevertheless, the computational cost of calculating those equations could decrease the system performance if the number of knot-points of the path is large. For this reason, normal maneuvers such as straight-lines use simple trigonometric theory to obtain the UAV orientation. On the other hand, the end-user is able to define the kind of orientation of the vehicle, this means that the vehicle is not constrained to be oriented just by the trajectory direction, hence, it will be

able to trace out the trajectory oriented towards to any fixed point defined.

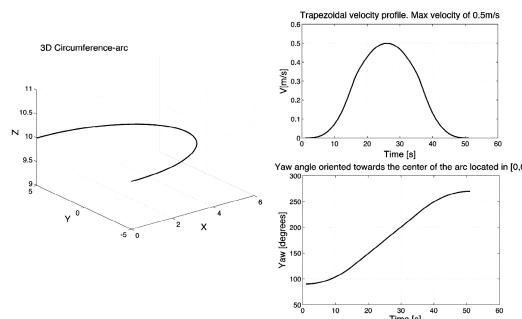
The yaw angle defined by the term  $\psi$  is given by:

$$\psi = \tan^{-1} \left( \frac{y_{diff}}{x_{diff}} \right) \quad (34)$$

Where  $x_{diff}$ ,  $y_{diff}$  correspond to the difference between the target fixed point and the current position of the UAV. In addition, depending of the motion quadrant, the term  $\psi$  in Eq. (34) must be fixed, this means that for the {x+,y+} and the {x-,y-} quadrant, the yaw angle is  $\psi = \psi + \pi$ , otherwise, for the {x+,y-} quadrant,  $\psi = \psi + 2\pi$ .

Figure 15 shows a circumference-arc generated using Eq. (24) as well as the yaw evolution of the UAV oriented towards the center of the arc located at [0,0] coordinate in the x-y plane using the previously theory described in Eq. (34).

This section has successfully introduced the mathematical treatment and methods for the generation of complex trajectories and simple maneuvers using the available theory reported in specialized literature (LaValle S.M, 2006), (Jaramillo-Botero et al., 2004). Geometrical trajectory generation and some techniques for its parameterization based on polynomial splines and function with trapezoidal velocity profile are an interest solution for this problem, actually, some of these methodologies are used for complex UAV trajectory definition nowadays. The novel solution presented in this book is the integration of these methods into a powerful environment that allows high-level user control of complex missions. The AVCL definitively brings those features and the next section will introduce some tests using the AVCL interpreter and the simulation environment.



**Figure 15.** Circumference 3D motion, trapezoidal velocity profile and yaw angle evolution

## TG<sup>2</sup>M SIMULATION

Results As shown in Fig. 1 the Mission Planner (MP) has two similar loops for mission planning and simulation/supervision. The difference is that in the Planning Loop the interpreter sends the projected waypoints back to the MP's Enhanced Reality, while in the Simulation Loop the interpreter commands the simulated vehicle, which in turn sends the simulated positions to the MP. Our research group has developed a Simulink-based model of a UAV helicopter named Vampira, which includes a position controller and is capable of real-time simulation. This simulator has been used with the Mission Planning and Simulation tool to test the TG2M. For Mission Supervision the AVCL commands would be sent to the real vehicle, and its position plotted alongside the projected and/or simulated paths. The Vampira helicopter was built within the framework of the project: "Guidance and Control of an Unmanned Aerial Vehicle" DPI 2003 01767 (VAMPIRA), granted by the Spain Ministry of Science and Technology, and it will be used for the real-world tests of the built-in TG2M framework. Figure 16 shows the Vampira prototype, which includes: a GPS, Wi-Fi link, IMU, and a PC104 computer for the low-level control (main rotor and tail servos). The Vampira's dynamics model has been obtained, identified and validated in previous works (Valero, 2005), (del Cerro et al., 2004). This work takes advantage from the AVCL simulation capabilities in order to validate the TG2M framework theory for trajectory planning.

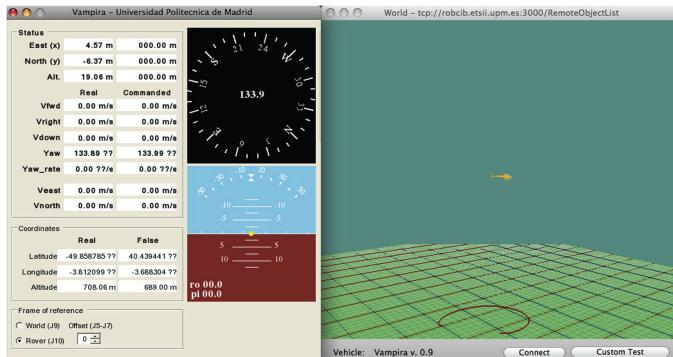


**Figure 16.** The Vampira's Helicopter prototype

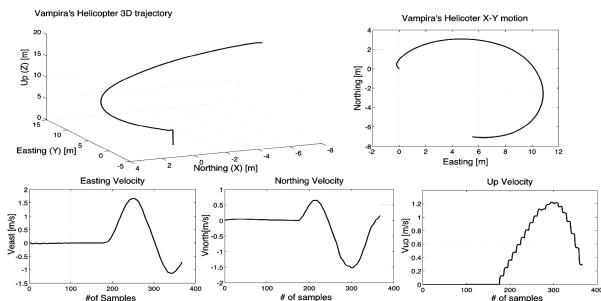
Three test scenarios showcase the TG2M validation process. These tests involve the whole methodology previously presented in the other sections of this chapter, as well as the numerical simulation results using the AVCL environment and the embedded dynamics and control algorithms for the Vampira's helicopter. Two complex maneuvers are presented using 3D and

4D splines respectively and a simple last test using analytical function to generate a parameterized circumference motion.

- 1). Semi-spiral using 3D splines for the velocity profile generation and the Frenet theory for UAV orientation: In this first test, we used a 3D spline to joint three knot control points: ( $P_0(0, 0, 0)$ ,  $P_1(3, 5, 10)$ ,  $P_2(6, -7, 20)$ ) at the desire time (given in seconds) for each point: ( $t(0, 10, 20)$ ) and the desire initial and final speed (given in m/s): ( $V_0(0, 0, 0)$ ,  $V_f(0, -0.2, 0.4)$ ):



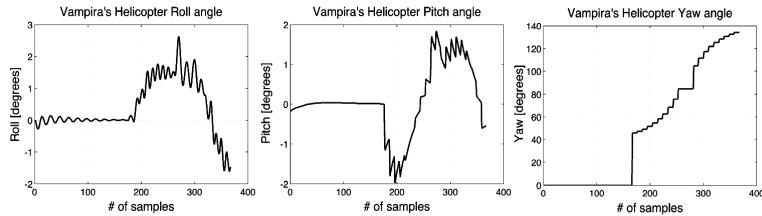
**Figure 17.** The AVCL simulation environment: Vampira's helicopter executing a semi-spiral motion using 3D splines



**Figure 18.** Test1: Cartesian UAV position and velocities with respect to the Inertial Frame

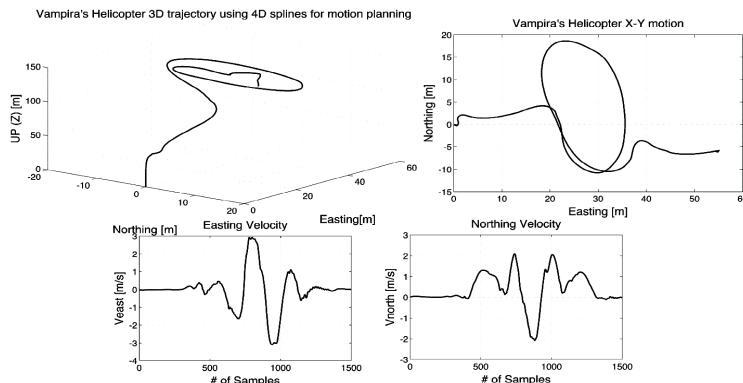
The UAV started from initial point located at (0, 0, 0) coordinate and finished its trajectory at (6, -7, 20). Visual simulation depicted in Fig 17, showed smooth motion across the trajectory due to the 3D spline approach. Nonetheless, 3D splines just allow the user to define the initial and final

velocities of the motion, lacking of velocity control for the rest of the knotpoints.

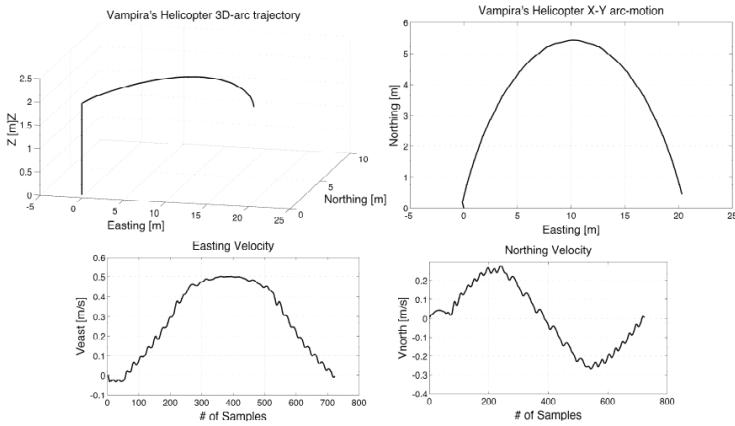


**Figure 19.** Test1: UAV orientation (Euler angles evolution)

To solve this problem, the following test introduces a more complex trajectory generation using 4D splines, addressing total user control of the UAV velocity profile. 2). Complex trajectory using 4D splines for the velocity profile generation and the Frenet theory for UAV orientation. This trajectory includes different kind of maneuvers joined into a single polynomial function (take-off, circumference-type motion and slow down in spiral-type motion). This test includes UAV long-endurance to high altitude (150 meters above ground) and a maximum easting displacement about of 60 meters. The following knot-control points (given in meters) have been defined:  $(P_0(0, 0, 0), P_1(0, 0, 20), P_2(0, 0, 40), P_3(0, 0, 60), P_4(10, 2, 80), P_5(20, 4, 110), P_6(25, -7, 130), P_7(30, -10, 150), P_8(35, -5, 140), P_9(30, 16, 125), P_{10}(20, 5, 130), P_{11}(33, -10, 145), P_{12}(40, -5, 135), P_{13}(55, -6, 125))$ :



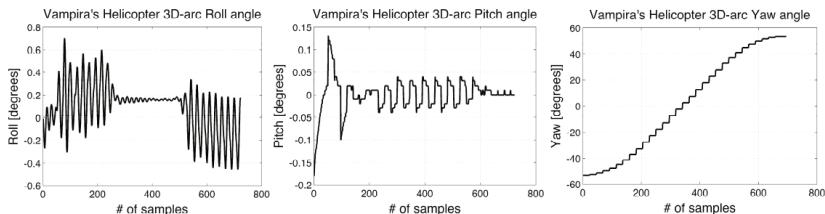
**Figure 20.** Test2: smooth 4D spline for complex maneuver



**Figure 21.** Test3: Arc-type motion using analytical functions

The advantage about using 4D splines relies in the possibility of defining feasible paths that matches with the knot-control points defined (with less match error percentage than the 3D polynomial splines). In addition, the user is able to define the set of velocities for each of the knot-points during the motion. The set of velocities (given in m/s) are:  $(V_0(0, 0, 0), V_1(0, 0, 0.8), V_2(0, 0, 1), V_3(0, 0, 1.2), V_4(0.5, 1, 1.4), V_5(0, 0.5, 1.7), V_6(2, 1.5, 2.5), V_7(3, 2.2, 3), V_8(2, 1.2, 2), V_9(1, 0.5, 1.5), V_{10}(-0.5, -1, 0.8), V_{11}(-3, -2, 0.4), V_{12}(-1, -0.5, 0.8), V_{13}(0, 0, 0))$ .

3). Simple arc-type maneuver with trapezoidal velocity profile parameterization: for the analytical AVCL feature of trajectory planning, the TG2M module supports straight-lines and circumferences motions. An arc defined by:  $P_0(0, 0, 2), P_1(10, 5.5, 2), P_2(20, 0, 2)$ ) with a maximum velocity of 0.5m/s is tested using the AVCL interpreter that allows the user to define the trapezoidal velocity profile configuration. For this case, the acceleration slopes of the curves (see Fig. 21) have been set to the 30% of the total motion.



**Figure 22.** Test3: UAV orientation (Euler angles evolution)

## FINAL OBSERVATIONS

For modeling continuous cartesian trajectories in the AVCL, several analytical functions and polynomial interpolation methods are available; all of which can be used in any combination. The TG<sup>2</sup>M module handles the definition of trajectories using knot control points as well as the incorporation of path constraints. It also supports the definition of complex tasks that require the construction of trajectories made up of different primitive paths in any combination of analytical and interpolated functions. The user-designed spatial trajectories can be visualized in three dimensions on the display window or plotted versus time using the embedded plotting library.

Simulation results have shown that the TG<sup>2</sup>M module works perfectly for the definition and testing of wide kind of smooth trajectories, allowing the user a high-level control of the mission due to the AVCL interpreter. The three different scenarios used for testing, allowed verifying that the mathematical framework used for the trajectory generation and guidance was really working during simulation flight. Percentage errors during maneuver execution were minimal, maintaining the UAV at the desired velocity limits and within the established path. We also incorporated velocity error fixing during flight. For high altitude tests, the velocity of the wind plays a mandatory role as a main disturbance external force. The TG<sup>2</sup>M module includes wind perturbation compensation. The Guidance module fixes the velocity commands in real-time flight maneuver, decreasing the error position tracking. For the three scenarios tests, the AVCL simulation environment includes normal wind conditions during simulation flight, introducing small perturbations into the UAV equations of motion. As shown in the obtained results, those perturbations were compensated, allowing the UAV to follow the desired trajectory within the less error as possible.

The Frenet-Serret formulas included for the UAV orientation also presented a good approach in order to obtain smooth UAV rotation rate during flight. The use of simple trigonometric theory to obtain and define the UAV orientation profile (Yaw angle) is not convenient for complex maneuvers. Splines sometimes require a lot of know-points for feasible trajectory guidance, hence, using these polynomial equations, the Frenet approach allowed smooth angle changes between knot-points, which it had not been obtained with the simple trigonometric angle calculation.

## REFERENCES

1. JAA & Eurocontrol,.A concept for the European Regulations for Civil Unmanned Aerial Vehicles. UAV Task-Force Final Report. 2004
2. Coifman, B., McCord, M., Mishalani, M., Redmill, K., Surface Transportation Surveillance from Unmanned Aerial Vehicles. Proc. of the 83rd Annual Meeting of the Transportation Research Board, 2004.
3. Held, Jason M; Brown, Shaun and Sukkarieh, Salah. Systems for Planetary Exploration. 15th NSSA Australian Space Science Conference, pp. 212-222, ISBN: 0864593740. RMIT University, Melbourne, Australia, from 14 to 16 September 2005.
4. Gutiérrez, P., Barrientos, A., del Cerro, J., San Martin., R. Mission Planning and Simulation of Unmanned Aerial Vehicles with a GIS-based Framework; AIAA Guidance, Navigation and Control Conference and Exhibit. Denver, EEUU, 2006.
5. Rysdyk, R. UAV path following for constant line-of-sight. In 2th AIAA Unmanned Unlimited. Conf. and Workshop and Exhibit, San Diego, CA, 2003.
6. Price, I.C, Evolving self organizing behavior for homogeneous and heterogeneous swarms of UAVs and UCAVS. PhD Thesis, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2006.
7. Herwitz, S. Developing Military COEs UAV applications. UAV Applications Center – NASA Ames Research Center, Moffett Field, CA. 2007.
8. Alison A. P., Bryan G., Suresh K. K., Adrian A. K., Henrik B. C. and Eric N. J. Ongoing Development of an Autonomous Aerial Reconnaissance System at Georgia Tech . UAV Laboratory, School of Aerospace Engineering. Georgia Institute of Technology, 2003
9. Jaramillo-Botero A., Correa J.F., and Osorio I.J., Trajectory planning in ROBOMOSP, Robotics and Automation Group GAR, Univ. Javeriana, Cali, Colombia, Tech. Rep. GAR-TR-10-2004, 2004.
10. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
11. Moitra, A., Mattheyses, R.M., Hoebel, L.J., Szczerba, R.J., Yamrom, B.: Multivehicle 5reconnaissance route and sensor planning. IEEE Transactions on Aerospace and Electronic Systems, 37 (2003).
12. Zheng, C., Li, L., Xu, F., Sun, F., Ding, M.: Evolutionary Route Planner

- for Unmanned Air Vehicles. *IEEE Transactions on Robotics* 21 (2005) 609–620
- 13. Frazzoli, E. Maneuver-based motion planning and coordination for single and multiple UAVs. AIAA's 1st technical conference and workshop on unmanned aerospace vehicles. University of Illinois at Urbana-Champaign, il 61801. S. Portsmouth, Virginia. May 2002.
  - 14. Angeles J., Fundamentals of Robotic Mechanical Systems. Theory, Methods, and Algorithms. Springer, New York, 1997.
  - 15. Siouris, G. M. Missile Guidance and Control Systems. Springer, New York, 2004.
  - 16. Valero, Julio. Modelo dinámico y sistema de supervisión de vuelo de un helicóptero autónomo. Proyecto de fin de carrera. ETSIIM-UPM. 2005.



# CHAPTER

# 6

## Modelling and Control Prototyping of Unmanned Helicopters

---

**Jaime del-Cerro, Antonio Barrientos and Alexander Martínez**

Universidad Politécnica de Madrid – Robotics and Cybernetics Group, Spain

### INTRODUCTION

The idea of using UAV's (Unmanned Aerial Vehicles) in civilian applications has created new opportunities for companies dedicated to inspections, surveillance or aerial photography amongst others. Nevertheless, the main drawback for using this kind of vehicles in civilian applications is the enormous cost, lack of safety and emerging legislation. The reduction in the cost of sensors such as Global Positioning System receivers (GPS) or

---

**Citation:** Jaime del-Cerro, Antonio Barrientos and Alexander Martinez (January 1st 2009). "Modelling and Control Prototyping of Unmanned Helicopters", Aerial Vehicles, Thanh Mung Lam, IntechOpen, DOI: 10.5772/6468.

**Copyright:** © 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License

nonstrategic Inertial Measurement Units (IMU), the low cost of computer systems and the existence of inexpensive radio controlled helicopters have contributed to creating a market of small aerial vehicles within an acceptable range for a wide range of applications.

On the other hand, the lack of safety is mainly caused by two main points: Mechanical and control robustness. The first one is due to the platform being used in building the UAV in order to reduce the cost of the system, which is usually a radio controlled helicopter that requires meticulous maintenance by experts.

The second is due to the complexity of the helicopter dynamics since it is affected by variations in flying altitude, weather conditions and changes in vehicle's configuration (for example: weight, payload or fuel quantity).

These scenarios disrupt the modeling process and, consequently, affect the systematic development of control systems, resulting to tedious and critical heuristic adjustment procedures.

Researchers around the World propose several modeling techniques or strategies for dynamic modeling of helicopters.

Some works on helicopter modeling such as (Godbole et al or Mahony et al,2000), (Gavrilets et al, 2001), (Metler et al and La Civita et al, 2002), and (Castillo et al, 2007) show broad approaches that have been done in this field of engineering. The lack of an identification procedure in some cases and the reduced field of application in others, make sometimes difficult to use them.

In this chapter, not only a modeling is described, but also the identification procedure that has been successfully tested.

The proposed model has been defined by using a hybrid (analytical and heuristic) algorithm based on the knowledge of flight dynamic and by resolving some critical aspects by means of heuristic equations that allow real time simulations to be performed without convergence problems. Evolutionary algorithms have been used for identification of parameters.

The proposed model has been validated in the different phases of the aircraft flight: hovering, lateral, longitudinal or vertical using a Benzin Trainer by Vario which relies on a 1.5 m of main rotor diameter and a payload of about five kilograms.



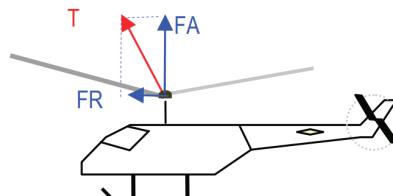
**Figure 1.** Benzin trainer by Vario with GNC (Guidance Navigation & Control) system onboard

A full structure of control has been developed and tested by using a proposed fast design method based on Matlab-Simulink for simulation and adjustment in order to demonstrate the usefulness of the developed tool. The proposed architecture describes low level control (servo actuators level), attitude control, position, velocity and maneuvers. Thus there are commands such as straight flying by maintaining a constant speed or maneuvers such as flying in circles i.e.

The results have confirmed that hybrid model with evolutionary algorithms for identification provides outstanding results upon modeling a helicopter. Real time simulation allows using fast prototyping method by obtaining direct code for onboard controllers and consequently, reducing the risk of bugs in its programming.

## HELICOPTERS FLIGHT PRINCIPLES

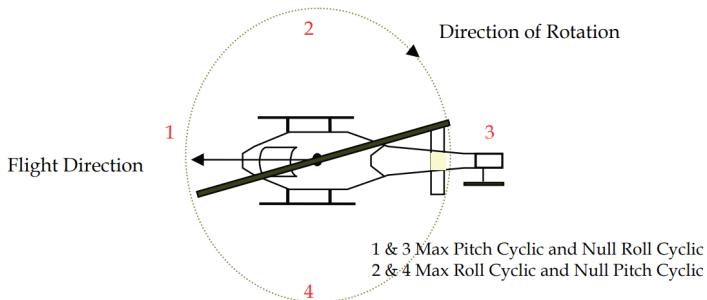
The required thrust for flying helicopters (elevation and advance) is only provided by the main rotor. In order to obtain a comprehensive knowledge about the forces involved in the main rotor, an exhaustive explanation would be required. The forces involved in the main rotor generate twisting and bending in the blades, as well as elevation forces and different kinds of aerodynamic resistance.



**Figure 2.** Basic Helicopter Thrust analysis

In a first approach, only a sustentation and a resistance force isolated from any other influence could be taken into account. The thrust ( $T$ ) generated by the air pressure against the blade has an inclination in relation to the rotation plane. This force can be divided in vertical force ( $F_A$ ) and resistance ( $F_R$ ) that is applied in the horizontal plane against rotation direction (Figure 2).

Helicopters rely on mechanisms to modify the attack angle of the blade in the main rotor. It allows controlling the movement of the fuselage through the inclination of the rotation plane. The fact is that the attack angle of the blade is not constant neither in time nor space. It continuously changes while the blade is rotating as azimuth angle indicates. It can be assumed that the attack angle is the addition of two components: The first is an average attack angle during one complete rotation of the blade, called collective angle. The second component depends on the azimuth angle. When the azimuth angle is  $0^\circ$  or  $180^\circ$ , the blade has the roll cyclic angle. When  $90^\circ$  or  $270^\circ$ , the blade has the pitch cyclic angle (Figure 3).



**Figure 3.** Attack angle during a blade revolution

Using these three signals (collective, roll and pitch cyclic), a pilot is able to control the main rotor. In addition to these signals, the pilot also controls the attack angle of tail rotor blades and the engine throttle.

Typically, radio-controlled helicopters rely upon a commercial control system to maintain the speed of the main rotor constant. The vertical control of the helicopter is done by changing the collective attack angle in the main rotor.

On other hand, the mission of the tail rotor is to compensate the torque that main rotor creates on the helicopter fuselage. The compensation torque can be adjusted by changing the attack angle of its blades. The tail rotor typically requires values between 5 to 30 per cent of the total power of the helicopter.

A lot of physical principles such as ground effect, downwash, flapping or atmospheric effects have to be considered in an in-depth study of helicopter flight dynamics. Taking into account the application scope for the proposed model, which is a small helicopter with small capabilities, no change of air density is considered. Moreover, the blades of the proposed model are also considered as solid bodies.

## MODEL DESCRIPTION

Mathematical models of helicopter dynamics can be either based on analytic or empirical equations. The former applies to the physical laws of aerodynamics, and the latter tries to define the observed behavior with simpler mathematical expressions.

The proposed model is a hybrid analytic-empirical model that harnesses the advantages of both: high-velocity and simplicity of the empirical method, as well as fidelity and the physical meaning of the analytic equations.

## Inputs and Outputs

The proposed model tries to replicate the behavior of a small radio controlled helicopter, therefore the inputs and outputs have been selected as the controls that the pilot relies when using a commercial emitter. Cyclic (Roll and Pitch) and collective controls have been considered as inputs.

**Table 1.** Model inputs and outputs

| Denomination                              | I/O    | Symbol  | Units            |
|---|--------|---|------------------|
| Collective                                | Input  | $\theta_{\text{col}}$   | Degrees          |
| Roll Cyclic                               | Input  | $\theta_{\text{Roll}}$  | Degrees          |
| Pitch Cyclic                              | Input  | $\theta_{\text{Pitch}}$                                       | Degrees          |
| Rotational speed over a main rotor shaft. | Input  | $\omega_z^h$  | Degrees/s        |
| Acceleration (Helicopter reference frame) | Output | $\bar{a}$   | m/s <sup>2</sup> |
| Velocity (Helicopter reference frame)     | Output | $\bar{v}$ (u <sub>a</sub> , v <sub>a</sub> , w <sub>a</sub> ) | m/s              |

Radio controlled helicopters usually rely on electronic systems based on gyroscopes to tail stabilization. Based on this fact, the yaw angle is controlled by giving rotational speed commands. Thus, the yaw rate has been considered as one of the inputs to the model.

It is also common that helicopters rely on a main rotor speed hardware controller. In such manner, the rotor maintains the speed and consequently, the vertical control is then performed by the changing of the collective angle.

The assumption in considering constant speed reduces the complexity of the model and maintains its quality. Furthermore, the use of this hardware controller decreases the number of inputs since no throttle command is required.

Accelerations and rotational speeds are the outputs of the model. Table 1 summarizes the inputs and outputs of the model.

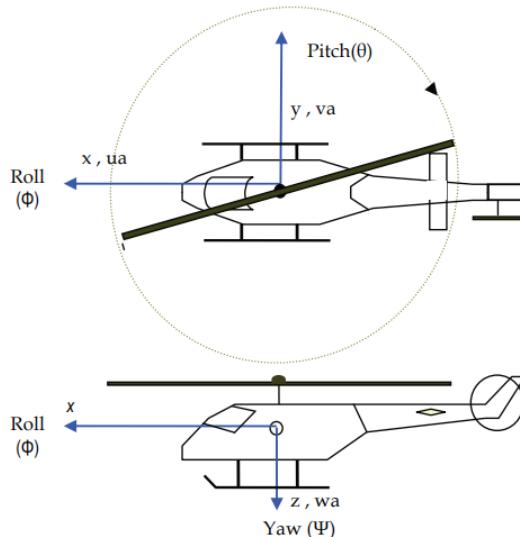
## Model Block Diagram

A block diagram of the proposed model is described in Figure 5. A brief definition of every part is also shown in the following sections.

### Main Rotor

It is modeled with analytic equations, derived from research on full-scale helicopters [Heffley 2000], which can be used to model small helicopters without fly-bars. The iterative algorithm computes thrust and induced wind speed while taking into account geometrical parameters, speed of the blades, commanded collective, roll and pitch cyclic.

One of the most important concepts to be known is the relationship among the Torque, Power and Speed. Such response arises from the induced velocity of the air passing through the disk of the rotor.

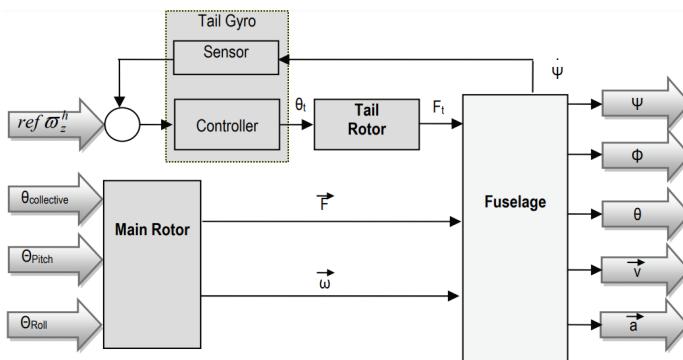


**Figure 4.** Outputs of the model

The airflow goes downward, and due to the action-reaction principle, it generates a vertical force that holds the helicopter in the air. The engine provides the torque to make the blades rotate and create the airflow.

Although there are many factors that make difficult to exactly determination of the relative speed between the helicopter and the airflow through the disk that the rotor creates when rotating, it is possible to work with a first order approach. In this way, it is possible to model using the momentum classic theory and estimating the force and induced velocity using an feedback aerodynamic block. Nevertheless, calculus turns to be difficult because the feedback is highly non-linear.

Another aspect regarding the induced speed is its influence on the surrounding surfaces, thus it can be affected by the ailerons and others fuselage parts and it changes depending on the speed and direction of the flight. In this model, torque and induced speed have been modeled assuming a uniform distribution of the air passing through the rotor disk.

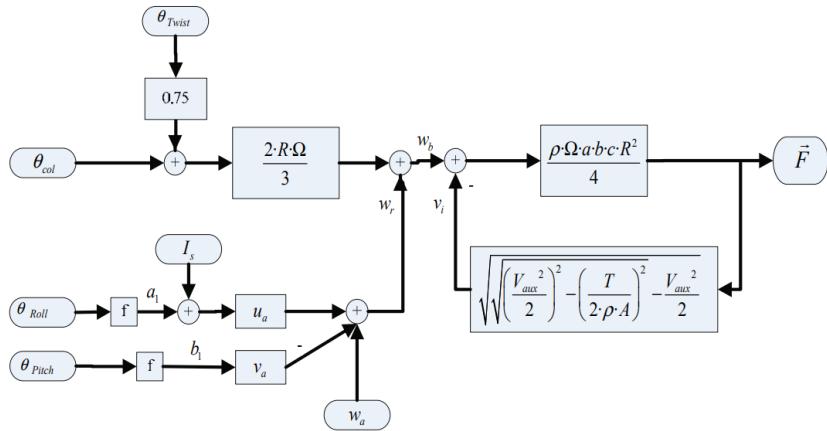


**Figure 5.** Model Block Diagram

The computation of the torque and induced speed is based on the classic momentum theory but using a recursive scheme that allows to reach a fast convergence. The equations used to model the main rotor have two groups of inputs, as Figure 6 shown.

The collective step ( $\theta_{Col}$ ) and the blade torsion ( $\theta_{Twis,t}$ ), compose the first branch. The rotor axis inclination ( $I_s$ ) and the cyclic roll and pitch the second one.

The attack angles  $a_1$  and  $b_1$  are derived from the cyclic roll and pitch references. The wind speed relative to the helicopter is present through its three components:  $u_a$ ,  $v_a$ , and  $w_a$ .



**Figure 6.** Main rotor block diagram

The equations corresponding to this part are:

$$w_r = w_a + (a_1 + I_s) u_a - b_1 \cdot v_a \quad (1)$$

$$w_b = w_r + \frac{2 \cdot \Omega \cdot R}{3} (\theta_{col} + \frac{3}{4} \theta_{Twist}) \quad (2)$$

The output of the block is the rotor's thrust ( $F$ ).  $R$  and  $\Omega$  represent the radius of the rotor and its angular speed respectively;  $\rho$  is the air density, and  $a$ ,  $b$  and  $c$  are geometrical blade factors. The relationship between thrust and angular rates has been derived from observation; therefore the model is also empirical.

$$V_{aux}^2 = u_a^2 + v_a^2 + w_r(w_r - 2v_i) \quad (3)$$

$$T = (w_b - v_i) \frac{\rho \cdot \Omega \cdot R \cdot a \cdot b \cdot c \cdot R}{4} \quad (4)$$

$$v_i^2 = \sqrt{\left(\frac{V_{aux}}{2}\right)^2 - \left(\frac{T}{2 \cdot \rho \cdot \Pi \cdot R^2}\right)^2 - \frac{V_{aux}^2}{2}} \quad (5)$$

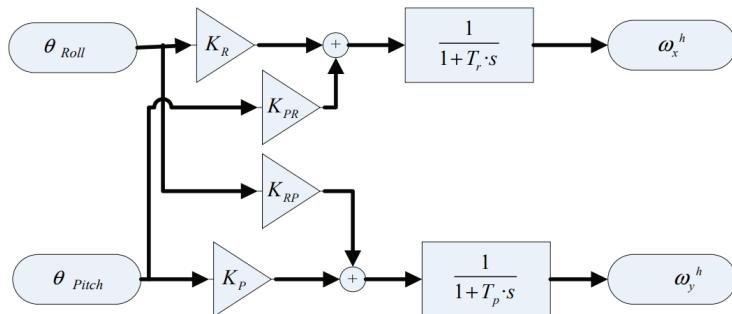
When the helicopter flies close to the ground (distances less than 1.25 times the diameter of the rotor) the ground effect turns to be very important. This effect has been modeled using the parameter  $\eta$  defined in (6) where  $h$  is the distance from the helicopter to the ground.

$$\eta = \frac{h}{2R} \quad (6)$$

In these cases, the thrust is modified using (7) where  $T'_h$  is the resulting thrust after correcting  $T_h$ . Values of  $T_0$ ,  $T_1$  and  $T_2$  have been calculated for no creating a discontinuity when  $h$  is 1.25 times the diameter of the rotor.

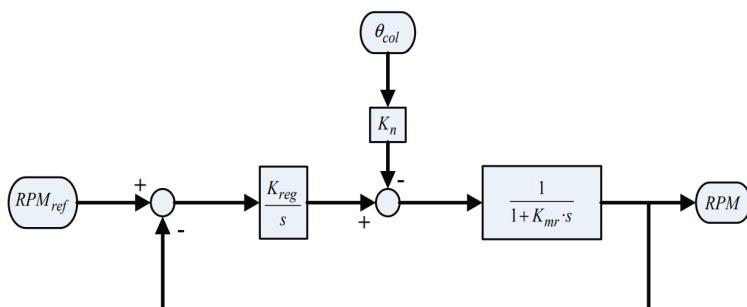
$$T'_h = T_h \left( T_0 + T_1 \eta + T_2 \eta^2 \right) \quad (7)$$

By other hand, the commanded Roll and Pitch cyclic have been considered as reference for rotations in x and y axes considering the helicopter frame. In addition to this, a coupling effect has been considered for simulating real coupling in the helicopter.



**Figure 7.** Roll and Pitch dynamic

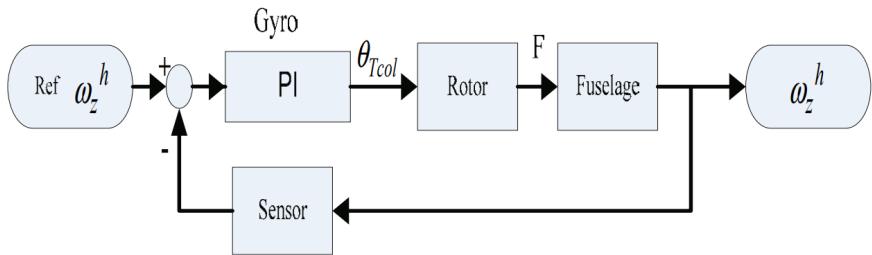
As it was mentioned in the last section, RC helicopters usually rely upon commercial speed controllers in the main rotor. These devices have been modeled using an ideal dynamic response. On the other hand, engine has been modeled as a first order system. Therefore, variations in the speed of the rotor have been considered only due to changes in the collective angle as Figure 8 shows.



**Figure 8.** Engine model

## Tail Rotor

The algorithm for estimating the thrust provided by tail rotor is similar to the main one but only the pitch angle of the blades has been considered as input. This signal is provided by the hardware controller that is in charge of stabilization of the tail. A PI classical controller has been used to model the controller and the sensor has been considered as no dynamic as Figure 9 shows.



**Figure 9.** Tail rotor model

## Fuselage

In a first step, all the forces (main and tail rotor, gravity, friction and aerodynamic resistances) have to be taking into account for computing the movement of the fuselage of the helicopter. After that, accelerations and velocities can be estimated.

Forces due to the aerodynamic frictions are estimated using the relative velocity of the helicopter with respect to the wind applying (8), where Ar is the equivalent area and vw is the wind velocity.

$$F = \frac{1}{2} A_r \rho v_w^2 \quad (8)$$

The resulting equations are summarized in (9).

$$\begin{aligned}
 F_x^h &= G_x^h + \text{Fus}_x^h \\
 F_y^h &= G_y^h + \text{Fus}_y^h \\
 F_z^h &= G_z^h + \text{Fus}_z^h + T^h \\
 M_z^h &= M_d^h + M_t^h + M_{vt}^h
 \end{aligned} \quad (9)$$

Where the prefix Fus means aerodynamic forces on fuselage and T the main rotor thrust. The prefix M means Torque where suffix d denotes main

rotor, the t denotes tail rotor and the v is the effect of the air in the tail of the helicopter. The prefix G means the gravity components.

Once forces and torques have been calculated, accelerations can be obtained and therefore velocities. Once the velocity referred to helicopter frame ( $v^h$ ) is calculated, a transformation is required in order to obtain an absolute value by using (10).

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\theta\sin\phi\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} v_x^h \\ v_y^h \\ v_z^h \end{bmatrix} \quad (10)$$

Flybars are important elements to take into account in the dynamic model. In this work, they have been modeled by using empiric simple models, and the stabilization effect that they produced on the helicopter has been simulated using (11). A more realistic model of flybars can be obtained in [Mettler-2003].

$$\begin{aligned} \ddot{\phi} &= \dot{\phi} - k_1\phi \\ \dot{\theta}' &= \dot{\theta} - k_2\theta \end{aligned} \quad (11)$$

## Identification of the parameters

Once the mathematic equations have been obtained, a procedure to give values to the parameters that appear into the model is required. Some of the parameters can be easily obtained by simple measurements or weighs but some of them turn to be very difficult to obtain or estimate.

Table 2 describes the parameter list to be identified.

Some methods have been studied for performing the parameters identification, such as multi-variable systems procedures (VARMAX), but they are difficult to apply because there is an iterative process into the model. Due to this, the selected method was evolutionary algorithms, after trying unsuccessfully stochastic methods.

Genetic algorithms may be considered as the search for a sub-optimal solution of a specific costbased problem.

The parameters are codified as chromosomes that are ranked with a fitness function, and the best fits are reproduced through genetic operators: crossover and mutation. This process continues until the fitness of the best chromosome reaches a preset threshold.

**Table 2.** Parameter to identify list

| #  | Name        | Meaning  |
|----|-------------|--|
| 1  | TwstMr      | Main rotor blade twist.  |
| 2  | TwstTr      | Tail rotor blade twist.  |
| 3  | KCol        | Collective step gain.  |
| 4  | IZ          | Moment of inertia around the z axis.   |
| 5  | HTr         | Vertical distance from tail rotor to centre of mass of the helicopter.                         |
| 6  | WLVt        | Vertical position of the aerodynamic centre of the tail.                                       |
| 7  | XuuFus      | Frontal effective area of the helicopter.  |
| 8  | YvvFus      | Lateral effective area of the helicopter.  |
| 9  | ZwwFus      | Effective area of the helicopter   |
| 10 | YuuVt       | Frontal area of the tail.  |
| 11 | YuvVt       | Lateral area of the tail.  |
| 12 | Corr1       | Correction parameter for Roll.   |
| 13 | Corr2       | Correction parameter for Pitch.  |
| 14 | Troll       | Time constant for Roll response.   |
| 15 | Tppitch     | Time constant for Pitch response.  |
| 16 | Kroll       | Gain for Roll input.   |
| 17 | Kpitch      | Gain for Pitch input.  |
| 18 | Kyaw        | Gain for Yaw input.  |
| 19 | DTr         | Horizontal distance from centre of the tail rotor to mass centre of the helicopter.            |
| 20 | DVt         | Horizontal distance from the aerodynamic centre of the tail and mass centre of the helicopter. |
| 21 | YMaxVt      | Saturation parameter (no physical meaning).  |
| 22 | KGyro       | Parameter of commercial gyro controller. (gain)  |
| 23 | KdGyro      | Parameter of commercial gyro controller. (derivative)  |
| 24 | Krp         | Cross gain for Roll and Pitch coupling.  |
| 25 | Kpr         | Cross gain for Pitch and Roll coupling.  |
| 26 | OffsetRoll  | Offset of Roll input (trimmer in radio transmitter).   |
| 27 | OffsetPitch | Offset of Pitch input (trimmer in radio transmitter).  |
| 28 | OffsetCol   | Offset of Collective input (trimmer in radio transmitter).                                     |

The main steps of the process for identification using GA's have been:

### **Parameters codification**

The parameters are codified with real numbers, as it is more intuitive format than large binary chains. Different crossover operations have been studied:

- The new chromosome is a random combination of two chains.

|      |    |    |    |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|----|----|----|
| Asc1 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
| Asc2 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Desc | 0  | 11 | 12 | 3  | 14 | 5  | 6  | 17 | 18 | 9  |

- The new chromosome is a random combination of random genes with values in the range defined by the ascendant genes.

|      |     |     |    |    |    |    |     |     |    |    |
|------|-----|-----|----|----|----|----|-----|-----|----|----|
| Asc1 | 0   | 1   | 2  | 3  | 4  | 5  | 6   | 7   | 8  | 9  |
| Asc2 | 10  | 11  | 12 | 13 | 14 | 15 | 16  | 17  | 18 | 19 |
| Desc | 6.5 | 7.6 | 2  | 13 | 4  | 15 | 9.2 | 8.5 | 8  | 19 |

The first operator transmits the positive characteristics of its ascendants while the second one generates diversity in the population as new values appear. In addition to the crossover operator there is a mutation algorithm. The probability of mutation for the chromosomes is 0.01, and the mutation is defined as the multiplication of random genes by a factor between 0.5 and 1.5.

When the genetic algorithms falls into a local minimum, (it is detected because there is no a substantial improvement of the fitness in the best chromosome during a large number of iterations), the probability of mutation have to be increased to 0.1. This improves mutated populations with increased probability of escaping from the local minimum.

### ***Initial population***

The initial population is created randomly with an initial set of parameters of a stable model multiplied by random numbers selected by a Monte-Carlo algorithm with a normal distribution with zero mean and standard deviation of 1.

The genetic algorithm has been tested with different population sizes, between thirty and sixty elements. Test results showed that the bigger population did not lead to significantly better results, but increased the computation time. Using 20 seconds of flight data and a population of 30 chromosomes, it took one hour to process 250 iterations of the genetic algorithm on a Pentium IV processor. Empirical data suggests that 100 iterations are enough to find a sub-optimal set of parameters.

### ***Fitness function***

The fitness function takes into consideration the following state variables: roll, pitch, yaw and speed. Each group of parameters (chromosome) represents a model, which can be used to compute simulated flight data from the recorded input commands. The difference between simulated and real data is determined with a weighted least-squares method and used as the fitness function of the genetic algorithm.

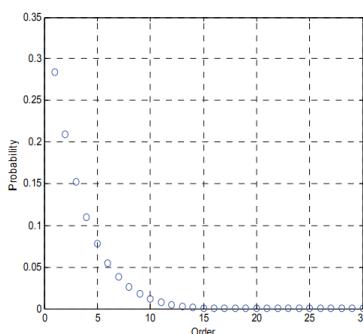
In order to reduce the effect of the error propagation to the velocity due to the estimated parameters that have influence in attitude, the global

process has been decomposed in two steps: Attitude and velocity parameters identification.

The first only identifies the dynamic response of the helicopter attitude, and the genetic algorithm modifies only the parameters related to the attitude. Once the attitude-related parameters have been set, the second process is executed, and only the velocity-related parameters are changed. This algorithm uses the real attitude data instead of the model data so as not to accumulate simulation errors. Using two separate processes for attitude and velocity yields significantly better results than using one process to identify all parameters at the same time.

The parameters related to the attitude process are: TwstTr, IZ, HTr, YuuVt, YuvVt, Corr1, Corr2, Tproll, Tppitch, Kroll, Kpitch, Kyaw, DTr, Dvt, YMaxVt, KGyro, KdGyro, Krp, Kpr, OffsetRoll, OffsetPitch and OffsetYaw. The parameters related to the vehicle's speed are TwstMr, KCol, Xuufus, YvvFus, ZwwFus and OffsetCol.

The fitness functions are based on a weighted mean square error equation, calculated by comparing the real and simulated responses for the different variables (position, velocity, Euler angles and angular rates) applying a weighting factor.



**Figure 10.** Probability function for selection of elements

The general process is described below:

- To create an initial population of 30 elements.
- To perform simulations for every chromosome;
- Computation of the fitness function.
- Classification the population using the fitness function as the index. The ten best elements are preserved. A Monte-

Carlo algorithm with the density function shown in Figure 10. Probability function for selection of elements is used to determine which pairs will be combined to generate 20 more chromosomes. The 10 ‘better’ elements are more likely (97%) to be combined with the crossover operators.

- To repeat from step 2 for a preset number of iterations, or until a preset value for the fitness function of the best element is reached.

### **Data acquisition**

The data acquisition procedure is shown in Figure 11. Helicopter is manually piloted using the conventional RC emitter. The pilot is in charge to make the helicopter performs different maneuvers trying to excite all the parameters of the model. For identification data is essential to have translational flights (longitudinal and lateral) and vertical displacements. All the commands provided by the pilot are gathered by a computer through to a USB port by using a hardware signal converter while onboard computer is performing data fusion from sensors and sending the attitude and velocity estimation to the ground computer using a WIFI link.

In this manner inputs and outputs of the model are stored in files to perform the parameters identification.

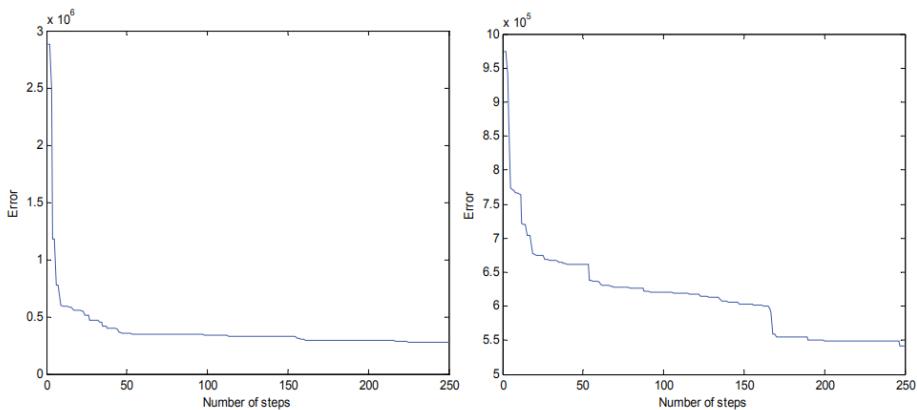


**Figure 11.** Data acquisition architecture

## **IDENTIFICATION RESULTS**

The identification algorithm was executed several times with different initial populations and, as expected, the sub-optimal parameters differ. Some of these differences can be explained by the effect of local minimum. Although the mutation algorithm reduces the probability of staying inside a local minimum, sometimes the mutation factor is not big enough to escape from it.

The evolution of the error index obtained with a least-squares method in different cases is shown in Figure 12. The left graph shows the quick convergence of the algorithm in 50 iterations. On the other hand the right graph shows an algorithm that fell in a local minimum and had an overall lower convergence speed. Around the 50th step a mutation was able to escape the local minimum, and the same behavior is observed in the 160th step.

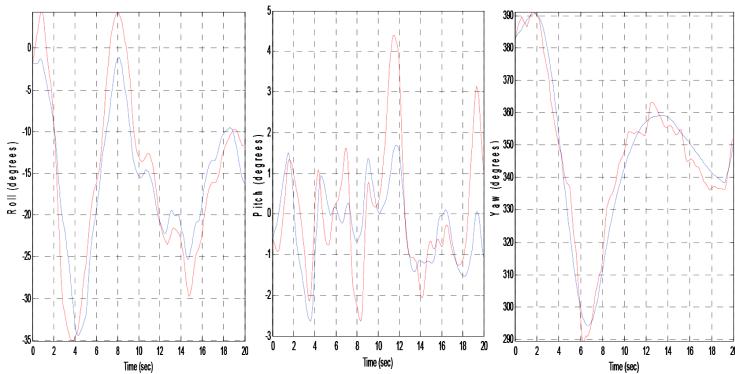


**Figure 12.** Error evolution for two different cases

The result may be used as the initial population for a second execution of both processes. In fact, this has been done three times to obtain the best solution. The analysis of cases where mutation was not able to make the algorithm escaped from local minimum, led to the change of the mutation probability from 0.1 to 1 when detected.

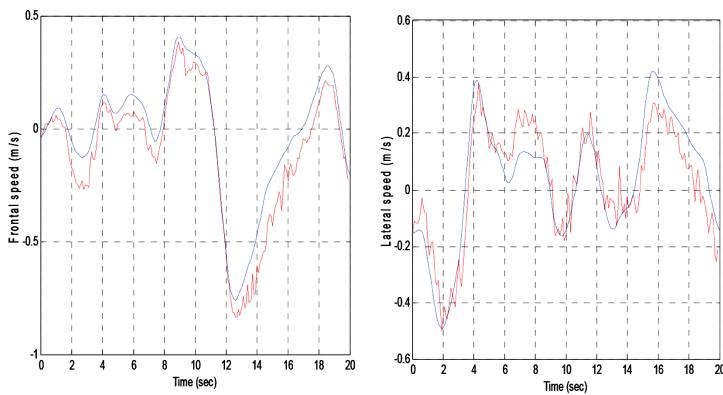
On the other hand, not all the parameters were identified at the same time, actually two iterative processes were used to identify all parameters. The first process used 100 steps to identify the parameters related to the modeling of the helicopter's attitude, beginning with a random variation of a valid solution. The second process preserved the best element of the previous process' population, and randomly mutated the rest. After 100 steps the parameters related to the helicopter's speed were identified.

The simulated attitude data plotted as a blue line against real flight data in red, is shown for roll, pitch, yaw angles in Figure 13. They give you an idea about how simulations follow real tendencies even for small attitude variations.



**Figure 13.** Roll, Pitch and Yaw real vs simulated

On the other hand, the results obtained for the velocity analysis are shown in Figure 14.



**Figure 14.** Velocity Simulation Analysis

The quality of the simulation results is the proof of a successful identification process both for attitude and speed.

The simulated roll and yaw fit accurately the registered helicopter response to input commands. The simulated pitch presents some problems due to the fact that the flight data did not have a big dynamic range for this signal. Nevertheless the error is always below three degrees.

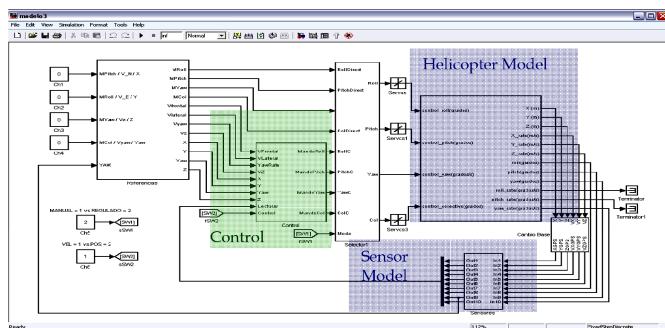
For the simulation of the vehicle's velocity good performance was obtained for both frontal and lateral movement. The unsuccessful modeling of vertical velocity can be linked to the sensors available onboard the

helicopter. Vertical velocity is registered only by the differential GPS, and statistical studies of the noise for this sensor show a standard deviation of 0.18 meters per second. In other words, the registered signal cannot be distinguished from the noise because the registered flight did not have a maneuver with significant vertical speed; therefore modeling this variable is not possible.

It is important to analyze the values of the obtained parameters since the parameters are identified using a genetic algorithm. The dispersion of the value of the parameters for five different optimization processes was analyzed. Most of the parameters converge to a defined value, which is coherent with the parameter's physical meaning, but sometimes, some parameters were not converged to specific values, usually when no complete set of flights was used. Thus, if no vertical flights were performed, the parameters regarding vertical movements turned to be with a great dispersion in the obtained values.

This conclusion can be extended to different optimization processes for different groups of flight data. In other words, several flights should be recorded, each with an emphasis on the behaviors associated to a group of parameters. With enough flight data it is trivial to identify all the parameters of the helicopter model.

## CONTROL PROTOTYPING



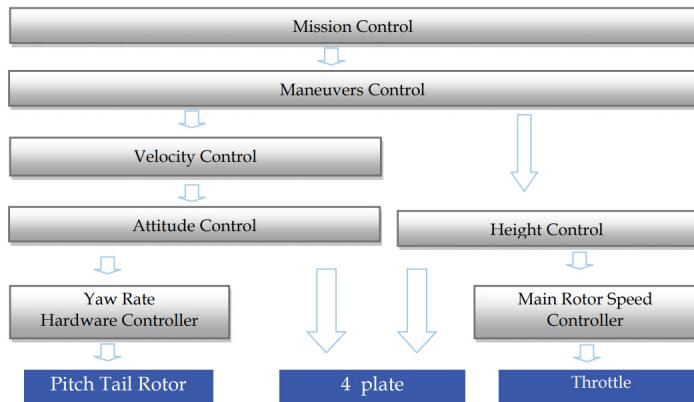
**Figure 15.** Control Developing template

In order to develop and test control algorithms in a feasible way, the proposed model has been encapsulated into a Matlab-Simulink S-Function, as a part of a prototyping template as Figure 15 shows. Others modules have been used for performing realistic simulations, thus a sensor model of GNC

systems has also been created. Auxiliary modules for automatic/manual switching have been required for testing purposes.

The proposed control architecture is based on a hierachic scheme with five control levels as Figure 16 shows. The lower level is composed by hardware controllers: speed of the rotor and yaw rate. The upper level is the attitude control. This is the most critical level for reaching the stability of the helicopter during the flight. Several techniques have been tested on it (classical PI or fuzzy controllers). The next one is the velocity and altitude level. The fourth is the maneuver level, which is responsible for performing a set of pre-established simple maneuvers, and the highest level is the mission control.

Following sections will briefly describe these control levels from the highest to the lowest one.



**Figure 16.** Control Architecture

## Mission control

In this level, the operator designs the mission of the helicopter using GIS (Geographic Information System) software.

These tools allow visualizing the 3D map of the area by using Digital Terrain Model files and describing a mission using a specific language. (Gutiérrez et al -06). The output of this level is a list of maneuvers (parametric commands) to be performed by the helicopter.

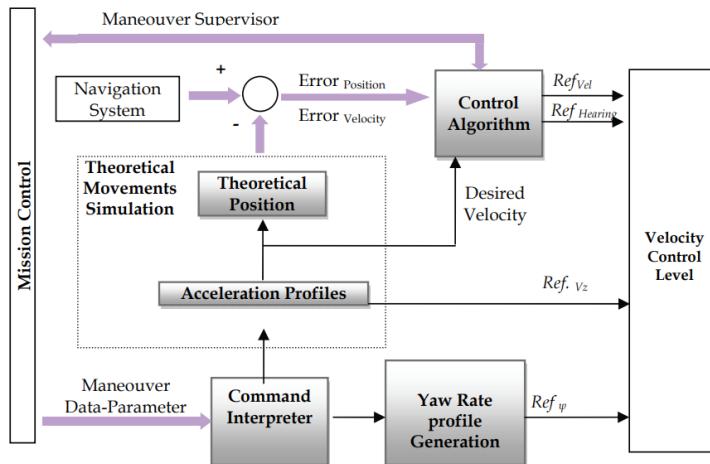
## Maneuver Control

This control level is in charge of performing parametric manoeuvres such as

flight maintaining a specific velocity during a period of time, hovering with a fix of changing yaw, forward, backward or sideward flights and circles among others. The output of this level are velocity commands.

Internally, this level performs a prediction of the position of the helicopter, applying acceleration profiles (Figure 17).

Velocity and heading references are computed by using this theoretical position and the desired velocity in addition to the real position and velocity obtained from sensors.

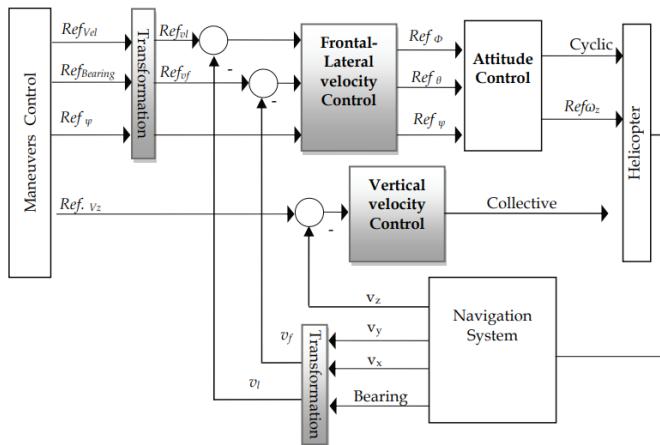


**Figure 17.** Maneuvers Control Scheme

A vertical control generates references for altitude control and Yaw control. In manoeuvres that no high precision of the position is required, i.e. forwarded flights, the position error is not taken into account, because the real objective of the control is maintain the speed. This system allow manage a weighting criteria for defining if the main objective is the trajectory or the speed profile. The manoeuvres that helicopter is able to perform are an upgradeable list.

## Velocity Control

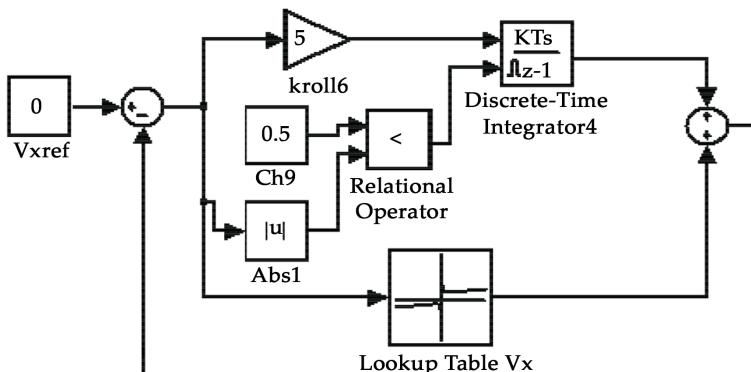
The velocity control is in charge of making the helicopter maintains the velocity that manoeuvres control level computes. The velocity can be referred to a ENU (East-North-Up) frame or defined as a module and a direction. Therefore, a coordinate transformation is required to obtain lateral and frontal velocities ( $v_l$ ,  $v_f$ ) that are used to provide Roll and Pitch references. This transformation is very sensitive to the yaw angle estimation.



**Figure 18.** Velocity control structure

Control maneuver has also to provide with the yaw reference due to the capability of helicopters for flying with derive (different bearing and heading).

Vertical velocity is isolated from the horizontal because it is controlled by using the collective command.



**Figure 19.** Velocity control example

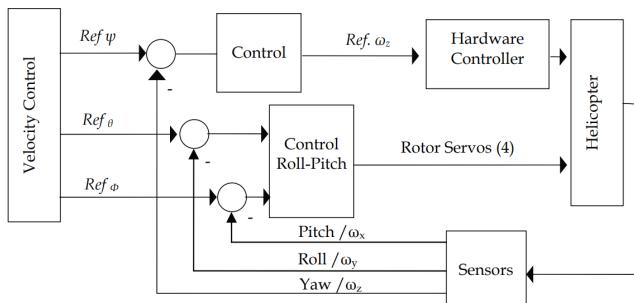
Concerning to the control algorithms used to test the control architecture, two main problems have been detected and solved: The first one is based on the fact that the speed is very sensitive to the wind and payload. Due to this, a gain-scheduller scheme with a strong integration efect has been required in PI algorithm. (Figure 19)

The second one arises from the solution of the first problem. Thus, when operator performs a manual-automatic swiching by using channel nine of the emitter, integral action needs to be reset for reducing the effect of the error integration when manual mode is active.

## Attitude Control

The attitude control is in charge of providing with commands to the servos (or hardware controllers). Several control techiques were tested in this level, but probably fuzzy controllers turned out to have the best performance. Figure 20 shows the proposed architecture and Figure 21 the control surfaces used by fuzzy controller.

As it can be observed, the Yaw have been isolated from the multivariable Roll-Pitch controller with a reasonable quality results.

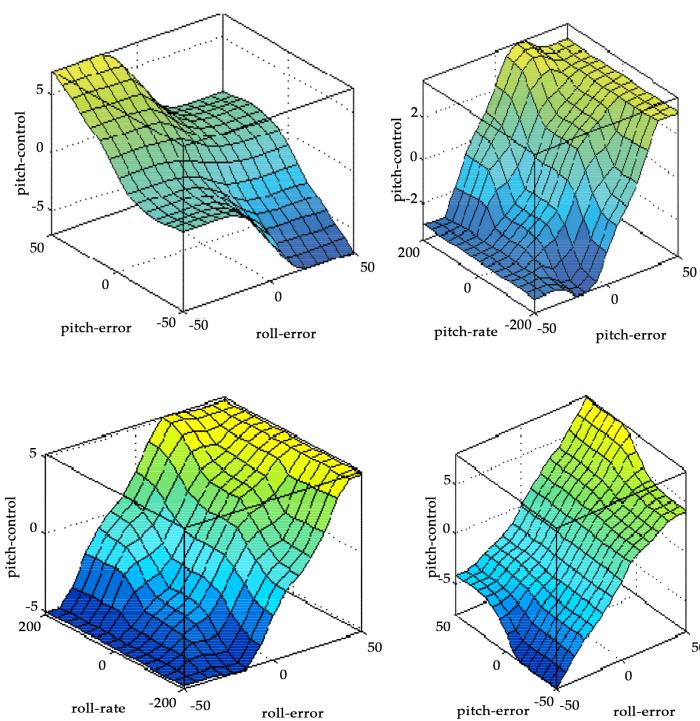


**Figure 20.** Attitude control structure

Once the control structure has been designed, the first step is to perform simulations in order to analyze if the control fulfills all the established requirements. Considering the special characteristics of the system to control, they main requirements to be taken into account are:

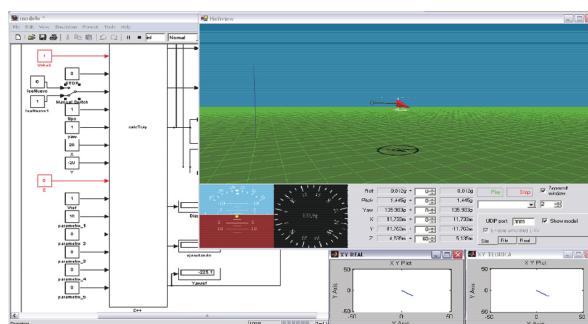
- No (small) oscillations for attitude control. In order to transmit a confidence in the control system to the operator during the tests.
- No permanent errors in velocity control. This is an important concept to consider for avoiding delays in long distance missions.
- Adjustable fitness functions for maneuvers control. This allows characterizing the mission where position or velocity is the most important reference to keep.

The operator relies upon all the tools that Matlab-Simulink provides and an additional 3D visualization of the helicopter as Figure 22 shows for test performing.

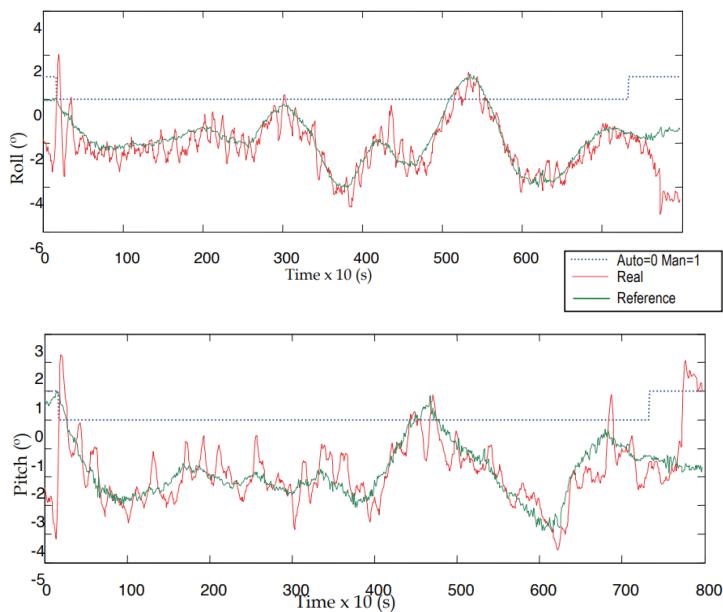


**Figure 21.** Control surfaces for attitude level

The controller block has to be isolated for automatic C codification by using the embedded systems coder of Matlab-Simulink when simulation results are satisfactory. Then, the obtained code can be loaded into onboard computer for real testing (Del-Cerro 2006).



**Figure 22.** Simulation framework



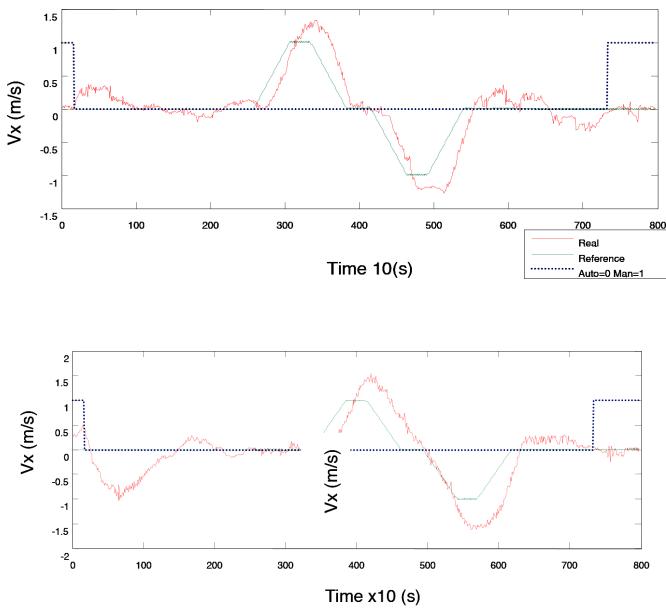
**Figure 23.** Roll and pitch Control results during real flight

## CONTROL RESULTS

Figure 23 shows the results obtained for attitude control during an eighty-seconds real flight with the Vario helicopter.

Lines in red show the real attitude of the helicopter while green ones are the references provided by the velocity control level.

Dotted blue line indicates when the helicopter is in manual (value of 1) or automatic (value of 0). It can be observed that, when helicopter is in manual mode, the red line does not follow the references of the automatic control, because the pilot is controlling the helicopter. Figure 24 shows the obtained results for velocity control. The same criteria that in last graphic has been used, thus red line indicates real helicopter response and green means references from maneuvers control level. Dotted blue line is used to show when the helicopter flies autonomously or remote piloted. Small delays can be observed.

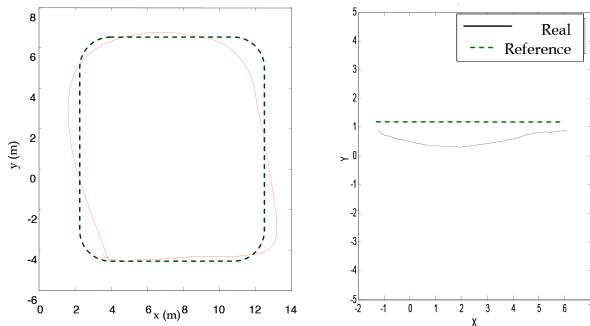


**Figure 24.** Velocity control results

Figure 25 shows two examples of the maneuvers control performance. The right side graph shows the position when helicopter is moving in a straight line. It can be observed that maximum error in transversal coordinates is less than one meter. This result can be considered as very good due to is not a scalable factor. The precision of the position control reduces the error when trajectory is close to the end by weighting the error position against the velocity response.

On the other hand, the left side shows a square trajectory. The speed reference performs smooth movements and therefore the position control is less strong than in the first case. Even in this case, position errors are smaller than two meters in the worst case.

In general, a high precision flight control has been developed, because sub-metric precision has been reached during the tests with winds slower than twenty kilometers per hour.



**Figure 25.** Maneuvers control results

## FINAL OBSERVATIONS AND FUTURE WORK

The first part of this chapter describes a procedure for modeling and identification a small helicopter. Model has been derived from literature, but changes introduced in the original model have contributed to improve the stability of the model with no reduction of the precision.

The proposed identification methodology based on evolutionary algorithms is a generic procedure, having a wide range of applications. The identified model allows performing realistic simulations, and the results have been validated.

The model has been used for designing real controllers on a real helicopter by using a fast prototyping method detailed in section 4.

The model has confirmed a robust behavior when changes in the flight conditions happen. It does work for hover and both frontal and lateral non-aggressive flight. The accuracy and convenience of a parametric model depends largely on the quality of its parameters, and the identification process often requires good or deep knowledge of the model and the modeled phenomena. The proposed identification algorithm does away with the complexity of model tuning: it only requires good-quality flight data within the planned simulation envelope. The genetic algorithm has been capable of finding adequate values for the model's 28 parameters, values that are coherent with their physical meaning, and that yields an accurate model.

It is not the objective of this chapter to study what the best control technique is. The aim is to demonstrate that the proposed model is good enough to perform simulations valid for control designing and implementation using

an automatic coding tool. Future work will focus on the following three objectives:

- To develop a model of the engine in order to improve the model behavior for aggressive flight maneuvers.
- To compare this model with others in the literature in a wide range of test cases.
- To validate the proposed model with different helicopters.

## REFERENCES

1. Castillo P, Lozano R, Dzul A. (2005) Modelling and Control of Mini-Flying Machines. Springer Verlag London Limited 2005. ISBN:1852339578.
2. Del-Cerro J, Barrientos A, Artieda J, Lillo E, Gutiérrez P, San- Martín R, (2006) Embedded Control System Architecture applied to an Unmanned Aerial Vehicle International Conference on Mechatronics. Budapest-Hungary
3. Gavrilets, V., E. Frazzoli, B. Mettler. (2001) Aggresive Maneuvering of Small Autonomous Helicopters: A. Human Centerred Approach. The International Journal of Robotics Research. Vol 20 No 10 , October 2001. pp 795-807
4. Godbole et al (2000) Active Multi-Model Control for Dynamic maneuver Optimization of Unmanned Air Vehicles. Proceedings of the 2000 IEEE International conference on Robotics and Automation. San Francisco, CA. April 2000.
5. Gutiérrez, P., Barrientos, A., del Cerro, J., San Martin., R. (2006) Mission Planning and Simulation of Unmanned Aerial Vehicles with a GIS-based Framework; AIAA Guidance, Navigation and Control Conference and Exhibit. Denver, EEUU, 2006.
6. Heffley R (1988). Minimum complexity helicopter simulation math model. Nasa center for AeroSpace Information . 88N29819.
7. La Civita M, Messner W, Kanade T. Modeling of Small-Scale Helicopters with Integrated First-Principles and System-Identification Techniques. American Helicopter Society 587h Annual Forum, Montreal, Canada, June 11-13,2002.
8. Mahony-R Lozano-R, Exact Path Tracking (2000) Control for an Autonomous helicopter in Hover Manoeuvres. Proceedings of the 2000 IEEE International Conference on Robotics & Automation. San Francisco, CA. APRIL 2000. P1245-1250
9. Mettler B.F., Tischler M.B. and Kanade T.(2002) System Identification Modeling of a ModelScale Helicopter. T. Journal of the American Helicopter Society, 2002, 47/1: p. 50-63. (2002).
10. Mettler B, Kanade T, Tischler M. (2003) System Identification Modeling of a Model-Scale Helicopter. Internal Report CMU-RI-TR-00-03.

# **CHAPTER**

# **7**

## **Contour Based Path Planning with B-Spline Trajectory Generation for Unmanned Aerial Vehicles (UAVs) over Hostile Terrain**

---

**Ee-May Kan<sup>1</sup> , Meng-Hiot Lim<sup>1</sup> , Swee-Ping Yeo<sup>2</sup> , Jiun-Sien Ho<sup>3</sup> ,  
Zhenhai Shao<sup>4</sup>**

<sup>1</sup> Nanyang Technological University, Singapore City, Singapore

<sup>2</sup> National University of Singapore, Singapore City, Singapore

<sup>3</sup> Temasek Polytechnic, Singapore City, Singapore

<sup>4</sup> University of Electronic Science Technology of China, Chengdu, China.

### **ABSTRACT**

This research focuses on trajectory generation algorithms that take into account the stealthiness of autonomous UAVs; generating stealthy paths

---

**Citation:** E. Kan, M. Lim, S. Yeo, J. Ho and Z. Shao, "Contour Based Path Planning with B-Spline Trajectory Generation for Unmanned Aerial Vehicles (UAVs) over Hostile Terrain," *Journal of Intelligent Learning Systems and Applications*, Vol. 3 No. 3, 2011, pp. 122-130. doi: 10.4236/jilsa.2011.33014.

**Copyright:** © 2011 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

through a region laden with enemy radars. The algorithm is employed to estimate the risk cost of the navigational space and generate an optimized path based on the user-specified threshold altitude value. Thus the generated path is represented with a set of low-radar risk waypoints being the coordinates of its control points. The radar-aware path planner is then approximated using cubic B-splines by considering the least radar risk to the destination. Simulated results are presented, illustrating the potential benefits of such algorithms.

**Keywords:** Unmanned Aerial Vehicles (UAVs), Radar, Path Planning, B-Splines

## INTRODUCTION

Unmanned aerial vehicles (UAVs) are increasingly being used in real-world applications [1-3]. They are typically surveillance and reconnaissance vehicles operated remotely by a human operator from a ground control station; they have no on-board guidance capabilities that give them some level of autonomy, for example, to re-plan a trajectory in the event of a change in the environment or mission. With such rudimentary capabilities, only simple tasks can be accomplished and the operation is also limited to simple or uncomplicated situations, typically in well-characterized environments. It is useful to endow UAVs with more advanced guidance capabilities, in particular capabilities that increase the vehicle's autonomy to allow for more complex missions or tasks. Currently operating UAVs use rudimentary guidance technologies, such as following pre-planned or manually provided waypoints. Over the years, advances in software and computing technology have fuelled the development of a variety of new guidance methodologies for UAVs. The availability of various guidance technologies and understanding of operational scenarios create opportunities towards refining and implementing such advanced guidance concepts. The basic difficulties include partially known and changing environment, extraneous factors, such as threats, evolving mission elements, tight timing and positioning requirements. Moreover, these must be tackled, while at the same time, explicitly accounting for the actual vehicle manoeuvring capabilities, including dynamics and flight-envelope constraints. The term flight envelope refers to the parameters within which an aerial vehicle may be safely flown under varying, though expected, wind speed, wing loading, wind shear, visibility and other flight conditions without resorting to extreme control measures such as abnormal spin, or stall recovery, or crash

landing. These aerial vehicles are usually mobilized to carry out critical missions in high-risk environments, particularly in situations where it may be hazardous for human operators. However, such missions demand a high level of stealth, which has a direct implication on the safety and success of the mission. Therefore it is important to minimize the risk of detection or more specifically, the probability of detection of the UAVs by enemy radars.

There has been extensive research in the area of path planning especially in the artificial intelligence, and optimization with most restricted to two-dimensional (2D) paths [4,5]. Different conventional approaches have been developed to solve the path planning problem, such as the cell decomposition, Djikstra's algorithm, road map and potential field [6-9]. Sleumer and Tschichold [6] present an algorithm for the automatic generation of a map that describes the operation environment of a building consisting of line segments, representing the walls as an input. The algorithm is based on the exact cell decomposition approach and generates a connectivity graph based on cell indices. However, for successful implementation of exact cell decomposition, it is required that the geometry of each cell be simple. Moreover, it is important to test the adjacency of two cells to find a path crossing the portion of the boundary shared by the two cells. Both exact cell decomposition and approximate cell decomposition methods are accurate, yet may be computationally expensive when used in a terrain environment since numerous obstacles at varying elevations could be found. They are only effective when the environment contains a countable number of obstacles.

Another technique used to effectively represent the alternate paths generated by a path planner is to use Bsplines. B-splines allow a parametric curved path to be described by only a few control points rather than the entire curve segmented which could be as many as thousands of sections, depending on the length and curvature of the path. Recent methods include modelling the trajectory using splines based on elastic band theory [10,11], and interactive manipulation of control points for spline trajectory generation [12,13]. A series of cubic splines was employed in [14] to connect the straight line segments in a near-optimal manner, whereas the algorithm presented in [15] yields extremal trajectories that transition between straight-line path segments smoothly in a time-optimal fashion.

Previous efforts [16,17] have concentrated on path planning over a hostile radar zones based on the amount of energy received by enemy radar at several locations. In this paper, we investigate on the number and placement of control points within a hostile region by considering the complexity of

the terrain. We take into account the path optimality within a hostile region by considering the complexity of the terrain. We propose an efficient algorithm to identify the waypoints based on the topographic information of the enemy terrain. The waypoint generation process is based on the user-specified threshold flying altitude. The threshold altitude to avoid radar detection and terrain collisions is determined based on the knowledge of the enemy terrain. The navigational path between the waypoints is considered when minimizing the exposure to a radar source. Additionally, the generated trajectory is of minimal length, subject to the stealthy constraint and satisfies the aircraft's dynamic constraints.

Details on the formulation of the problem are presented in this paper. We model this problem as described in the following section and adopt a heuristic-based approach to solve it. The rest of this paper is organized as follows: Section 2 gives a description of the problem model used for measuring stealth in this work and illustrates the calculation of the radar risks. Section 3 demonstrates the details of the problem formulation and Section 4 presents the solution approach of the route planner, substantiated with simulated results. The conclusion and future work are presented in Section 5.

## MODELLING

In this section, the model we used throughout this work is presented. The integrated model of the UAV and radar has the following features. The Radar Cross Section (RCS) of the UAV depends on both the aspect and bank angles whereas the turn rate of the UAV is determined by its bank angle. Hence, the RCS and aircraft dynamics are coupled through the aspect and bank angles.

### Measuring Stealth Based on Aircraft Model

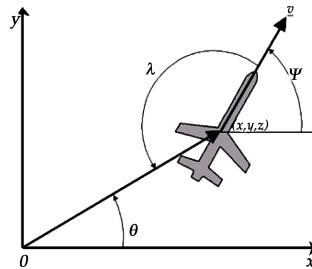
The bank-to-turn aircraft is assumed to move in a horizontal plane at a constant altitude according to the equations [18],

$$x = v \cos \varphi, \quad y = v \sin \varphi, \quad \varphi = \frac{u}{v}, \quad |u| < U \quad (1)$$

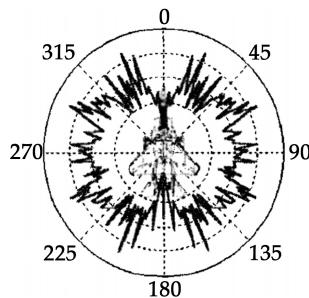
where  $x$  and  $y$  are the Cartesian coordinates of the aircraft,  $\varphi$  is the heading angle as shown in Figure 1,  $v$  is the constant speed,  $u$  is the input signal and the acceleration is normal to the flight path vector, followed by  $U$  is the maximum allowable lateral acceleration. Figures 2 and 3 show the dependence of RCS on aspect and bank angles.

$$\theta = \arctan\left(\frac{y}{x}\right), \lambda = \theta - \varphi + \pi,$$

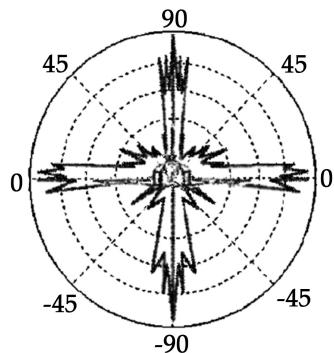
$$\phi = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \quad (2)$$



**Figure 1.** Aircraft position, velocity, azimuth, heading, and aspect angles.



**Figure 2.** RCS as a function of aspect angle.



**Figure 3.** RCS as a function of bank angle.

be the azimuth, aspect, and elevation angles, respectively, where  $z$  is the aircraft altitude. Let the bank angle  $\mu$  be given by

$$u = \arctan\left(\frac{u}{g}\right) \quad (3)$$

where  $g$  is the acceleration of gravity. We model the RCS of the aircraft as function of the aspect angle  $\lambda$ , the elevation angle  $\varphi$ , and the bank angle  $\mu$ , so that

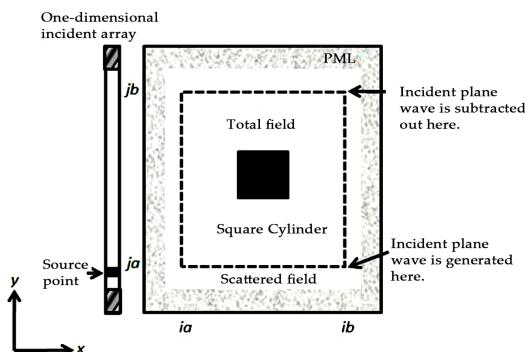
$$\text{RCS} = \text{RCS} = \sigma(\lambda, \varphi, \mu) \quad (4)$$

The RCS obtained from Equation (4) is used as an estimate value in Equation (6) for measuring stealth in this work. As an example, real aircraft RCS measurements as functions of aspect and bank angles are shown in Figures 3 and 4 [18]. The following section illustrates the calculation of the cost associated to radar risk that will be undertaken by the UAV.

## Radar Model

The radar model in this work will be presented in terms of its inputs (aircraft range and RCS) and output (an estimate of the probability that an aircraft can be tracked for an interval of time). For the sake of simplicity, we assume that the radar is located at  $(x, y, z)$  of the navigational space. Let

$$R = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad (5)$$



**Figure 4.** Square cylinder for RCS modelling.

be the aircraft range from the enemy radar to the aircraft. The radar detects the aircraft by receiving a sequence of radio frequency pulses reflected from

it at fixed observation times. To generate a stealthy path for the UAV, we have to take into account the energy reflected to enemy radar site as the enemy radar tracks the location, speed, and direction of an aircraft. The range at which radar can detect an object is related to the power transmitted by the radar, the fidelity of the radar antenna, the wavelength of the radar signal, and the RCS of the aircraft [19]. The RCS is the area a target would have to occupy to produce the amount of reflected power that is detected back at the radar. RCS is integral to the development of radar stealth technology, particularly in applications involving aircraft. For example, a stealth aircraft which is designed to be undetectable will have design features that give it a low RCS. Low RCS offers advantages in detection range reduction as well as increasing the effectiveness of decoys against radar-seeking threats.

The size of a target's image on radar is measured by the RCS and denoted by the symbol  $\sigma$  and expressed in square meters. The azimuth and range detected by the radar serve as the inputs to a tracking system, typically based on one or more Kalman filters.

The purpose of the tracking system is to provide a predicted aircraft position and velocity so that a decision can be made to launch a missile and guide it to intercept. RCS modelling and reduction is crucial in designing stealth weapon systems. We make use of Finite-Difference Time-Domain (FDTD) Method technique in RCS modelling. The numerical technique is general, geometry-free, and can be used arbitrarily for any target, within the limitations of computer memory and speed. Figure 4 shows the geometric surface for the RCS simulation with the FDTD algorithm.

The FDTD-based package was supplied in [20] for the RCS prediction of various simple canonical targets. The complete RCS signature of a target is described by the frequency response of the target illuminated by plane waves from all possible physical angles. Realistic targets are frequently very large and are often largely made up of highly conductive materials.

The FDTD-based RCS prediction procedure is as follows:

- The target, located in the computational space, is illuminated by a Gaussian-pulse plane wave from a given direction;
- The scattered fields are simulated inside the computational volume until all transients dissipate;
- The time-domain far fields in the spherical coordinates are then extrapolated by using equivalent current via the near-field to far-field routine, over a closed, virtual surface surrounding the target.

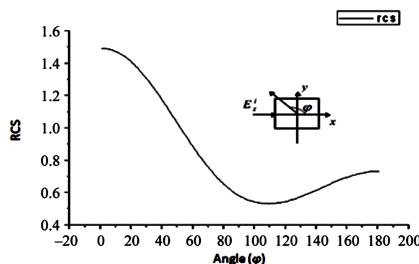
For either vertical or horizontal planes, co-polarized or cross-polarized RCS behaviour is computed after the simulation for a given frequency range. Figure 5 illustrates the RCS behaviour with respect to the azimuth angle based on the RCS modelling described above.

Subsequently the computation of the energy received by a radar is given by the radar range equation [21]

$$S \cong \frac{P_{\text{avg}} G \sigma t_{\text{tot}} A_e}{(4\pi^2) R^4} \quad (6)$$

where  $S$  is the signal energy received by the radar,  $P_{\text{avg}}$  is the average power transmitted by the radar,  $G$  is the gain of the radar antenna,  $\sigma$  is the radar cross section of the target,  $A_e$  is the effective area of the radar antenna,  $t_{\text{tot}}$  is the time the radar antenna is pointed at the target, and  $R$  is the range to the target. Every radar has a minimum signal energy that it can detect,  $S_{\text{min}}$ . This minimum signal energy determines the maximum range ( $R_{\text{max}}$ ) at which the radar can detect a given target.

$$R_{\text{max}} \cong \sqrt[4]{\frac{P_{\text{avg}} G \sigma A_e t_{\text{tot}}}{(4\pi^2) S_{\text{min}}}} \quad (7)$$



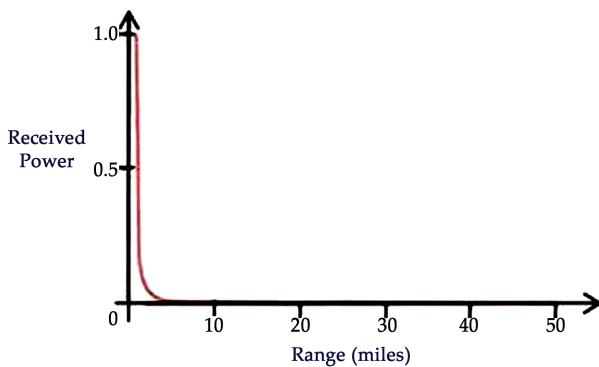
**Figure 5.** RCS behaviour with respect to the azimuth angle.

According to Equation (7), the distance at which a target can be detected for given radar configuration varies with the fourth root of its RCS. Figure 6 gives some understanding of just how little radar power is typically reflected back from the target and received by the radar. In this case, the target presents the same aspect to the radar at ranges from 1 to 50 miles. At a range of 50 miles, the relative power received by the radar is only  $1.6 \times 10^{-5}\%$  of the strength at one mile. This diagram graphically illustrates how significant the effect of energy dissipation is with distance, and how sensitive radars

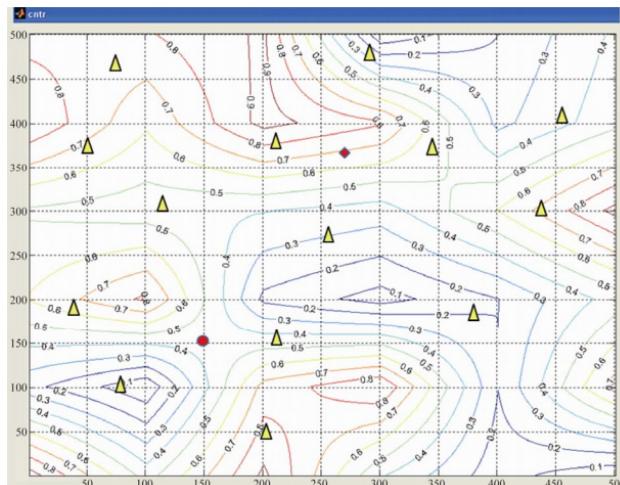
must be to detect targets at even short ranges. Hence the cost associated to radar risk is based on a UAV's exposure to enemy radar. In this work, the cost of radar risk involves the amount of energy received by enemy radar at several locations based on the distance between the UAV and the radar. We assume that the UAV's radar signature is uniform in all directions and is proportional to  $1/R^4$  (where R is the distance between the UAV and the enemy radar); the UAVs are assigned simplified flight dynamics and performance constraints in two-dimensions; and all tracking radars are given simplified detection properties. Based on the assumptions stated above, we classify the navigational space into different levels of risk as described in the following chapter. The next step is to compute a stealthy path, steering the UAV clear and around known radar locations. The generated path should carry minimal radar risks, and adhere to physical and motion constraints. Also, the distance between the UAV and the radars should be maximized in order to minimize the energy received at the radar.

## PROBLEM FORMULATION

To describe the problem, consider the contour based navigational space shown in Figure 7. The objective is to find a path which minimizes the UAV's exposure to radar sites (small triangles) from the current UAV position (circle) to the target location (star). In the present study, it is assumed that the hostile radar zones are known beforehand. The contours are used to denote elevation and depth on maps. From these contours, a sense of the general terrain can be determined. The details of the radar zone can be acquired using satellite data or from surveillance data. The start point and the end point of the flight are known and it is required to find a feasible path connecting these two points. The threshold altitude value is determined by the user which assumes knowledge of the enemy terrain. By flying at a userspecified threshold altitude value, the aircraft can take advantage of terrain masking and avoid detection by enemy radars. The aircraft works by transmitting a radar signal towards the ground area and the signal returns can then be analysed to see how the terrain ahead varies, which can then be used by the aircraft's autopilot to specify the threshold altitude value. The specified threshold allows the aircraft to fly at reasonably constant height above the earth so as to avoid the radar detection. In this work, the path of the UAV is smoothed using cubic Bsplines, which is controlled by manipulating a number of control points. The generated path should carry minimal radar risks, and adhere to physical and motion constraints.



**Figure 6.** Reduction in the strength of target echoes with range.



**Figure 7.** Contour based navigational space with known radar sites.

Given a single radar located at the origin and a single aircraft travelling from initial point to the destination, Pachter [22] showed that an objective function for minimizing cost, is

$$J = \int_0^l \frac{1}{R^4(t)} dt \quad (8)$$

where  $v$  is the constant speed of the aircraft and  $l$  is the path length. A closed form solution to this problem was obtained using the calculus of variations, with the limitation that the aircraft traverses an angle with respect

to the radar of less than  $60^\circ$ . Beyond this limit, the optimal path length is infinite but the cost remains finite. The objective cost function Equation (8), is augmented to include the rest of the radar at the navigational space, giving

$$J = \int_0^L \sum_{i=1}^m \frac{\alpha_i}{R^4(t)} dt \quad (9)$$

The cost associated with radar risk involves the amount of energy  $\alpha_i$ , received by enemy radar based on the distance from the UAV to the radar. We first look for appropriate waypoints that the UAV should traverse in planning a path which minimizes the radar exposure.

## GENERATION OF NODES

The procedure starts by requesting from the user the detection altitude or maximum elevation desired which assumes knowledge of the enemy terrain. Nodes below the detection altitude are extracted and used as base in the generation of straight line segments. The nodes above the detection altitude are discarded, given that at those elevations the UAVs could be detected. Appropriate value for the parameter is based on the user's knowledge of: the terrain; the operation of the UAVs; the purpose, logistics, and safety of the mission. This means that the user's input is a determinant in the quality of the resulting path. As shown in Figure 8, the initial nodes can be searched by specifying the detection altitude for the UAV. The next step is to make use of heuristic search algorithm to compute a stealthy path, steering the UAV clear and around known radar locations.

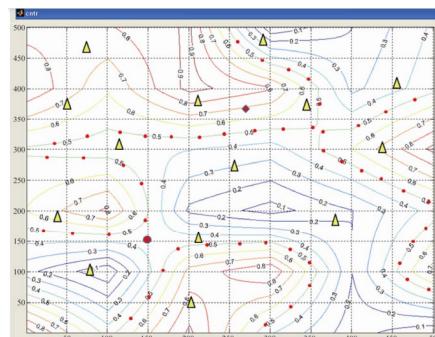
## IMPLEMENTATION

### Generation of Nodes

This search algorithm works by starting at UAV's present position (red circle). It adds the heuristic function value and determines the expansion order of nodes. We make use of a distance-plus-cost heuristic function,  $f(n)$  to determine the order in which the search visits nodes in the space. We assume that the UAV's radar signature is uniform in all directions and is proportional to  $1/R^4$  where  $R$  is the aircraft range. We modify the evaluation function by adding the cost associated with radar risk to the  $f(n)$  as follows:

$$f(n) = w \times g(n) + (1-w) \times h(n) + \sum_{i=0}^m \frac{c_i}{R_i^4} \quad (10)$$

where  $w$  is a value in the range  $[0, 1]$ ;  $g(n)$  is the pathcost function which shows the intensity between the current node and node  $n$ ;  $h(n)$  is the distance value which is estimated from node  $n$  to the goal node;  $c_i$  is relative to  $R_i$  and  $m$  is the number of enemy radars located at the navigational space. The  $h(n)$  plays a significant role in the search process when  $w$  value is smaller; the number of vertices to be explored would be decreased. The choice of  $w$  between 0 and 1 gives the flexibility to place weight on exposure to threats or fuel expenditure depending on the particular mission scenario. For example when  $w = 0$ , the search process would proceed from the starting location to the target location by exploring the least number of vertices. However the generated path is not stealthy and may cause UAV to enter high radar risk region. A  $w$  value closer to 0 would result in the shortest paths, with little regard for the exposure to enemy radar, while a value closer to 1 would result in paths that avoid threat exposure at the expense of longer path length. For this case, the cost weighting parameter  $w$  was chosen to give a reasonable tradeoff between proximity of the path to threats and the path length. The evaluation function, (10) is tested empirically and compared in terms of the number of nodes visited. During the test, the cost weighting parameter  $w$  is varied from 0.0 to 1.0 and the test result is demonstrated in Table 1. The experimentally defined value for the  $w$  ranges from  $[0.1, 0.3]$  in which the generated path is optimal and computationally inexpensive, which is in line with our algorithm objectives. The generated path is neither the safest possible path nor the shortest possible path, but represents a compromise between the two objectives.



**Figure 8.** Generated waypoints based on user-specified threshold altitude value.

The algorithm traverses various paths from the starting location to the goal. At the same time, it compares the risk cost of all the vertices. Starting with the initial node, it maintains a priority queue of nodes to be traversed, known as the openQueue (see Listing 1). The lower  $f(n)$  for a given node  $n$ , the higher its priority. At each step of the algorithm, the node with the lowest  $f(n)$  value is removed from the queue, the  $f$  and  $h$  values of its neighbors are updated accordingly, and these neighbors with the least radar cost are added to the openQueue. The conditions of adding a node to the openQueue are as follows:

- The node is not in high radar risk region;
- The node is movable and does not exist in the openQueue;
- The distance between the starting point and the current node is shorter than the earlier estimated distance;
- The distance between the starting point and the current node does not violate the pre-specified detection altitude.

If the four conditions are satisfied, then the current node will be added to the openQueue. The algorithm continues until a goal node has a lower  $f$

| Cost Weighting Parameter, $w$ | Explored Nodes | Cost of the Generated Path |
|-------------------------------|----------------|----------------------------|
| $w$ is not used               | 2405           | 103                        |
| 0                             | 385            | 117.5                      |
| 0.1                           | 369            | 112.5                      |
| 0.2                           | 365            | 110.5                      |
| 0.3                           | 365            | 106.5                      |
| 0.4                           | 468            | 104                        |
| 0.5                           | 2405           | 103                        |
| 0.6                           | 4975           | 101                        |
| 0.7                           | 7073           | 100.5                      |
| 0.8                           | 7521           | 99                         |
| 0.9                           | 7766           | 97                         |
| 1.0                           | 8085           | 97                         |

The algorithm traverses various paths from the starting location to the goal. At the same time, it compares the risk cost of all the vertices. Starting with the initial node, it maintains a priority queue of nodes to be traversed, known as the openQueue (see Listing 1). The lower  $f(n)$  for a given node  $n$ , the higher its priority. At each step of the algorithm, the node with the lowest  $f(n)$  value is removed from the queue, the  $f$  and  $h$  values of its neighbors are updated accordingly, and these neighbors with the least radar cost are added to the openQueue. The conditions of adding a node to the openQueue are as follows:

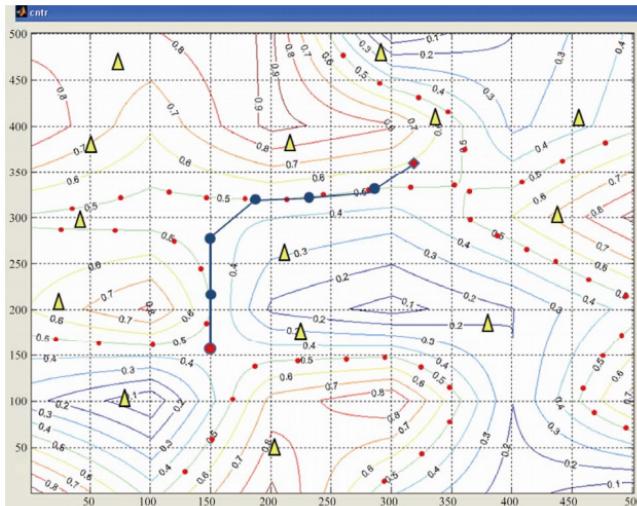
- The node is not in high radar risk region;
- The node is movable and does not exist in the openQueue;
- The distance between the starting point and the current node is shorter than the earlier estimated distance;
- The distance between the starting point and the current node does not violate the pre-specified detection altitude.

If the four conditions are satisfied, then the current node will be added to the openQueue. The algorithm continues until a goal node has a lower  $f$

value than any node in the queue. If the actual optimal path is desired, the algorithm may also update each neighbor with its immediate predecessor in the best path found so far; this information can then be used to reconstruct the path by working backwards from the goal node. The output of this stage is a set of straight line segments connecting a subset of the nodes resulting from the waypoint generation process, as long as the link does not violate the pre-specified detection altitude. The path planning algorithm is implemented in a MATLAB environment and a sample of the result can be seen in Figure 9.

The pseudo-code for the algorithm is as shown in Listing 1.

The next step is to modify the generated path by using a series of cubic splines to optimize the path by moving away from high radar risks and improving the navigability of the path for the UAV.



**Figure 9.** Generated path with least radar risk based on user-specified detection altitude.

## Trajectory Generation

As a first step, the path is parameterized in both x and y directions. An initial spline knot is fixed at the UAV starting location, while the final spline knot is fixed at the first edge of the path. We begin by splitting each edge of the path into three segments, each one described by a different B-spline curve [23]. In this work, we calculate the radar risk cost at several locations along each edge and take the length of the edge into account. The radar risk cost

was calculated at three points along each edge:  $Li/6$ ,  $Li/2$ , and  $5Li/6$ , where  $Li$  is the length of edge  $i$ . The radar risk cost for each edge may be calculated by

$$J_e = L_i \sum_{j=1}^N \sum_{i=1}^M \left( \frac{1}{d_{1/6,i,j}^4} + \frac{1}{d_{1/2,i,j}^4} + \frac{1}{d_{5/6,i,j}^4} \right) \alpha_j \quad (11)$$

where  $N$  is the number of enemy radars;  $M$  is the number of discrete point and  $d_{1/6,i,j}$  is the distance from the  $1/6$ th point on the  $i$ th edge to the  $j$ th enemy radar. The goal now is to find the  $(x, y)$  locations which minimize the exposure to the enemy radar for the middle knots. We apply the graph search algorithm to look for the middle knots with minimal risk cost for each edge of the path. We determine the interpolating curve based on tangent vectors and curvature vectors for each pair of points. The interpolation problem is solved by assuming  $p = 3$ , which produces the  $C^2$  cubic spline [24]. The parameters  $\hat{u}_k$  are used to determine the knot vector  $u$  as

$$\begin{aligned} u_0 &= u_1 = u_2 = \hat{u}_0, & u_{n+4} &= u_{n+5} = u_{n+6} = \hat{u}_n, \\ u_{j+3} &= \hat{u}_j, & j &= 0, \dots, n \end{aligned} \quad (12)$$

Since the number of control points,  $m + 1$ , and the number of the knots,  $n_{\text{knot}} + 1$ , are related by  $n_{\text{knot}} = m + 4$  and, as can be easily deduced from (12),  $n_{\text{knot}} = n + 6$ , the unknown variables  $p_j$  are  $n + 3$ . Once the control points  $p_j, j = 0, \dots, n + 2$  are known, and given the knot vector  $u$ , the B-spline is completely defined. Once this step has been completed, the procedure switches to the algorithm (as shown in Listing 2) to determine how to connect the knots. Note that some experimentally defined knots are added by the algorithm in order to smoothen the spline curve.

```

Function search_path (start, goal)
closedQueue: = the empty queue/*The set of nodes
already evaluated*/
openQueue: = queue containing the initial node/
*The set of tentative nodes to be evaluated*/
g[start]: = 0/*Distance from start along optimal
path*/
h[start]: = estimate_of_distance (start, goal)
f[start]:= h[start]/*Estimated total distance from
start to goal through y*/
while openQueue is not empty
n: = the node in openQueue having the lowest
f[] value
if n = goal
return reconstruct_path (came_from, goal)
remove n from openQueue
add n to closedQueue
for each y in neighbor_nodes (n)
if y in closedQueue
continue
tentative_g: = g [n] + dist_between (n, y) +
risk_cost
/*Compares the risk cost*/
tentative_is_better:= false
if y not in openQueue
add y to openQueue
h[y]: = estimate_of_distance (y, goal)
tentative_is_better:= true
elseif tentative_g < g[y]
tentative_is_better:= true
if tentative_is_better = true
came_from[y]: = n
g[y]: = tentative_g
f[y]: = g[y] + h[y]
return failure
function reconstruct_path (came_from,current_node)
if came_from[current_node] is set p = reconstruct_path (came_from, came_
from[current_node])
return (p + current_node)
else
return the empty path

```

**Listing 1.** Pseudo-code for the path planning algorithm.

```

Function spline_path (start, goal)
locate_points()/*get coordinates (x, y) of middle
knots*/
m_points: = the vector list/* to store control
points*/
m_nodes: = the vector list/* to store small nodes*/
m_steps: = number of steps/*steps for spline*/
add_points (x, y)/*store (x, y) in a vector list*/
while (x, y) is not goal
new_controlpoint_added ()
/*spline curve based on the control points and
nodes*/
for int i = 0 to m_steps
compute blending function

```

$$S_i(t) = [t^3 \quad t^2 \quad t \quad 1] \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix}$$

```

for t ∈ [0, 1]
get_point (double u, const GLVector & P0,
const GLVector & P1, const GLVector & P2, const
GLVector & P3)
/*return coordinate (x, y) for particular t*/
add_node (const GLVector & node)/*add
small nodes in between the control points and we
make use of 100 small nodes in this work*/
end for
end while
render_spline()/*display generated curve*/
return the spline path

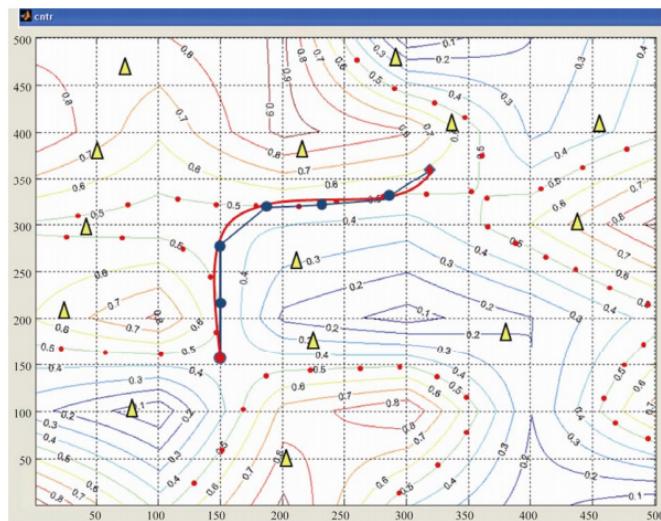
```

**Listing 2.** Pseudo-code for generating a spline path.

It is worth noticing that the interpolation by means of cubic B-splines guarantees the C2 continuity of the geometric path. Owing to the continuity of the curve derivatives, it is not possible to obtain a path with sharp corners that is not flyable by the UAVs. An illustration of the generated solution is shown in Figure 10.

## CONCLUSIONS AND FUTURE WORKS

This paper presents a contour based path planner to generate an optimal path for UAVs over hostile terrain. The proposed method is flexible given that it allows the user to input a threshold altitude value to avoid radar detection and terrain collisions. The key strengths of the method are: 1) the ability to plot a safe path to the target while minimizing exposure to the enemy radar; 2) the generated paths are smoothed using cubic splines to improve navigability and 3) computational efficiency is achieved. Using this planning approach, the ability to generate feasible paths has been demonstrated through simulations. The information provided in this paper is significant for the planning of an air reconnaissance mission. However the procedure has limitations and is sensitive to some of the input parameters, leaving room for improvements in future research based on computational intelligence approaches [25-28].



**Figure 10.** Optimal path based on heuristic search algorithm and user-specified threshold altitude.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the funding support from Temasek Defence Systems Institute, Singapore.

## REFERENCES

1. A. Agarwal, M. H. Lim, M. J. Er and T. N. Nguyen, "Rectilinear Workspace Partitioning for Parallel Coverage Using Multiple UAVs," *Advanced Robotics*, Vol. 21, No. 1, 2007, pp. 105-120.
2. K. K. Lim, Y. S. Ong, M. H. Lim and A. Agarwal, "Hybrid Ant Colony Algorithm for Path Planning in Sparse Graphs," *Soft Computing Journal*, Vol. 12, No. 10, 2008, pp. 981-994.
3. C. W. Yeu, M. H. Lim, G. Huang, A. Agarwal and Y. S. Ong, "A New Machine Learning Paradigm for Terrain Reconstruction," *IEEE Geoscience and Remote Sensing Letters*, Vol. 3, No. 3, 2006, pp. 981-994. doi:10.1109/LGRS.2006.873687
4. L. Davis, "Warp Speed: Path Planning for Star Trek: Armada," *AAAI Spring Symposium*, AAAI Press, Menlo Park, 2000.
5. E. Frazzoli, M. Dahleh and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control and Dynamics*, Vol. 25, No. 1, 2002, pp. 116-129. doi:10.2514/2.4856
6. N. H. Sleumer and N. Tschichold-Gürman, "Exact Cell Decomposition of Arrangements Used for Path Planning in Robotics," *Technical Reports 329*, ETH Zürich, Institute of Theoretical Computer Science, 1999.
7. J. C. Latombe, "Robot Motion Planning," *Kluwer Academic Publishers*, Boston, 1991.
8. T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *Communications of the ACM*, Vo1. 22, No. 10, 1979, pp. 565-570.
9. M. Jun, "Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environments," In: S. Butenko, R. Murphrey and P. Pardalos, Eds., *Cooperative Control: Models, Applications and Algorithms*, Kluwer, 2003, pp. 95-111.
10. J. Hilgert, K. Hirsch, T. Bertram and M. Hiller, "Emergency Path Planning for Autonomous Vehicles Using Elastic Band Theory," *Advanced Intelligent Mechatronics*, Vol. 2, 2003, pp. 1390-1395.
11. T. Sattel and T. Brandt, "Ground Vehicle Guidance Along Collision-Free Trajectories Using Elastic Bands," *Proceedings of 2005 American*

- Control Conference, Portla, 2005, pp. 4991-4999.
- 12. J. Hwang, R. C. Arkin and D. Kwon, "Mobile Robots at Your Fingertip: Bezier Curve On-Line Trajectory Generation for Supervisory Control," Proceedings of Intelligent Robots and Systems (IROS 2003), Las Vegas, Vol. 2, 2003, pp. 1444-1449.
  - 13. J. Aleotti, S. Caselli and G. Maccherozzi, "Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration," Proceedings of Computational Intelligence in Robotics and Automation, Barcelona, 2005, pp. 73-78.
  - 14. K. B. Judd and T. W. McLain, "Spline Based Path Planning for Unmanned Air Vehicles," AIAA Guidance, Navigation, and Control Conference and Exhibit, Montreal, 2001.
  - 15. E. P. Anderson, R. W. Beard, and T. W. McLain, "RealTime Dynamic Trajectory Smoothing for Unmanned Air Vehicles," IEEE Transactions on Control Systems Technology, Vol. 13, No. 3, 2005, pp. 471-477. doi:10.1109/TCST.2004.839555
  - 16. E. M. Kan, M. H. Lim, S. P. Yeo, S. H. Shao and J. S. Ho, "Radar Aware Path Planning for UAVs," IEEE Symposium on Intelligent Systems and Applications, Trabzon, June 2009.
  - 17. E. M. Kan, S. P. Yeo, C. S. Tan, S. H. Shao and J. S. Ho, "Stealth Path Planning for UAVs in Hostile Radar Zones," Proceedings of the 11th IASTED International Conference on Control and Applications, Cambridge, July 2009, pp. 54-60.
  - 18. F. W. Moore, "Radar Cross-Section Reduction via Route Planning and Intelligent Control," IEEE Transactions on Control Systems Technology, Vol. 10, No. 5, 2002, pp. 696-700. doi:10.1109/TCST.2002.801879
  - 19. U. F. Knott, "Radar Cross Section Measurements," Van Nostrand Reinhold, New York, 1993.
  - 20. L. Sevgi, "Complex Electromagnetic Problems and Numerical Simulation Approaches," IEEE Press/John Wiley & Sons, New York, 2003.
  - 21. E. F. Knott, J. F. Shaeffer, and M. T. Tuley, "Radar Cross Section," 2nd Edition, Artech House, Norwood, 1993, p. 231.
  - 22. M. Pachter, D. R. Jacques and J. M. Hebert, "Minimizing Radar Exposure in Air Vehicle Path Planning," Proceedings of the 41st Israel Annual Conference on Aerospace Sciences, Tel-Aviv, February 2001.
  - 23. M. G. Cox, "The Numerical Evaluation of B-Splines," IMA Journal of

Applied Mathematics, Vol. 10, No. 2, 1972, pp. 134-149. doi:10.1093/imamat/10.2.134

24. R. H. Bartels, J. C. Beatty and B. A. Barsky, "An Introduction to Splines for Use in Computer Graphics and Geometric Modeling," Morgan Kaufmann Publishers, Massachusetts, 1987.
25. Q. Cao, M. H. Lim, J. H. Li, Y. S. Ong and W. L. Ng, "A Context Switchable Fuzzy Inference Chip," IEEE Transactions on Fuzzy Systems, Vol. 14, No. 4, 2006, pp. 552- 567. doi:10.1109/TFUZZ.2006.876735
26. M. H. Lim and Y Takefuji, "Implementing Fuzzy RuleBased Systems on Silicon Chips," IEEE Expert, Vol. 5, No. 1, 1990, pp. 31-45. doi:10.1109/64.50855
27. R. Meuth, M. H. Lim, Y. S. Ong and D. C. Wunsh, "A Proposition on Memes and Meta-Memes in Computing for Higher-Order Learning," Memetic Computing, Vol. 1, No. 2, 2009, pp. 85-100. doi:10.1007/s12293-009-0011-1
28. M. H. Lim, Q. Cao, J. H. Li and W. L. Ng, "Evolvable Hardware Using Context Switchable Fuzzy Inference Processor," IEEE Proceedings: Computers and Digital Techniques, Vol. 151, No. 4, 2004, pp. 301-311. doi:10.1049/ip-cdt:20040666



# **CHAPTER**

# **8**

## **Trajectory Tracking of Quadrotor Aerial Robot Using Improved Dynamic Inversion Method**

---

**Lifeng Wang, Yichong He, Zhixiang Zhang, Congkui He**

Field Bus Technology & Automation Lab, North China University of Technology, Beijing, China

### **ABSTRACT**

This paper presents trajectory tracking control works concerning quadrotor aerial robot with rigid cross structure. The quadrotor consists of four propellers which are two paired clockwise rotate and anticlockwise rotate. A nonlinear dynamic model of the quadrotor is provided, and a controller based on the improved dynamic inverse is synthesized for the purpose of stabilization and trajectory tracking. The proposed control strategy has been tested in simulation that can balance the deviation of model inaccuracy well.

---

**Citation:** L. Wang, Y. He, Z. Zhang and C. He, "Trajectory Tracking of Quadrotor Aerial Robot Using Improved Dynamic Inversion Method," *Intelligent Control and Automation*, Vol. 4 No. 4, 2013, pp. 343-348. doi: 10.4236/ica.2013.44040.

**Copyright:** © 2013 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

**Keywords:** Aerial Robot; Quadrotor; Nonlinear Model; Dynamic Inversion; Robust Control

## INTRODUCTION

Since the beginning of 20<sup>th</sup> century, many research efforts have been done to create effective flying machines with improved performance and capabilities. The quadrotor aerial robot is emerging as a kind of popular platform for unmanned aerial vehicle (UAV) research due to its simple structure, hovering ability and vertical take-off and landing (VTOL) capability. It is especially suitable for reconnaissance and surveillance missions in complex and tough environment. Because of quadrotor's complexity of dynamic and uncertainty of parameters, it is usually chosen as a testbed for validating the advanced control algorithm of nonlinear control.

The quadrotor is described as aero robot in [1,2]. The core problems of quadrotor are attitude control and trajectory tracking control. The flight control unit and its control programming code are priority to its structure when treated as a testbed for control algorithm. With the emerging of open code project for autonomous control, it's more convenient to transplant the advanced control algorithms from modeling and simulation to validating test, which also provide great help to research enthusiast or aerial robot hobbyist. In [3], the differential flat control and nonlinear inverse control are proposed to process the nonlinear trajectory control for differential flatness property system. In [4], an real-time nonlinear optimal control, which is called  $\theta-D$  technique, is employed to solve the resultant challenging problem considering the full nonlinear dynamics without gaining scheduling techniques and timescale separations, which leads to a closedform suboptimal control law. In [5,6], the back-stepping procedure is synthesized for the purposes of stabilization and trajectory tracking, which is generally robustness applied to the system of parameters uncertainty and unmodeled dynamic to some extent. The various types of aerial robot UAV based on quadrotor concept are put forward in [5-8], using the superiority of fix-wing plant and rotor-wing helicopter to enhance the VTOL ability and flying range. So, in general, the dynamic inverse method is more suitable for various types of aerial robots due to its start from original model. In [9], the dynamic inverse control with zero dynamic for non-linear affine model system is proposed. In [10], the hierarchical flight control is synthesis for rotorcraft-based UAV. In present paper, the hierarchical structure of improved dynamic inverse method for quadrotor trajectory tracking is

proposed, which can balance the deviation of model inaccuracy to enhance the robustness of control system.

In this paper, we first describe the mechanism of quadrotor aerial robot in section 2. Next, we establish the dynamic model of quadrotor obtained from NewtonEuler formula in section 3. In secion 4, a hierarchical structure of improved dynamic inversion method is proposed. Furthermore, robustness with respect to parameters uncertainty and unmodeled dynamic has been observed in simulation. Finally, a trajectory tracking example is simulated and remarked.

## QUADROTOR MECHANISM

The typical quadrotor mechanism and coordinate system is briefly showed in **Figure 1**, where four rotors are cross structure, providing the three required moments (roll, pitch and yaw) and lifting force. For describing the flight condition of quadrotor exactly, we can respectively define flat-earth frame E and rigid body frame B according to right-hand rule.

The aerodynamic forces and moments of propeller depend on various parameters, such as the angle of attack, aerofoil and inflow ratio. We assumed that the aerodynamic forces  $F_i$  and moments  $T_i$  of four propellers are proportional to the square of angular velocity  $\omega_i$ . Which is given by:

$$\begin{aligned} F_i &= k_f \omega_i^2 \\ T_i &= k_t \omega_i^2 \end{aligned} \quad (1)$$

where  $i = 1, 2, 3, 4$ . The  $k_t$ ,  $k_f$  is respectively aerodynamic force and anti-torque coefficient. Since the forces and moments are derived by modifying the speed of each rotor, then the lifting force and three moments can be given as follows:

Lifting force (the sum of four rotor forces):

$$\begin{aligned} F &= F_1 + F_2 + F_3 + F_4 \\ &= k_f (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{aligned} \quad (2)$$

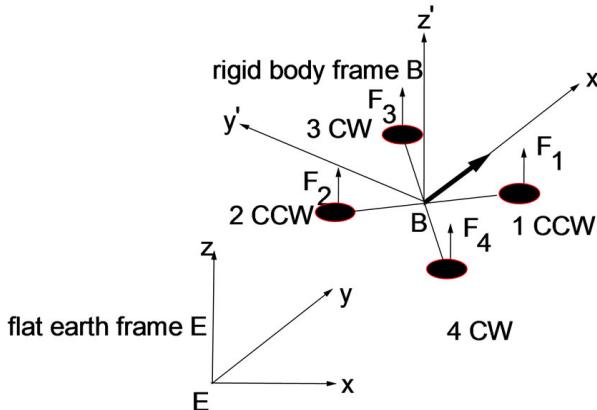
Roll torque (right-hand rule):

$$\begin{aligned} T_p &= L(-F_1 + F_2 + F_3 - F_4) \\ &= Lk_f (-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \end{aligned} \quad (3)$$

Pitch torque (right –hand rule):

$$\begin{aligned} T_q &= L(-F_1 + F_2 - F_3 + F_4) \\ &= Lk_f(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{aligned} \quad (4)$$

Yaw torque (right-hand rule):



**Figure 1.** Quadroter UAV coordinate frame systems.

$$\begin{aligned} T_r &= L(T_1 + T_2 - T_3 - T_4) \\ &= Lk_t(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \end{aligned} \quad (5)$$

where L is the distance from motor position to the center of quadrotor mass. Rewrite the matrix form of Equations (2)-(5) as follows:

$$\begin{bmatrix} T_p \\ T_q \\ T_r \\ F_z \end{bmatrix} = \begin{bmatrix} -Lk_f & Lk_f & Lk_f & -Lk_f \\ -Lk_f & Lk_f & -Lk_f & Lk_f \\ k_t & k_t & -k_t & -k_t \\ k_f & k_f & k_f & k_f \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (6)$$

## DYNAMIC MODEL OF QUADROTOR

The frame E stands for earth coordinate system, and the body frame B stands for the rigid quadrotor coordinate system. The body coordinate B is attached to the center of mass of the quadrotor. The Euler angles which denote the rotation of quadrotor in the flat-earth frame coordinate E are the variables,  $\phi$ ,  $\theta$  and  $\psi$ , which respectively represents the roll, pitch and yaw angles of the quadrotor in the body-fixed coordinates.

## Six DOF Model of Rigid Body

The dynamics of a rigid body under external forces applied to the center of mass expressed in the body fixed frame are in Newton-Euler formalism:

$$\begin{aligned} m\dot{V}_b + \Omega_b \times mV_b &= F_b + F_g + F_e \\ I\dot{\Omega}_b + \Omega_b \times I\Omega_b &= T_b \end{aligned} \quad (7)$$

where the  $V_b = [u \ v \ w]^T \in R^3$  is the velocity in the body frame,  $\Omega_b = [p \ q \ r]^T$  is the angular velocity in the body frame.  $F_b, F_g, F_e \in R^3$  denote the external propeller forces acted on the rotors, the weight forces of mass center, the elastic force of landing gear only on the ground, respectively.  $T_b = [T_p \ T_q \ T_r]^T \in R^3$  are external torques in the body frame.  $I^{3x3} \in R^3$  are the inertia matrix of quadrotor,  $m$  is the mass of quadrotor.

The  $3 \times 3$  symmetrical inertia matrix  $I$  is defined as:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (8)$$

and the cross product operator  $\times$  is defined as :

$$\Omega_b \times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \quad (9)$$

The external propeller forces  $F_b$  acted on the rotors is defined as:

$$F_b = [0 \ 0 \ F_z]^T \quad (10)$$

The weight forces of mass center is as follows:

$$F_g = [mgs\theta \ -mgc\theta s\varphi \ -mgc\theta c\varphi]^T \quad (11)$$

where  $s$  and  $c$  are shorthand form of cosine and sine.

The elastic force of landing gear on the ground is modeling the characteristics of taking off situation for quadrotor, so it disappears while flying or leaving the ground. It is modeling as follows:

$$F_e = \begin{bmatrix} 0 & 0 & k_l(z_l - z)\delta(z_l - z) \end{bmatrix}^T \quad (12)$$

where  $k_l$  and  $z_l$  is elastic coefficient and elastic deformation on ground.

And the function  $\delta(z_l - z)$  denotes:

$$\delta(z_l - z) = \begin{cases} 1 & z_l - z > 0 \\ 0 & z_l - z \leq 0 \end{cases} \quad (13)$$

Then the Equation (7) can be expanded as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = -\begin{bmatrix} -rv + qw \\ ru - pw \\ -qu + pv \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_z/m \end{bmatrix} + \begin{bmatrix} gs\theta \\ -gc\theta s\varphi \\ -gc\theta c\varphi \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k_l(z_l - z)\delta(z_l - z)/m \end{bmatrix} \quad (14)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = -\begin{bmatrix} qr(I_{zz} - I_{yy})/I_{xx} \\ pr(I_{xx} - I_{zz})/I_{yy} \\ pq(I_{yy} - I_{xx})/I_{zz} \end{bmatrix} + \begin{bmatrix} T_p/I_{xx} \\ T_q/I_{yy} \\ T_r/I_{zz} \end{bmatrix} \quad (15)$$

## Navigation Equations

To obtain the translational and rotational motions from the body frame to the flat earth coordinate system, the well-known navigation equations are described as follows:

$$\begin{aligned} \dot{E}_e &= S\Omega_b \\ \dot{N}_e &= RV_b \end{aligned} \quad (16)$$

where the vector  $N_e = [x \ y \ z]^T$  stand for the positions of quadrotor in flat earth coordinate, the vector  $E_e = [\phi \ \theta \ \psi]^T$  stand for the attitude angles of body frame in the flat earth coordinate. The matrix R and S is translational and rotational transform matrix. The Equation (16) can be extended as follows:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (17)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (18)$$

where s, c and t stand for sine, cosine and tangent, respectively.

## Motor Dynamics

The revolving speed of propeller driven by the brushless DC-motor is proportional to the input voltage which is regulated by speed control unit. Input voltage signals come from the flight control system in the light of flight task. Assumed that the load anti-torque is proportional to the square of revolving speed and the inductance of brushless DC-motor is omitted, the dynamic model of DC-motor is described in following formula:

$$\dot{\omega}_i = -k_1 \omega_i - k_2 \omega_i^2 + k_3 \omega_{ic} \quad (19)$$

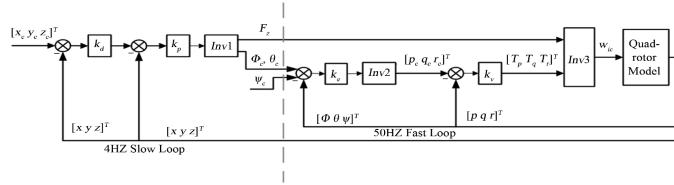
where the coefficient,  $k_1$ ,  $k_2$  and  $k_3$  is constant in dynamic model,  $\omega_{ic}$  is the regulated signal pwm from the flight control unit and the  $i$ th is the index of DC-motor.

## IMPROVED DYNAMIC INVERSE CONTROLLER

The above airframe dynamics of typical quadrotor involves Equations (14), (15), (17)-(19), including 6 equations in Equations (14) and (15) stand for airframe rigid-body dynamics, 6 equations in Equations (17) and (18) stand for navigation transformation, 4 equations in Equations (19) stand for motor dynamics. In these equations describe the dynamics of quadrotor, the total state variables include 16 state variables and 4 control input variables. The inherent characteristics of strong coupling in pitch-roll-yaw and underactuation are shown in state equations. For solving the coupling in pitch-roll-yaw, the dynamic inverse controller is applied to the controller. In parallel, the hierarchical structure of slow-loop and fast-loop is adopted to solve the underactuation. Nevertheless, the accurate model which is necessary in dynamic inverse method is hardly to acquire, so it's important to develop an improved dynamic inverse model to enhance the robustness of controller. The overall controller depicted in **Figure 2** has fast loop as attitude control loop and slow loop as trajectory tracking loop. The details of improving dynamic inverse of hierarchical structure will be described in following process.

## Slow Loop Control

The desired trajectory can be divided into sequence of segments with waypoints. So quadrotor tracks from one waypoint to the next waypoint with desired velocity and acceleration constraints, then the segments are connected



**Figure 2.** Improved dynamic inverse controller of hierarchical structure for quadrotor.

to approximate the continuous trajectory well. The slow loop frequency which depends on the response of translational motion and position sensor is about 4 Hz.

A given trajectory command signal include position command  $(x_c \ y_c \ z_c)$  and orientation command  $\psi_c$ . We assumed that the pitch and roll angle are small enough to combine the formula (14) and (18). Then, symbol inv1 can be denoted by the relationship of attitude angle and position derivative as follows:

$$\begin{aligned} F_z &= (\ddot{z} + g)m \\ \varphi_c &= (\ddot{x}s\psi - \dot{y}c\psi)m/F_z \\ \theta_c &= (\ddot{x}c\psi + \dot{y}s\psi)m/F_z \end{aligned} \quad (20)$$

Trajectory control is considered to exponentially stabilize the waypoint position error. Then the desired acceleration command would satisfy the relationship as follows:

$$\begin{aligned} \ddot{x} &= -2\xi\omega\dot{x} - \omega^2(x - x_c) \\ &= k_u(k_x(x_c - x) - \dot{x}) = -k_u\dot{x} - k_u k_x(x - x_c) \\ \ddot{y} &= -2\xi\omega\dot{y} - \omega^2(y - y_c) \\ &= k_v(k_y(y_c - y) - \dot{y}) = -k_v\dot{y} - k_v k_y(y - y_c) \\ \ddot{z} &= -2\xi\omega\dot{z} - \omega^2(z - z_c) \\ &= k_w(k_z(z_c - z) - \dot{z}) = -k_w\dot{z} - k_w k_z(z - z_c) \end{aligned} \quad (21)$$

As depicted in **Figure 2**,  $\xi$ ,  $\omega$  is the damping factor and natural frequency of system, respectively. The factor  $k_p = [k_x \ k_y \ k_z]^T$  and  $k_d = [k_u \ k_v \ k_w]^T$ . From Equations (20) and (21) above, the control problem of quadrotor is addressed by decoupling the position control and attitude control, and position control of slow loop provides the attitude set points for the attitude controller of fast loop. We have testified that the desired system dynamics could be perfect when damping factor is constant 0.4 - 0.8. Only if the natural frequency of system required to aware, the controller factor can be quantified as follows:

$$\begin{aligned} k_x = k_y = k_z &= \frac{\omega}{2\xi} \\ k_u = k_v = k_w &= 2\omega\xi \end{aligned} \quad (22)$$

## Fast Loop Control

Fast loop is attitude control loop which is required to response the change of attitude angle quickly. Its frequency which settled by system dynamics and control unit is 50 Hz.

Equation (17) can be transformed as follows:

$$\begin{aligned} \begin{bmatrix} p_c \\ q_c \\ r_c \end{bmatrix} &= \begin{bmatrix} 1 & s\varphi t\theta & c\varphi t\theta \\ 0 & c\varphi & -s\varphi \\ 0 & s\varphi/c\theta & c\varphi/c\theta \end{bmatrix}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & s\varphi t\theta & c\varphi t\theta \\ 0 & c\varphi & -s\varphi \\ 0 & s\varphi/c\theta & c\varphi/c\theta \end{bmatrix}^{-1} \begin{bmatrix} k_\varphi(\varphi_c - \varphi) \\ k_\theta(\theta_c - \theta) \\ k_\psi(\psi_c - \psi) \end{bmatrix} \end{aligned} \quad (23)$$

where the matrix inverse stands for symbol inv2. We can acquire  $(p_c \ q_c \ r_c)$  by expected attitude angle command  $(\phi_c \ \theta_c \ \psi_c)$ .

Equation (15) can be transformed as follows:

$$\begin{aligned} \begin{bmatrix} T_p \\ T_q \\ T_r \end{bmatrix} &= \begin{bmatrix} \dot{p}I_{xx} \\ \dot{q}I_{yy} \\ \dot{r}I_{zz} \end{bmatrix} + \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} k_p(p_c - p)I_{xx} \\ k_q(q_c - q)I_{yy} \\ k_r(r_c - r)I_{zz} \end{bmatrix}}_{\text{Omitted}} + \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \end{aligned} \quad (24)$$

In Equation (24), the second item is intentionally omitted in order to maintain zero dynamics while angular velocity error is zero. Thus, the model error or unmodel dynamic characteristics in dynamic inverse method can be balanced through improving the dynamic inverse model. Though the proposed model of dynamic inverse is inaccurate, the zero dynamics can be kept even if the model is not valid. We can acquire  $(T_p \ T_q \ T_r)$  by expected attitude angle command  $(p_c \ q_c \ r_c)$ .

In above formula, the factors  $k_e = [k_\phi \ k_\theta \ k_\psi]^T$  and  $k_v = [k_p I_{xx} \ k_q I_{yy} \ k_r I_{zz}]^T$  can be quantified in the same process. For example:

$$\begin{aligned}\ddot{\theta}_c &= -2\xi\omega\dot{\theta} - \omega^2(\theta - \theta_c) \\ &= k_q(k_\theta(\theta_c - \theta) - \dot{\theta}) \approx -k_q q - k_q k_\theta(\theta - \theta_c)\end{aligned}\quad (25)$$

The factor  $k_\theta = \frac{\omega}{2\xi}$  and  $k_q = 2\omega\xi$  is gained in the same formula.

## Motor Speed Control

Equation (6) can be transform as follows:

$$\begin{bmatrix} \omega_{1c} \\ \omega_{2c} \\ \omega_{3c} \\ \omega_{4c} \end{bmatrix} = \sqrt{\begin{bmatrix} -1/4 & -1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & -1/4 & -1/4 & 1/4 \\ -1/4 & 1/4 & -1/4 & 1/4 \end{bmatrix}} \begin{bmatrix} T_p/(Lk_f) \\ T_q/(Lk_f) \\ T_r/(Lk_t) \\ F_z/k_f \end{bmatrix} \quad (26)$$

where the square root stands for symbol inv3.

## SIMULATION RESULTS

In this section, the demonstrating results are presented in order to observe the performances of proposed control principle. We considered a case of a rectangle shape trajectory-tracking problem. The parameters used for the aircraft model and the controller are given in **Table 1**. These parameters are referred to the quadrotor prototype designed by our lab, just refer **Figure 3**.

**Table 1.** Model parameters in demonstrator of quadrotor.

| parameter       | description                 | value                 |
|-----------------|-----------------------------|-----------------------|
| m               | Mass of quadrotor           | 2.3 kg                |
| L               | Lever arm length            | 0.7 m                 |
| I <sub>xx</sub> | Body inertia of x axis      | 8.04*10^-3 kgm^2      |
| I <sub>yy</sub> | Body inertia of y axis      | 8.46*10^-3 kgm^2      |
| I <sub>zz</sub> | Body inertia of z axis      | 14.68*10^-3 kgm^2     |
| k <sub>f</sub>  | Force coefficient           | 0.65016e-3N/(rad/s)^2 |
| k <sub>t</sub>  | Torque coefficient          | 0.82218e-5N/(rad/s)^2 |
| k <sub>1</sub>  | Motor model coefficient     | 20                    |
| k <sub>2</sub>  | Motor model coefficient     | 0.01                  |
| k <sub>3</sub>  | Motor model coefficient     | 3.5                   |
| z <sub>l</sub>  | Elastic deformation of gear | 0.003 m               |
| k <sub>l</sub>  | Elastic coefficient of gear | 6000 N/m              |

In the improving dynamic inverse controller, the parameters are defined as follows: The natural frequency of slow loop in Equation (21) is guessed as 1 Hz, the damping ratio is chosen as 0.72 according the experience in simulation. So the proportional parameters in controller are defined in Equation (22). In the same way, the natural frequency of pitch and roll attitude in Equation (25) is guessed as 10 Hz and the damping ratio is chosen as 0.7. But the natural frequency of yaw attitude is chosen as 2 Hz and damping ratio is chosen as 0.7.

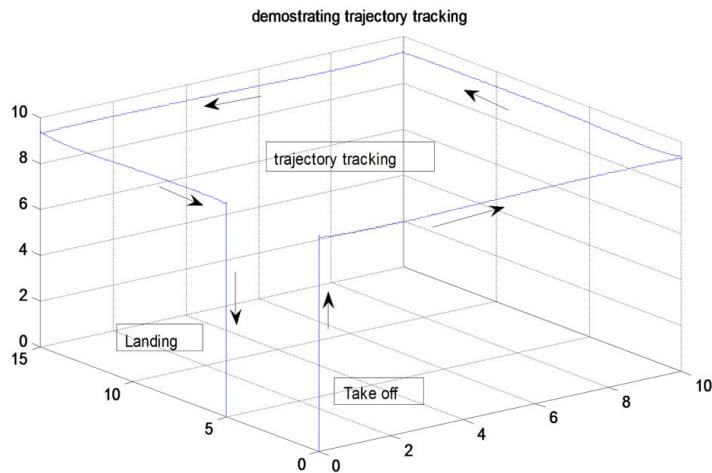
The rectangle shape trajectory is 10 m × 15 m field with an altitude about 10 m altitude, the tracking simulation task is described as taking off ,tracking the rectangle shape trajectory, then landing. The results shown in Figure 4 illustrate the trajectory tracking performance of control strategy developed in this paper. The modeling difference between plant model and dynamic inverse controller include two aspects: no motor dynamic model in controller and 10% deviation of quadrotor body inertia and mass. But we can see the proposed controller behaves perfectly in **Figure 4**, except for the altitude due to quadrotor mass error.

## CONCLUSION

The analysis presented here has shown the hover and



**Figure 3.** The demonstrator of simulating quadrotor.



**Figure 4.** Illustrate the trajectory tracking performance of improved dynamic inverse strategy.

Lifeng Wang, Yichong He, Zhixiang Zhang, Congkui He trajectory tracking capability of proposed controller. A nonlinear dynamic model of the quadrotor is provided, and a controller based on the improved dynamic inverse is synthesized for the purpose of stabilization and trajectory tracking. The proposed control strategy can balance the deviation of model inaccuracy.

## REFERENCES

1. T. Tomic, K. Schmid, P. Lutz, et al., "Towards a Fully Autonomous UAV," IEEE Robotics & Automation Magazine, Vol. 19, No. 3, 2012, pp. 46-56.
2. H. Lim, J. Park, D. Lee and H. J. Kim, "Build Your Own Quadrotor: Open-Source Projects on Unmanned Aerial Vehicles," IEEE Robotics & Automation Magazine, Vol. 19, No. 3, 2012, pp. 33-45.
3. A. Drouin, S. Simoes-Cunha, and A. C. Brandao-Ramos, "Differential Flatness and Control of Nonlinear Systems," Proceedings of the 30th Chinese Control Conference, 22-24 July 2011, Yantai, pp. 643-648.
4. M. Xin, Y. J. Xu and R. Hopkins, "Trajectory Control of Miniature Helicopters Using a Unified Nonlinear Optimal Control Technique," Journal of Dynamic Systems, Measurement and Control, Vol. 133, No. 6, 2011, 14 p.
5. K. T. Oner, E. Cetinsel, M. Unel, et al., "Dynamic Model and Control of a New Quadrotor Unmanned Aerial vehicle with Tilt-Wing Mechanism," World Academy of Science, Engineering and Technology, 2008.
6. Z. Fang and W. N. Gao, "Adaptive Integral Backstepping Control of a Micro-Quadrotor," The 2nd International Conference on Intelligent Control and Information Processing, Harbin, 25-28 July 2011, pp. 910-915.
7. I. F. Kendoul and R. Lozano, "Modeling and Control of a Small Autonomous Aircraft Having Two Tilting Rotors," IEEE Transactions on Robotics, Vol. 22, No. 6, 2006, pp. 1297-1302.
8. K. T. Oner, E. Cetinsel, E. Sirimoglu, et al., "Mathematical Modeling and Vertical Flight Control of a Tilt-Wing UAV," Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 20, No. 1, 2012, p. 149.
9. A. Das, K. Subbarao and F. Lewis, "Dynamic Inversion with Zero-Dynamics Stabilisation for Quadrotor Control," IET Control Theory Applications, Vol. 3, No. 3, 2009, pp. 303-314. <http://dx.doi.org/10.1049/iet-cta:20080002>
10. H. Shim, "Hierarchical Flight Control System Synthesis for Rotorcraft-Based Unmanned Aerial Vehicles," Doctor of Philosophy Dissertation, University Of California, Berkeley, 2000.



**SECTION 3**

**QUADROTORs AND**

**SMALL-SIZE AERIAL**

**VEHICLES**



# **CHAPTER**

# **9**

## **Neural Network Control and Wireless Sensor Network-based Localization of Quadrotor UAV Formations**

---

**Travis Dierks and S. Jagannathan**

Missouri University of Science and Technology, United States of America

### **INTRODUCTION**

In recent years, quadrotor helicopters have become a popular unmanned aerial vehicle (UAV) platform, and their control has been undertaken by many researchers (Dierks & Jagannathan, 2008). However, a team of UAV's working together is often more effective than a single UAV in scenarios like surveillance, search and rescue, and perimeter security. Therefore, the formation control of UAV's has been proposed in the literature.

---

**Citation:** Travis Dierks and S. Jagannathan (January 1st 2009). “Neural Network Control and Wireless Sensor Network-based Localization of Quadrotor UAV Formations”, Aerial Vehicles, Thanh Mung Lam, IntechOpen, DOI: 10.5772/6477.

**Copyright:** © 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License

Saffarian and Fahimi present a modified leader-follower framework and propose a model predictive nonlinear control algorithm to achieve the formation (Saffarian & Fahimi, 2008). Although the approach is verified via numerical simulations, proof of convergence and stability is not provided. In the work of Fierro et al., cylindrical coordinates and contributions from wheeled mobile robot formation control (Desai et al., 1998) are considered in the development of a leader-follower based formation control scheme for aircrafts whereas the complete dynamics are assumed to be known (Fierro et al., 2001). The work by Gu et al. proposes a solution to the leader-follower formation control problem involving a linear inner loop and nonlinear outer-loop control structure, and experimental results are provided (Gu et al., 2006). The associated drawbacks are the need for a dynamic model and the measured position and velocity of the leader has to be communicated to its followers. Xie et al. present two nonlinear robust formation controllers for UAV's where the UAV's are assumed to be flying at a constant altitude. The first approach assumes that the velocities and accelerations of the leader UAV are known while the second approach relaxes this assumption (Xie et al., 2005). In both the designs, the dynamics of the UAV's are assumed to be available. Then, Galzi and Shtessel propose a robust formation controller based on higher order sliding mode controllers in the presence of bounded disturbances (Galzi & Shtessel, 2006).

In this work, we propose a new leader-follower formation control framework for quadrotor UAV's based on spherical coordinates where the desired position of a follower UAV is specified using a desired separation,  $s_d$ , and a desired- angle of incidence,  $\alpha_d$  and bearing,  $\beta_d$ . Then, a new control law for leader-follower formation control is derived using neural networks (NN) to learn the complete dynamics of the UAV online, including unmodeled dynamics like aerodynamic friction in the presence of bounded disturbances. Although a quadrotor UAV is underactuated, a novel NN virtual control input scheme for leader follower formation control is proposed which allows all six degrees of freedom of the UAV to be controlled using only four control inputs. Finally, we extend a graph theory-based scheme for discovery, localization and cooperative control. Discovery allows the UAV's to form into an ad hoc mobile sensor network whereas localization allows each UAV to estimate its position and orientation relative to its neighbors and hence the formation shape. This chapter is organized as follows. First, in Section 2, the leader-follower formation control problem for UAV's is introduced, and required background information is presented. Then, the NN control law is developed for the follower UAV's as well as the formation leader, and the

stability of the overall formation is presented in Section 3. In Section 4, the localization and routing scheme is introduced for UAV formation control while Section 5 presents numerical simulations, and Section 6 provides some concluding remarks.

## BACKGROUND

### Quadrotor UAV Dynamics

Consider a quadrotor UAV with six DOF defined in the inertial coordinate frame ,  $E^a$  , as  $[x, y, z, \phi, \theta, \psi]^T \in E^a$  where  $\rho = [x, y, z]^T \in E^a$  are the position coordinates of the UAV and  $\Theta = [\phi, \theta, \psi]^T \in E^a$  describe its orientation referred to as roll, pitch, and yaw, respectively. The translational and angular velocities are expressed in the body fixed frame attached to the center of mass of the UAV,  $E^b$  , and the dynamics of the UAV in the body fixed frame can be written as (Dierks & Jagannathan, 2008)

$$M \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \bar{S}(\omega) \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} N_1(v) \\ N_2(\omega) \end{bmatrix} + \begin{bmatrix} G(R) \\ 0_{3 \times 1} \end{bmatrix} + U + \tau_d \quad (1)$$

$$\text{where } U = \begin{bmatrix} 0 & 0 & u_1 & u_2^T \end{bmatrix}^T \in \mathfrak{R}^6,$$

$$M = \begin{bmatrix} mI_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & J \end{bmatrix} \in \mathfrak{R}^{6 \times 6}, \quad \bar{S}(\omega) = \begin{bmatrix} -mS(\omega) & 0_{3 \times 3} \\ 0_{3 \times 3} & S(J\omega) \end{bmatrix} \in \mathfrak{R}^{6 \times 6}$$

and  $m$  is a positive scalar that represents the total mass of the UAV,  $J \in \mathbb{R}^{3 \times 3}$  represents the positive definite inertia matrix,  $v(t) = [v_{xb}, v_{yb}, v_{zb}]^T \in \mathfrak{R}^3$  represents the translational velocity,  $\omega(t) = [\omega_{xb}, \omega_{yb}, \omega_{zb}]^T \in \mathfrak{R}^3$  represents the angular velocity,  $N_i(\bullet) \in \mathfrak{R}^{3 \times 1}, i = 1, 2$ , are the nonlinear aerodynamic effects,  $u_1 \in \mathfrak{R}^1$  provides the thrust along the z-direction,  $u_2 \in \mathfrak{R}^3$  provides the rotational torques,  $\tau_d = [\tau_{d1}^T, \tau_{d2}^T]^T \in \mathfrak{R}^6$  and ,  $\tau_{di} \in \mathfrak{R}^3, i = 1, 2$  represents unknown, but bounded disturbances such that  $\|\tau_d\| < \tau_M$  for all time  $t$  , with  $\tau_M$  being a known positive constant,  $I_{nxn} \in \mathfrak{R}^{nxn}$  is an nxn identity matrix, and  $0_{mxl} \in \mathfrak{R}^{mxl}$  represents an mxl matrix of all zeros. Furthermore,  $G(R) \in \mathfrak{R}^3$  represents the gravity vector defined as  $G(R) = mgR^T(\Theta)E_z$  where  $E_z = [0, 0, 1]^T$  is a unit vector in the inertial coordinate frame,  $g = 9.81 \text{ m/s}^2$  , and  $S(\bullet) \in \mathfrak{R}^{3 \times 3}$  is the general form of a skew symmetric matrix defined as in (Dierks & Jagannathan, 2008). It is important to highlight  $w^T S(\gamma) w = 0$  for any vector  $w \in \mathfrak{R}^3$  , and this property is commonly referred to as the skew symmetric property (Lewis et al., 1999).

The matrix  $R(\Theta) \in \Re^{3x3}$  is the translational rotation matrix which is used to relate a vector in the body fixed frame to the inertial coordinate frame defined as (Dierks & Jagannathan, 2008)

$$R(\Theta) = R = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \quad (2)$$

where the abbreviations  $s_{(\bullet)}$  and  $c_{(\bullet)}$  have been used for  $\sin(\bullet)$  and  $\cos(\bullet)$ , respectively. It is important to note that  $R^{-1} = R^T$ ,  $\dot{R} = RS(\omega)$  and  $\dot{R}^T = -S(\omega)R^T$ . It is also necessary to define a rotational transformation matrix from the fixed body to the inertial coordinate frame as (Dierks & Jagannathan, 2008)

$$T(\Theta) = T = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \quad (3)$$

where the abbreviation  $t_{(\bullet)}$  has been used for  $\tan(\bullet)$ . The transformation matrices  $R$  and  $T$  are nonsingular as long as  $-(\pi/2) < \phi < (\pi/2)$ ,  $-(\pi/2) < \theta < (\pi/2)$  and  $-\pi \leq \psi \leq \pi$ . These regions will be assumed throughout the development of this work, and will be referred to as the stable operation regions of the UAV. Under these flight conditions, it is observed that  $\|R\|_F = R_{\max}$  and  $\|T\|_F < T_{\max}$  for known constants  $R_{\max}$  and  $T_{\max}$  (Neff et al., 2007). Finally, the kinematics of the UAV can be written as

$$\begin{aligned} \dot{\rho} &= Rv \\ \dot{\Theta} &= T\omega \end{aligned} \quad (4)$$

## Neural Networks

In this work, two-layer NN's are considered consisting of one layer of randomly assigned constant weights  $V_N \in \Re^{axL}$  in the first layer and one layer of tunable weights  $W_N \in \Re^{Lxb}$  in the second with  $a$  inputs,  $b$  outputs, and  $L$  hidden neurons. A compromise is made here between tuning the number of layered weights with computational complexity. The universal approximation property for NN's (Lewis et al., 1999) states that for any smooth function  $f_N(x_N)$ , there exists a NN such that  $f_N(x_N) = W_N^T \sigma(V_N^T x_N) + \epsilon_N$  where  $\epsilon_N$  is the bounded NN functional approximation error such that  $\|\epsilon_N\| < \epsilon_M$ , for

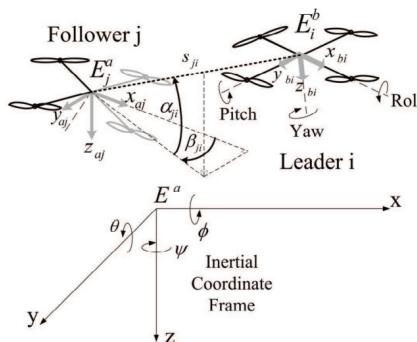
a known constant  $\varepsilon_M$  and a  $\sigma(\cdot) : \Re^a \rightarrow \Re^L$  is the activation function in the hidden layers. It has been shown that by randomly selecting the input layer weights  $V_N$ , the activation function  $\sigma(\bar{x}_N) = \sigma(V_N^T x_N)$  forms a stochastic basis, and thus the approximation property holds for all inputs,  $a \in \mathbb{R}^n$ , in the compact set  $S$ . The sigmoid activation function is considered here. Furthermore, on any compact subset of  $\mathbb{R}^n$ , the target NN weights are bounded by a known positive value,  $W_M$ , such that  $\|W_N\|_F \leq W_M$ . For complete details of the NN and its properties, see (Lewis et al., 1999).

### Three Dimensional Leader-Follower Formation Control

Throughout the development, the follower UAV's will be denoted with a subscript 'j' while the formation leader will be denoted by the subscript 'i'. To begin the development, an alternate reference frame is defined by rotating the inertial coordinate frame about the z-axis by the yaw angle of follower  $j$ ,  $\psi_j$ , and denoted by a  $E_j^a$ . In order to relate a vector in a  $E$  to  $E_j^a$ , the transformation matrix is given by

$$R_{aj} = \begin{bmatrix} \cos \psi_j & \sin \psi_j & 0 \\ -\sin \psi_j & \cos \psi_j & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

where  $R_{aj}^T = R_{aj}^{-1}$ .



**Figure 1.** UAV leader-follower formation control

The objective of the proposed leader-follower formation control approach is for the follower UAV to maintain a desired separation,  $s_{jid}$ , at a desired angle of incidence,  $\alpha_{jid} \in E_j^a$ , and bearing,  $\beta_{jid} \in E_j^a$ , with respect to its leader. The incidence angle is measured from the  $x_{aj}-y_{aj}$  plane of follower  $j$  while the bearing angle is measured from the positive  $x_{aj}$ -axis as shown

in Figure 1. It is important to observe that each quantity is defined relative to the follower  $j$  instead of the leader  $i$  (Fierro et al., 2001), (Desai et al., 1998). Additionally, in order to specify a unique configuration of follower  $j$  with respect to its leader, the desired yaw of follower  $j$  is selected to be the yaw angle of leader  $i$ ,  $\psi_i \in E^a$  as in (Saffarian & Fahimi, 2008). Using this approach, the measured separation between follower  $j$  and leader  $i$  is written as

$$\rho_i - \rho_j = R_{aj}^T S_{ji} \Xi_{ji}, \quad (6)$$

where

$$\Xi_{ji} = \begin{bmatrix} \cos \alpha_{ji} \cos \beta_{ji} \\ \cos \alpha_{ji} \sin \beta_{ji} \\ \sin \alpha_{ji} \end{bmatrix}. \quad (7)$$

Thus, to solve the leader-follower formation control problem in the proposed framework, a control velocity must be derived to ensure

$$\left. \begin{aligned} \lim_{t \rightarrow \infty} (s_{jid} - s_{ji}) &= 0, & \lim_{t \rightarrow \infty} (\beta_{jid} - \beta_{ji}) &= 0, \\ \lim_{t \rightarrow \infty} (\alpha_{jid} - \alpha_{ji}) &= 0, & \lim_{t \rightarrow \infty} (\psi_{jd} - \psi_j) &= 0 \end{aligned} \right\}. \quad (8)$$

Throughout the development,  $s_{jid}$ ,  $\alpha_{jid}$  and  $\beta_{jid}$  will be taken as constants, while the constant total mass,  $m_j$ , is assumed to be known. Additionally, it will be assumed that reliable communication between the leader and its followers is available, and the leader communicates its measured orientation,  $\Theta_i$ , and its desired states,  $\psi_{id}$ ,  $\dot{\psi}_{id}$ ,  $\ddot{\psi}_{id}$ ,  $v_{id}$ ,  $\dot{v}_{id}$ . This is a far less stringent assumption than assuming the leader communicates all of its measured states to its followers (Gu et al., 2006). Additionally, future work will relax this assumption. In the following section, contributions from single UAV control will be considered and extended to the leader-follower formation control of UAV's.

## LEADER-FOLLOWER FORMATION TRACKING CONTROL

In single UAV control literature, the overall control objective UAV  $j$  is often to track a desired trajectory,  $\rho_{jd} = [x_{jd}, y_{jd}, z_{jd}]^T$ , and a desired yaw  $\psi_{jd}$  while maintaining a stable flight configuration (Dierks & Jagannathan,

2008). The velocity  $v_{jzb}$  is directly controllable with the thrust input. However, in order to control the translational velocities  $v_{jxb}$  and  $v_{jyb}$ , the pitch and roll must be controlled, respectively, thus redirecting the thrust. With these objectives in mind, the frameworks for single UAV control are extended to UAV formation control as follows.

## Follower UAV Control Law

Given a leader  $i$  subject to the dynamics and kinematics (1) and (4), respectively, define a reference trajectory at a desired separation  $s_{jid}$ , at a desired angle of incidence,  $\alpha_{jid}$ , and bearing,  $\beta_{jid}$  for follower  $j$  given by

$$\rho_{jd} = \rho_i - R_{ajd}^T s_{jid} \Xi_{jid} \quad (9)$$

where  $R_{ajd}$  is defined as in (5) and written in terms of  $\psi_{jd}$ , and  $\Xi_{jid}$  is written in terms of the desired angle of incidence and bearing,  $\alpha_{jid}$   $\beta_{jid}$ , respectively, similarly to (7). Next, using (6) and (9), define the position tracking error as

$$e_{jp} = \rho_{jd} - \rho_j = R_{aj}^T s_{ji} \Xi_{ji} - R_{ajd}^T s_{jid} \Xi_{jid} \in E^a \quad (10)$$

which can be measured using local sensor information. To form the position tracking error dynamics, it is convenient to rewrite (10) as  $e_{jp} = \rho_i - \rho_j - R_{ajd}^T s_{jid} \Xi_{jid}$  revealing

$$\dot{e}_{jp} = R_i v_i - R_j v_j - \dot{R}_{ajd}^T s_{jid} \Xi_{jid}. \quad (11)$$

Next, select the desired translational velocity of follower  $j$  to stabilize (11)

$$v_{jd} = [v_{jdx} \ v_{jdy} \ v_{jdz}]^T = R_j^T (R_i v_{id} - \dot{R}_{ajd}^T s_{jid} \Xi_{jid} + K_{jp} e_{jp}) \in E^b \quad (12)$$

where  $K_{jp} = \text{diag}\{k_{jp\alpha}, k_{jp\gamma}, k_{jp\zeta}\} \in \mathbb{R}^{3 \times 3}$  is a diagonal positive definite design matrix of positive design constants and  $v_{id}$  is the desired translational velocity of leader  $i$ . Next, the translational velocity tracking error system is defined as

$$e_{jv} = \begin{bmatrix} e_{jvx} \\ e_{jvy} \\ e_{jvz} \end{bmatrix} = \begin{bmatrix} v_{jdx} \\ v_{jdy} \\ v_{jdz} \end{bmatrix} - \begin{bmatrix} v_{jxb} \\ v_{jyb} \\ v_{jzb} \end{bmatrix} = v_{jd} - v_j. \quad (13)$$

Applying (12) to (11) while observing  $v_j = v_{jd} - e_{jv}$  and similarly  $e_{iv} = v_{id} - v_i$ , reveals the closed loop position error dynamics to be rewritten as

$$\dot{e}_{jp} = -K_{jp}e_{jp} + R_j e_{jv} - R_i e_{iv}. \quad (14)$$

Next, the translational velocity tracking error dynamics are developed. Differentiating (13), observing

$$\dot{v}_{jd} = -S(\omega_j)v_{jd} + R_j^T(R_i S(\omega_i)v_{id} + R_i \dot{v}_{id} - \ddot{R}_{ajd}^T S_{jid} \Xi_{jid}) + R_j^T K_{jp}(R_i v_i - R_j v_j - \dot{R}_{ajd}^T S_{jid} \Xi_{jid}),$$

substituting the translational velocity dynamics in (1), and adding and subtracting  $R_j^T(K_{jp}(R_i v_{id} + R_j v_{jd}))$  reveals

$$\begin{aligned} \dot{e}_{jv} = & \dot{v}_{jd} - \dot{v}_j = -N_{jl}(v_j)/m_j - S(\omega_j)e_{jv} - G(R_j)/m_j - u_{jl}E_{jz}/m_j - \bar{\tau}_{jd1} \\ & + R_j^T(R_i S(\omega_i)v_{id} + R_i \dot{v}_{id} - \ddot{R}_{ajd}^T S_{jid} \Xi_{jid} + K_{jp}(R_j e_{jv} - K_{jp}e_{jp})) - R_j^T K_{jp}(R_i e_{iv} - R_j e_{jv}) \end{aligned} \quad (15)$$

Next, we rewrite (2) in terms of the scaled desired orientation vector,  $\bar{\Theta}_{jd} = [\bar{\theta}_{jd} \bar{\phi}_{jd} \psi_{jd}]^T$  where  $\bar{\theta}_{jd} = \pi\theta_{jd}/(2\theta_{d\max})$ ,  $\bar{\phi}_{jd} = \pi\phi_{jd}/(2\phi_{d\max})$ , and  $\theta_{d\max} \in (0, \pi/2)$  and  $\phi_{d\max} \in (0, \pi/2)$  are the maximum desired roll and pitch, respectively, define  $R_{jd} = R_j(\bar{\Theta}_{jd})$ , and add and subtract  $G(R_{jd})/m_j$  and  $R_{jd}^T \Lambda_j$  with  $\Lambda_j = R_j \dot{v}_{id} - \ddot{R}_{ajd}^T S_{jid} \Xi_{jid} + K_{jp}R_j e_{jv} - K_{jp}e_{jp}$  to  $\dot{e}_{jv}$  yield

$$\dot{e}_{jv} = -G(R_{jd})/m_j + R_{jd}^T \Lambda_j + A_{jc1} f_{jc1}(x_{jc1}) - u_{jl}E_{jz}/m_j - K_{jp}R_i e_{iv} - \bar{\tau}_{jd1} \quad (16)$$

where  $A_{jc1} = \text{diag}\{\cos(\bar{\theta}_{jd}), \cos(\bar{\phi}_{jd}), 1\} \in \mathfrak{R}^{3 \times 3}$  and

$$\begin{aligned} f_{jc1}(x_{jc1}) = & A_{jc1}^{-1} \left( G(R_{jd})/m_j - G(R_j)/m_j + (R_j^T - R_{jd}^T) \Lambda_j \right) + \\ & A_{jc1}^{-1} \left( K_{jp}R_j e_{jv} - S(\omega_j)e_{jv} - N_{jl}(v_j)/m_j + R_j^T R_i S(\omega_i)v_{id} + R_j^T K_{jp}(1 - K_{jp})e_{jp} \right) \end{aligned} \quad (17)$$

is an unknown function which can be rewritten as  $f_{jc1}(x_{jc1}) = [f_{jc11} \ f_{jc12} \ f_{jc13}]^T \in \mathfrak{R}^3$ . In the forthcoming development, the approximation properties of NN will be utilized to estimate the unknown function  $f_{jc1}(x_{jc1})$  by bounded ideal weights  $W_{jc1}^T, V_{jc1}^T$ , such that  $\|W_{jc1}\|_F \leq W_{Mc1}$  for an unknown constant  $W_{Mc1}$ , and written as  $f_{jc1}(x_{jc1}) = W_{jc1}^T \sigma(V_{jc1}^T \hat{x}_{jc1}) + \epsilon_{jc1}$  where  $\epsilon_{jc1} \leq \epsilon_{Mc1}$  is the bounded NN approximation error where  $\epsilon_{Mc1}$  is a known constant. The NN estimate of  $f_{jc1}$  is written as  $\hat{f}_{jc1} = \hat{W}_{jc1}^T \sigma(V_{jc1}^T \hat{x}_{jc1}) = \hat{W}_{jc1}^T \hat{\sigma}_{jc1}$  where  $\hat{W}_{jc1}^T = [\hat{W}_{jc11}^T \hat{\sigma}_{jc1} \hat{W}_{jc12}^T \hat{\sigma}_{jc1} \hat{W}_{jc13}^T \hat{\sigma}_{jc1}]^T$  and  $\hat{W}_{jc1i}^T, i = 1, 2, 3$  is the NN estimate of  $T W_{jc1}, i = 1, 2, 3$ ,  $\hat{x}_{jc1}$  is the NN input defined as

$$\hat{x}_{jc1} = [1 \ \Theta_j^T \ \Theta_i^T \ \Lambda_j^T \ v_{jd}^T \ v_{id}^T \ \dot{v}_{id}^T \ \psi_{jd} \ \dot{\psi}_{jd} \ \ddot{\psi}_{jd} \ \omega_j^T \ v_j^T \ e_{jv}^T \ e_{jp}^T]^T.$$

Note that  $\hat{x}_{jcl}$  is an estimate of  $x_{jcl}$  since the follower does not know  $\omega_i$ . However,  $\Theta_i$  is directly related to  $\omega_i$ ; therefore, it is included instead.

Remark 1: In the development of (16), the scaled desired orientation vector was utilized as a design tool to specify the desired pitch and roll angles. If the un-scaled desired orientation vector was used instead, the maximum desired pitch and roll would remain within the stable operating regions. However, it is desirable to saturate the desired pitch and roll before they reach the boundaries of the stable operating region.

Next, the virtual control inputs  $\theta_{jd}$  and  $\phi_{jd}$  are identified to control the translational velocities  $v_{jxb}$  and  $v_{jyb}$ , respectively. The key step in the development is identifying the desired closed loop velocity tracking error dynamics. For convenience, the desired translational velocity closed loop system is selected as

$$\dot{e}_{jv} = -K_{jv} e_{jv} - \bar{\tau}_{jd1} - K_{j\rho} R_i e_{iv} \quad (18)$$

where  $K_{jv} = diag\{k_{jv1} \cos(\bar{\theta}_{jd}), k_{jv2} \cos(\bar{\phi}_{jd}), k_{v3}\}$  is a diagonal positive definite design matrix with each  $k_{vi} > 0$ ,  $i=1,2,3$ , and  $\bar{\tau}_{jd1} = \tau_{jd1}/m_j$ . In the following development, it will be shown that  $\theta_d \in (-\pi/2, \pi/2)$  and  $\phi_d \in (-\pi/2, \pi/2)$ ; therefore, it is clear that  $K_v > 0$ . Then, equating (16) and (18) while considering only the first two velocity error states reveals

$$-g \begin{bmatrix} -s_{\bar{\theta}_{jd}} \\ c_{\bar{\theta}_{jd}} s_{\bar{\theta}_{jd}} \end{bmatrix} + \begin{bmatrix} c_{\bar{\theta}_{jd}} (k_{jv1} e_{jvx} + f_{jc11}) \\ c_{\bar{\theta}_{jd}} (k_{jv2} e_{jvy} + f_{jc12}) \end{bmatrix} + \begin{bmatrix} c_{\bar{\theta}_{jd}} c_{\psi d} & c_{\bar{\theta}_{jd}} s_{\psi d} & -s_{\bar{\theta}_{jd}} \\ s_{\bar{\theta}_{jd}} c_{\bar{\theta}_{jd}} c_{\psi d} - c_{\bar{\theta}_{jd}} s_{\psi d} & s_{\bar{\theta}_{jd}} c_{\bar{\theta}_{jd}} s_{\psi d} + c_{\bar{\theta}_{jd}} c_{\psi d} & s_{\bar{\theta}_{jd}} c_{\bar{\theta}_{jd}} \\ c_{\bar{\theta}_{jd}} & \Lambda_{j1} \\ \Lambda_{j2} & 0 \\ \Lambda_{j3} & 0 \end{bmatrix} = \begin{bmatrix} \Lambda_{j1} \\ \Lambda_{j2} \\ \Lambda_{j3} \end{bmatrix} \quad (19)$$

where  $\Lambda_j = [\Lambda_{j1} \ \Lambda_{j2} \ \Lambda_{j3}]^T$  was utilized. Then, applying basic math operations, the first line of (19) can be solved for the desired pitch  $\theta_{jd}$  while the second line reveals the desired roll  $\phi_{jd}$ . Using the NN estimates,  $f_{cj1}$ , The desired pitch  $\theta_{jd}$  can be written as

$$\theta_{jd} = \frac{2\theta_{\max}}{\pi} a \tan\left(\frac{N_{\theta d}}{D_{\theta d}}\right) \quad (20)$$

where  $N_{\theta d} = c_{\psi d} \Lambda_{j1} + s_{\psi d} \Lambda_{j2} + k_{jv1} e_{jvx} + \hat{f}_{jc11}$  and  $D_{\theta d} = \Lambda_{j3} - g$ . Similarly, the desired roll angle,  $\phi_{jd}$ , is found to be

$$\phi_{jd} = \frac{2\phi_{\max}}{\pi} a \tan\left(\frac{N_{\phi d}}{D_{\phi d}}\right) \quad (21)$$

where  $N_{\phi d} = s_{\psi d} \Lambda_{j1} - c_{\psi d} \Lambda_{j2} - k_{jv2} e_{jvy} + \hat{f}_{jc12}$  and  
 $D_{\phi d} = c_{\bar{\theta}d} (\Lambda_{j3} - g) + s_{\bar{\theta}d} c_{\psi d} \Lambda_{j1} + s_{\bar{\theta}d} s_{\psi d} \Lambda_{j2}$ .

**Remark 2:** The expressions for the desired pitch and roll in (20) and (21) lend themselves very well to the control of a quadrotor UAV. The expressions will always produce desired values in the stable operation regions of the UAV. It is observed that a  $\tan(\bullet)$  approaches  $\pm \pi / 2$  as its argument increases. Thus, introducing the scaling factors in  $\bar{\theta}_{jd}$  and  $\bar{\phi}_{jd}$  results in  $\theta_{jd} \in (-\theta_{\max}, \theta_{\max})$  and  $\phi_{jd} \in (-\phi_{\max}, \phi_{\max})$ , and the aggressiveness of the UAV's maneuvers can be managed.

Now that the desired orientation has been found, next define the attitude tracking error as

$$e_{j\Theta} = \Theta_{jd} - \Theta_j \in E^a \quad (22)$$

where the dynamics are found using (4) to be  $\dot{e}_{j\Theta} = \dot{\Theta}_{jd} - T_j \omega_j$ . In order to drive the orientation errors (22) to zero, the desired angular velocity,  $\omega_{jd}$ , is selected as

$$\omega_{jd} = T_j^{-1} (\dot{\Theta}_{jd} + K_{j\Theta} e_{j\Theta}) \quad (23)$$

where  $K_{j\Theta} = \text{diag}\{k_{j\Theta 1}, k_{j\Theta 2}, k_{j\Theta 3}\} \in \Re^{3 \times 3}$  is a diagonal positive definite design matrix all with positive design constants. Define the angular velocity tracking error as

$$e_{j\omega} = \omega_{jd} - \omega_j \quad (24)$$

and observing  $\omega_j = \omega_{jd} - e_{j\omega}$ , the closed loop orientation tracking error system can be written as

$$\dot{e}_{j\Theta} = -K_{j\Theta} e_{j\Theta} + T_j e_{j\omega} \quad (25)$$

Examining (23), calculation of the desired angular velocity requires knowledge of  $\dot{\Theta}_{jd}$ ; however,  $\dot{\Theta}_{jd}$  is not known in view of the fact  $\dot{\Lambda}_j$  and  $\dot{\hat{f}}_{jc1}$  are not available. Further, development of  $u_{,}$  in the following section will reveal  $\dot{\omega}_{jd}$  is required which in turn implies  $\ddot{\Lambda}_j$  and  $\ddot{\hat{f}}_{jc1}$  must be known. Since these requirements are not practical, the universal approximation property of NN is invoked to estimate  $\omega_{jd}$  and  $\dot{\omega}_{jd}$  (Dierks and Jagannathan, 2008).

To aid in the NN virtual control development, the desired orientation,  $\Theta_{jd} \in E^a$ , is reconsidered in the fixed body frame,  $E^b$ , using the relation  $\dot{\Theta}_{jd}^b = T_j^{-1} \dot{\Theta}_{jd}$ . Rearranging (23), the dynamics of the proposed virtual controller when all dynamics are known are revealed to be

$$\begin{aligned}\dot{\Theta}_{jd}^b &= \omega_{jd} - T_j^{-1} K_{j\Theta} e_{j\Theta} \\ \dot{\omega}_{jd} &= \dot{T}_j^{-1} (\dot{\Theta}_{jd} + K_{j\Theta} e_{j\Theta}) + T_j^{-1} (\ddot{\Theta}_{jd} + K_{j\Theta} \dot{e}_{j\Theta})\end{aligned}. \quad (26)$$

For convenience, we define a change of variable as  $\Omega_d = \omega_d - T^{-1} K_\Theta e_\Theta$ , and the dynamics (26) become

$$\begin{aligned}\dot{\Theta}_{jd}^b &= \Omega_{jd} \\ \dot{\Omega}_{jd} &= \dot{T}_j^{-1} \dot{\Theta}_{jd} + T_j^{-1} \ddot{\Theta}_{jd} = f_{j\Omega}(x_{j\Omega}) = f_{j\Omega}\end{aligned}. \quad (28)$$

Defining the estimates of  $\Theta_{jd}^b$  and  $\Omega_{jd}$  to be  $\hat{\Theta}_{jd}^b$  and  $\hat{\Omega}_{jd}$ , respectively, and the estimation error  $\tilde{\Theta}_{jd}^b = \Theta_{jd}^b - \hat{\Theta}_{jd}^b$ , the dynamics of the proposed NN virtual control inputs become

$$\begin{aligned}\dot{\hat{\Theta}}_{jd}^b &= \hat{\Omega}_{jd} + K_{j\Omega 1} \tilde{\Theta}_{jd}^b \\ \dot{\hat{\Omega}}_{jd} &= \hat{f}_{j\Omega} + K_{j\Omega 2} \tilde{\Theta}_{jd}^b\end{aligned}. \quad (28)$$

where  $K_{j\Omega 1}$  and  $K_{j\Omega 2}$  are positive constants. The estimate  $\hat{\omega}_{jd}$  is then written as

$$\hat{\omega}_{jd} = \hat{\Omega}_{jd} + K_{j\Omega 3} \tilde{\Theta}_{jd}^b + T_j^{-1} K_{j\Theta} e_{j\Theta} \quad (29)$$

where  $K_{j\Omega 3}$  is a positive constant.

In (28), universal approximation property of NN has been utilized to estimate the unknown function  $f_{j\Omega}(x_{j\Omega})$  by bounded ideal weights  $W_{j\Omega}^T, V_{j\Omega}^T$ , such that  $\|W_{j\Omega}\|_F \leq W_{M\Omega}$  for a known constant  $W_{M\Omega}$ , and written as  $f_{j\Omega}(x_{j\Omega}) = W_{j\Omega}^T \sigma(V_{j\Omega}^T x_{j\Omega}) + \varepsilon_{j\Omega}$  where  $\varepsilon_{j\Omega}$  is the bounded NN approximation error such that  $\|\varepsilon_{j\Omega}\| \leq \varepsilon_{\Omega M}$  for a known constant  $\varepsilon_{\Omega M}$ . The NN estimate of  $f_{j\Omega}$  is written as  $\hat{f}_{j\Omega} = \hat{W}_{j\Omega}^T \sigma(\hat{V}_{j\Omega}^T \hat{x}_{j\Omega}) = \hat{W}_{j\Omega}^T \hat{\sigma}_{j\Omega}$  where  $\hat{W}_{j\Omega}^T$  is the NN estimate of  $W_{j\Omega}^T$  and  $\hat{x}_{j\Omega}$  is the NN input written in terms of the virtual control estimates, desired trajectory, and the UAV velocity. The NN input is chosen to take the form of  $\hat{x}_{j\Omega} = [1 \ \Lambda_j^T \ (\Theta_{jd}^b)^T \ \hat{\Omega}_{jd}^T \ v_j^T \ \omega_j^T]^T$ .

Observing  $\tilde{\omega}_{jd} = \omega_{jd} - \hat{\omega}_{jd} = \tilde{\Omega}_{jd} - K_{j\Omega^3} \tilde{\Theta}_{jd}^b$ , subtracting (28) from (27) and adding and subtracting  $W_{j\Omega}^T \hat{\sigma}_{j\Omega}$ , the virtual controller estimation error dynamics are found to be

$$\begin{aligned}\dot{\tilde{\Theta}}_{jd}^b &= \tilde{\omega}_{jd} - (K_{j\Omega^1} - K_{j\Omega^3}) \tilde{\Theta}_{jd}^b \\ \dot{\tilde{\Omega}}_{jd} &= \tilde{f}_{j\Omega} - K_{j\Omega^2} \tilde{\Theta}_{jd}^b + \xi_{j\Omega}\end{aligned}\quad (30)$$

where  $\tilde{\Omega}_{jd} = \Omega_{jd} - \hat{\Omega}_{jd}$ ,  $\tilde{f}_{j\Omega} = \tilde{W}_{j\Omega}^T \hat{\sigma}_{j\Omega}$ ,  $\tilde{W}_{j\Omega}^T = W_{j\Omega}^T - \hat{W}_{j\Omega}^T$ ,  $\xi_{j\Omega} = \varepsilon_{j\Omega} + W_{j\Omega}^T \tilde{\sigma}_{j\Omega}$ , and  $\tilde{\sigma}_{j\Omega} = \sigma_{j\Omega} - \hat{\sigma}_{j\Omega}$ . Furthermore,  $\|\xi_{j\Omega}\| \leq \xi_M$  with  $\xi_M = \varepsilon_M + 2W_{M\Omega} \sqrt{N_\Omega}$  a computable constant with  $N\Omega$  the constant number of hidden layer neurons in the virtual control NN. Similarly, the estimation error dynamics of (29) are found to be

$$\dot{\tilde{\omega}}_{jd} = -K_{j\Omega^3} \tilde{\omega}_{jd} + \tilde{f}_{j\Omega} - \bar{K}_{j\Omega} \tilde{\Theta}_{jd}^b + \xi_{j\Omega} \quad (31)$$

where  $\bar{K}_{j\Omega} = K_{j\Omega^2} - K_{j\Omega^3}(K_{j\Omega^1} - K_{j\Omega^3})$ . Examination of (30) and (31) reveals  $\tilde{\Theta}_{jd}^b$ ,  $\tilde{\omega}_{jd}$ , and  $\tilde{f}_{j\Omega}$  to be equilibrium points of the estimation error dynamics when  $\|\xi_{j\Omega}\| = 0$ .

To this point, the desired translational velocity for follower j has been identified to ensure the leader-follower objective (8) is achieved. Then, the desired pitch and roll were derived to drive  $v_{jxb} \rightarrow v_{jdx}$  and  $v_{jyb} \rightarrow v_{jdy}$ , respectively. Then, the desired angular velocity was found to ensure  $\Theta_j \rightarrow \Theta_{jd}$ . What remains is to identify the UAV thrust to guarantee  $v_{jzb} \rightarrow v_{jdz}$  and rotational torque vector to ensure  $\omega_j \rightarrow \omega_{jd}$ . First, the thrust is derived.

Consider again the translational velocity tracking error dynamics (16), as well as the desired velocity tracking error dynamics (18). Equating (16) and (18) and manipulating the third error state, the required thrust is found to be

$$\begin{aligned}u_{j1} &= m_j c_{\bar{\theta}jd} c_{\bar{\phi}jd} (\Lambda_{j3} - g) + m_j (c_{\bar{\phi}jd} s_{\bar{\theta}jd} c_{\psi jd} + s_{\bar{\phi}jd} s_{\psi jd}) \Lambda_{j1} \\ &+ m_j (c_{\bar{\phi}jd} s_{\bar{\theta}jd} s_{\psi jd} - s_{\bar{\phi}jd} c_{\psi jd}) \Lambda_{j2} + m_j k_{jvz} e_{vj3} + m_j \hat{f}_{jc13}\end{aligned}\quad (32)$$

where  $\hat{f}_{jc13}$  is the NN estimate in (17) previously defined. Substituting the desired pitch (20), roll (21), and the thrust (32) into the translational velocity tracking error dynamics (16) yields

$$\dot{e}_{jv} = -K_{jv} e_{jv} + A_{jcl} (W_{jcl}^T \sigma_{jcl} + \epsilon_{jc}) - A_{jcl} \hat{W}_{jcl}^T \hat{\sigma}_{jcl} - K_{jp} R_i e_{iv} - \bar{\tau}_{jd},$$

and adding and subtracting  $A_{jcl} W_{jcl}^T \hat{\sigma}_{jcl}^T$  reveals

$$\dot{e}_{jv} = -K_{jv} e_{jv} + A_{jcl} \tilde{W}_{jcl}^T \hat{\sigma}_{jcl} - K_{jp} R_i e_{iv} + \xi_{jcl} \quad (33)$$

with

$$\xi_{jcl} = A_{jcl} W_{jcl}^T \tilde{\sigma}_{jcl}^T + A_{jcl} \epsilon_{jc} - \bar{\tau}_{jd}$$

,  $\tilde{W}_{jcl} = W_{jcl} - \hat{W}_{jcl}$ , and  $\tilde{\sigma}_{jcl} = \sigma_{jcl} - \hat{\sigma}_{jd}$  for a known constant

,  $\|A_{jcl}\|_F = A_{c1\max}$ , and  $\|\xi_{jcl}\| \leq \xi_{Mcl}$  for a computable constant

$$\xi_{Mcl} = A_{c1\max} \epsilon_{Mcl} + 2A_{c1\max} W_{Mcl} \sqrt{N_c} + \tau_M / m_j.$$

Next, the rotational torque vector,  $u_{j2}$ , will be addressed. First, multiply the angular velocity tracking error (24) by the inertial matrix  $J_j$ , take the first derivative with respect to time, and substitute the UAV dynamics (1) to reveal

$$J_j \dot{e}_{j\omega} = f_{jc2}(x_{jc2}) - u_{j2} - \tau_{jd} \quad (34)$$

with  $f_{jc2}(x_{jc2}) = J_j \dot{\omega}_{jd} - S(J_j \omega_j) \omega_j - N_{j2}(\omega_j)$ . Examining

$f_{jc2}(x_{jc2})$ , it is clear that the function is nonlinear and contains unknown terms; therefore, the universal approximation property of NN is utilized to estimate the function  $f_{jc2}(x_{jc2})$  by bounded ideal weights  $W_{jc2}^T, V_{jc2}^T$

, such that  $\|W_{jc2}\|_F \leq W_{Mc2}$  for a known constant  $W_{Mc2}$  and written as

$f_{jc2}(x_{jc2}) = W_{jc2}^T \sigma(V_{jc2}^T x_{jc2}) + \epsilon_{jc2}$  where  $\epsilon_{jc2}$  is the bounded NN functional

reconstruction error such that  $\|\epsilon_{jc2}\| \leq \epsilon_{Mc2}$  for a known constant  $\epsilon_{Mc2}$ . The

NN estimate of  $f_{jc2}$  is given by  $\hat{f}_{jc2} = \hat{W}_{j2}^T \sigma(V_{jc2}^T \hat{x}_{jc2}) = \hat{W}_{j2}^T \hat{\sigma}_{j2}$  where  $\hat{W}_{j2}^T$  is

the NN estimate of  $W_{jc2}^T$  and  $\hat{x}_{jc2} = [1 \ \omega_j^T \ \dot{\Omega}_{jd}^T \ \tilde{\Theta}_{jd}^{b,T} \ e_{j\theta}^T]^T$  is the input to the NN

written in terms of the virtual controller estimates. By the construction of the virtual controller,  $\dot{\hat{\omega}}_{j\theta}$  is not directly available; therefore, observing (29),

the terms  $\dot{\hat{\Omega}}_{jd}^T$ ,  $\tilde{\Theta}_{jd}^{b,T}$ , and  $e_{j\theta}^T$  have been included instead.

Using the NN estimate  $\hat{f}_{jc2}$  and the estimated desired angular velocity tracking error  $\hat{e}_{j\omega} = \hat{\omega}_{jd} - \omega_j$ , the rotational torque control input is written as

$$u_{j2} = \hat{f}_{j\omega 2} + K_{j\omega} \hat{e}_{j\omega}, \quad (35)$$

and substituting the control input (35) into the angular velocity dynamics (34) as well as adding and subtracting  $\tilde{W}_{jc2}^T \hat{\sigma}_{jc}$ , the closed loop dynamics become

$$J_j \dot{e}_{j\omega} = -K_{j\omega} e_{j\omega} + \tilde{W}_{jc2}^T \hat{\sigma}_{jc2} + K_{j\omega} \tilde{\omega}_{jd} + \xi_{jc2} \quad (36)$$

$$\text{where } \tilde{W}_{jc2}^T = W_{jc2}^T - \hat{W}_{jc2}^T, \quad \xi_{jc2} = \varepsilon_{jc2} + W_{jc2}^T \tilde{\sigma}_{jc} - \tau_{jd2}, \quad \text{and}$$

$\tilde{\sigma}_{jc2} = \sigma_{jc2} - \hat{\sigma}_{jc2}$ . Further,  $\|\xi_{jc2}\| \leq \xi_{Mc2}$  for a computable constant  $\xi_{Mc2} = \varepsilon_{Mc2} + 2W_{Mc2} \sqrt{N_{c2}} + \tau_{dM}$  where  $N_{c2}$  is the number of hidden layer neurons.

As a final step, we define  $\tilde{W}_{jc} = [\tilde{W}_{jc1} \ 0; 0 \ \tilde{W}_{jc2}]$  and  $\hat{\sigma}_{jc} = [\hat{\sigma}_{jc1}^T \ \hat{\sigma}_{jc2}^T]^T$  so that a single NN can be utilized with  $N_c$  hidden layer neurons to represent  $\hat{f}_{jc} = [\hat{f}_{jc1}^T \ \hat{f}_{jc2}^T]^T \in \mathcal{R}^6$ . In the following theorem, the stability of the follower  $j$  is shown while considering  $e_{iv} = 0$ . In other words, the position, orientation, and velocity tracking errors are considered along with the estimation errors of the virtual controller and the NN weight estimation errors of each NN for follower  $j$  while ignoring the interconnection errors between the leader and its followers. This assumption will be relaxed in the following section.

*Theorem 3.1.1:* (Follower UAV System Stability) Given the dynamic system of follower  $j$  in the form of (1), let the desired translational velocity for follower  $j$  to track be defined by (12) with the desired pitch and roll defined by (20) and (21), respectively. Let the NN virtual controller be defined by (28) and (29), respectively, with the NN update law given by

$$\dot{\hat{W}}_{j\Omega} = F_{j\Omega} \hat{\sigma}_{j\Omega} \left( \tilde{\Theta}_{jd}^b \right)^T - \kappa_{j\Omega} F_{j\Omega} \hat{W}_{j\Omega}, \quad (37)$$

where  $F_{j\Omega} = F_{j\Omega}^T > 0$  and  $\kappa_{j\Omega} > 0$  are design parameters. Let the dynamic NN controller for follower  $j$  be defined by (32) and (35), respectively, with the NN update given by

$$\dot{\hat{W}}_{jc} = F_{jc} \hat{\sigma}_{jc} (A_{jc} \hat{e}_{js})^T - \kappa_{jc} F_{jc} \hat{W}_{jc}, \quad (38)$$

where  $A_{jc} = [A_{jel} \ 0_{3x3} \ 0_{3x3} \ I_{3x3}] \in \Re^{6x6}$ ,  $\hat{e}_{js} = [e_{jv}^T \ \hat{e}_{j\omega}^T]^T$ ,  $F_{jc} = F_{jc}^T > 0$  are constant design parameters. Then there exists positive design constants  $K_{j\Omega 1}, K_{j\Omega 2}, K_{j\Omega 3}$  and positive definite design matrices  $K_{j\rho}, K_{j\Theta}, K_{jv}, K_{j\omega}$ , such that the virtual controller estimation errors  $\tilde{\Theta}_{jd}^b, \tilde{\omega}_{jd}$  and the virtual control NN weight estimation errors,  $\tilde{W}_{j\Omega}$ , the position, orientation, and translational and angular velocity tracking errors,  $e_{j\rho}, e_{j\Theta}, e_{jv}, e_{j\omega}$ , respectively, and the dynamic controller NN weight estimation errors,  $\tilde{W}_{jc}$ , are all SGUUB. Proof: Consider the following positive definite Lyapunov candidate

$$V_j = V_{j\Omega} + V_{jc}, \quad (39)$$

where

$$V_{j\Omega} = \frac{1}{2} \tilde{\Theta}_{jd}^{b\ T} \bar{K}_{j\Omega} \tilde{\Theta}_{jd}^b + \frac{1}{2} \tilde{\omega}_{jd}^{T} \tilde{\omega}_{jd} + \frac{1}{2} \text{tr}\{\tilde{W}_{j\Omega}^T F_{j\Omega}^{-1} \tilde{W}_{j\Omega}\}$$

$$V_{jc} = \frac{1}{2} e_{j\rho}^T e_{j\rho} + \frac{1}{2} e_{j\Theta}^T e_{j\Theta} + \frac{1}{2} e_{jv}^T e_{jv} + \frac{1}{2} e_{j\omega}^T J_j e_{j\omega} + \frac{1}{2} \text{tr}\{\tilde{W}_{jc}^T F_{jc}^{-1} \tilde{W}_{jc}\}$$

whose first derivative with respect to time is given by  $\dot{V}_j = \dot{V}_{j\Omega} + \dot{V}_{jc}$

. Considering first  $\dot{V}_{j\Omega}$ , and substituting the closed loop virtual control estimation error dynamics (30) and (31) as well as the NN tuning law (37), reveals

$$\dot{V}_{j\Omega} = -\bar{K}_{j\Omega 2} \tilde{\Theta}_{jd}^{b\ '} \tilde{\Theta}_{jd}^b - K_{j\Omega 3} \tilde{\omega}_{jd}^{T} \tilde{\omega}_{jd} + \tilde{\omega}_{jd}^{T} \xi_{j\Omega} + \text{tr}\{\tilde{W}_{j\Omega}^T (K_{j\Omega} \tilde{W}_{j\Omega} - \hat{\sigma}_{j\Omega} [\tilde{\Theta}_{jd}^b]') + \hat{\sigma}_{j\Omega} \tilde{\omega}_{jd}^T\}$$

$$\text{where } \bar{K}_{j\Omega 2} = (K_{j\Omega 1} - K_{j\Omega 3})(K_{j\Omega 2} - K_{j\Omega 3}(K_{j\Omega 1} - K_{j\Omega 3})) \text{ and } \bar{K}_{j\Omega 2} > 0$$

provided  $K_{j\Omega 1} > K_{j\Omega 3}$  and  $K_{j\Omega 2} > K_{j\Omega 3}(K_{j\Omega 1} - K_{j\Omega 3})$ . Observing  $\|\hat{\sigma}_{j\Omega}\| \leq \sqrt{N_{j\Omega}}$ ,  $\|W_{\Omega j}\|_F \leq W_{M\Omega}$  for a known constant,  $W_{M\Omega}$ , and

$$\text{tr}\{\tilde{W}_{j\Omega}^T (W_{j\Omega} - \tilde{W}_{j\Omega})\} \leq \|\tilde{W}_{j\Omega}\|_F W_{M\Omega} - \|\tilde{W}_{j\Omega}\|_F^2, \quad \dot{V}_{j\Omega} \text{ can then be rewritten}$$

$$\text{as } \dot{V}_{j\Omega} \leq -\bar{K}_{j\Omega} \|\tilde{\Theta}_{jd}^b\|^2 - K_{j\Omega} \|\tilde{\omega}_{jd}\|^2 - \kappa_{j\Omega} \|\tilde{W}_{j\Omega}\|_F^2 + \|\tilde{\omega}_{jd}\| \|\xi_{\Omega M}\| + \|\tilde{\Theta}_{jd}^b\| \|\tilde{W}_{j\Omega}\|_F \sqrt{N_{j\Omega}} + \|\tilde{\omega}_{jd}\| \|\tilde{W}_{j\Omega}\|_F$$

$$\|_F \sqrt{N_{j\Omega}} + \kappa_{j\Omega} \|\tilde{W}_{j\Omega}\|_F W_{M\Omega}.$$

Now, completing the squares with respect to  $\|\tilde{W}_{j\Omega}\|_F$ ,  $\|\tilde{\Theta}_{jd}^b\|$ , and  $\|\tilde{\omega}_{jd}\|$ , an upper bound for  $\dot{V}_{j\Omega}$  is found to be

$$\dot{V}_{j\Omega} \leq -\left(\bar{K}_{j\Omega 2} - \frac{N_{j\Omega}}{\kappa_{j\Omega}}\right) \|\tilde{\Theta}_{jd}^b\|^2 - \left(\frac{K_{j\Omega 3}}{2} - \frac{N_{j\Omega}}{\kappa_{j\Omega}}\right) \|\tilde{\omega}_{jd}\|^2 - \frac{\kappa_{j\Omega}}{4} \|\tilde{W}_{j\Omega}\|_F^2 + \eta_{j\Omega} \quad (40)$$

where  $\eta_{j\Omega} = \kappa_{j\Omega} W_{M\Omega}^2 + \xi_{\Omega M}^2 / (2K_{j\Omega 3})$ . Next, considering  $\dot{V}_{jc}$  and substituting the closed loop kinematics (14) and (25), dynamics (33) and (36), and NN tuning law (38) while considering  $e_{iv} = 0$  reveals

$$\begin{aligned} \dot{V}_{jc} &= -e_{jp}^T K_{jp} e_{jp} - e_{j\Theta}^T K_{j\Theta} e_{j\Theta} - e_{jv}^T K_{jv} e_{jv} - e_{j\omega}^T K_{j\omega} e_{j\omega} + e_{jp}^T R_j e_{jv} + e_{j\Theta}^T T_j e_{j\omega} + e_{jv}^T \xi_{jcl} + e_{j\omega}^T K_{j\omega} \tilde{\omega}_{jd} + e_{j\omega}^T \xi_{jcl} \\ &+ \kappa_{jc} \operatorname{tr} [\tilde{W}_{jc}^T (W_{jc} - \tilde{W}_{jc})] + \operatorname{tr} [\tilde{W}_{jc}^T \hat{\sigma}_{jc2} (e_{j\omega}^T - \hat{e}_{j\omega}^T)] \end{aligned}$$

Then, observing  $\tilde{\omega}_{jd} = e_{j\omega} - \hat{e}_{j\omega}$  and completing the squares with respect to  $e_{jp}, e_{j\Theta}, e_{jv}, e_{j\omega}$  and  $\tilde{W}_{jc}$  and  $\tilde{W}_{jc} \sim$ , and upper bound for  $\dot{V}_{jc}$  is found to be

$$\begin{aligned} \dot{V}_{jc} &\leq -\frac{K_{j\rho \min}}{2} \|e_{jp}\|^2 - \frac{K_{j\Theta \min}}{2} \|e_{j\Theta}\|^2 - \left(\frac{K_{jv \min}}{2} - \frac{R_{\max}^2}{2K_{j\rho \min}}\right) \|e_{jv}\|^2 - \frac{\kappa_{jc}}{3} \|\tilde{W}_{jc}\|_F^2 - \left(\frac{K_{j\omega \min}}{3} - \frac{T_{\max}^2}{2K_{j\Theta \min}}\right) \|e_{j\omega}\|^2 \\ &+ \frac{3K_{j\omega \min}}{4} \|\tilde{\omega}_{jd}\|^2 + \frac{3N_{jc}}{4\kappa_{jc}} \|\tilde{\omega}_{jd}\|^2 + \eta_{jc} \end{aligned} \quad (41)$$

where  $K_{j\rho \min}, K_{j\Theta \min}, K_{jv \min}$ , and  $K_{j\omega \min}$ , and  $\kappa_{j\omega \min}$  are the minimum singular values of  $K_{jp}, K_{j\Theta}, K_{jv}$ , and  $K_{j\omega}$ , respectively, and  $\eta_{jc} = \xi_{c1M}^2 / (2K_{jv \min}) + \xi_{c2M}^2 / (2K_{j\omega \min}) + 3W_{Mc} \kappa_{jc} / 4$ . Now, combining (40) and (41), an upper bound for  $\dot{V}_j$  is written as

$$\begin{aligned} \dot{V}_j &\leq -\left(\bar{K}_{j\Omega} - \frac{N_{j\Omega}}{\kappa_{j\Omega}}\right) \|\tilde{\Theta}_{jd}^b\|^2 - \left(\frac{K_{j\Omega 3}}{2} - \frac{N_{j\Omega}}{\kappa_{j\Omega}} - \frac{3K_{j\omega \min}}{4} - \frac{3N_{jc}}{4\kappa_{jc}}\right) \|\tilde{\omega}_{jd}\|^2 - \frac{K_{j\rho \min}}{2} \|e_{jp}\|^2 - \frac{K_{j\Theta \min}}{2} \|e_{j\Theta}\|^2 \\ &- \left(\frac{K_{jv \min}}{2} - \frac{R_{\max}^2}{2K_{j\rho \min}}\right) \|e_{jv}\|^2 - \left(\frac{K_{j\omega \min}}{3} - \frac{T_{\max}^2}{2K_{j\Theta \min}}\right) \|e_{j\omega}\|^2 - \frac{\kappa_{jc}}{4} \|\tilde{W}_{jc}\|_F^2 - \frac{\kappa_{jc}}{3} \|\tilde{W}_{jc}\|_F^2 + \eta_{j\Omega} + \eta_{jc} \end{aligned} \quad (42)$$

Finally, (42) is less than zero provided

$$\bar{K}_{j\Omega 2} > \frac{N_{j\Omega}}{\kappa_{j\Omega}}, \quad K_{j\Omega 3} > \frac{2N_{j\Omega}}{\kappa_{j\Omega}} + \frac{3K_{j\omega \min}}{2} + \frac{3N_{jc}}{2\kappa_{jc}}, \quad K_{jv \min} > \frac{R_{\max}^2}{K_{j\rho \min}}, \quad K_{j\omega \min} > \frac{3T_{\max}^2}{2K_{j\Theta \min}} \quad (43)$$

and the following inequalities hold:

$$\begin{aligned}
\|\tilde{\Theta}_{jd}^b\| &> \sqrt{\frac{\eta_{j\Omega} + \eta_{jc}}{K_{j\Omega\Omega} - N_{j\Omega}/\kappa_{j\Omega}}} \quad \text{or} \quad \|\tilde{W}_{jc}\|_F > \sqrt{\frac{3(\eta_{j\Omega} + \eta_{jc})}{\kappa_{jc}}} \quad \text{or} \quad \|e_{j\rho}\| > \sqrt{\frac{2(\eta_{j\Omega} + \eta_{jc})}{K_{j\rho\min} - R_{\max}^2/K_{\rho\min}}} \\
\text{or} \quad \|e_{j\theta}\| &> \sqrt{\frac{2(\eta_{j\Omega} + \eta_{jc})}{K_{j\theta\min}}} \quad \text{or} \quad \|e_{jv}\| > \sqrt{\frac{2(\eta_{j\Omega} + \eta_{jc})}{K_{jv\min}}} \quad \text{or} \quad \|\tilde{W}_{j\Omega}\|_F > \sqrt{\frac{4(\eta_{j\Omega} + \eta_{jc})}{\kappa_{j\Omega}}} \\
\text{or} \quad \|e_{j\omega}\| &> \sqrt{\frac{\eta_{j\Omega} + \eta_{jc}}{K_{j\omega\min}/3 - T_{\max}^2/(2K_{j\theta\min})}} \quad \text{or} \quad \|\tilde{\omega}_{jd}\| > \sqrt{\frac{\eta_{j\Omega} + \eta_{jc}}{\frac{K_{j\Omega\Omega}}{2} - \frac{N_{j\Omega}}{\kappa_{j\Omega}} - \frac{3K_{j\theta\min}}{4} - \frac{3N_{jc}}{4\kappa_{jc}}}}
\end{aligned} \tag{44}$$

Therefore, it can be concluded using standard extensions of Lyapunov theory (Lewis et al., 1999) that  $\dot{V}_j$  is less than zero outside of a compact set, revealing the virtual controller estimation errors,  $\tilde{\Theta}_{jd}^b$ ,  $\tilde{\omega}_{jd}$ , and the NN weight estimation errors,  $\tilde{W}_{j\Omega}$ , the position, orientation, and translational and angular velocity tracking errors,  $e_{j\rho}, e_{j\theta}, e_{jv}, e_{j\omega}$ , respectively, and the dynamic controller NN weight estimation errors,  $\tilde{W}_{jc}$ , are all SGUUB.

## Formation Leader Control Law

The dynamics and kinematics for the formation leader are defined similarly to (1) and (4), respectively. In our previous work (Dierks and Jagannathan, 2008), an output feedback control law for a single quadrotor UAV was designed to ensure the robot tracks a desired path,  $\rho_{id} = [x_{id}, y_{id}, z_{id}]^T$ , and desired yaw angle,  $\psi_{id}$ . Using a similarly approach to (10)- (14), the state feedback control velocity for leader i is given by (Dierks and Jagannathan, 2008)

$$v_{id} = [v_{idx} \ v_{idy} \ v_{idz}]^T = R_i^T (\dot{\rho}_{id} + K_{i\rho} e_{i\rho}) \in E^b \tag{45}$$

The closed loop position tracking error then takes the form of

$$\dot{e}_{i\rho} = -K_{i\rho} e_{i\rho} + R_i e_{iv} \tag{46}$$

Then, using steps similar to (15)-(21), the desired pitch and roll angles are given by

$$\theta_{id} = \frac{2\theta_{\max}}{\pi} a \tan\left(\frac{N_{i\theta d}}{D_{i\theta d}}\right) \tag{47}$$

where  $N_{i\theta d} = c_{iyd}(\ddot{x}_{id} + k_{ip} \dot{x}_{id} - v_{iRl}) + s_{iyd}(\ddot{y}_{id} + k_{ip} \dot{y}_{id} - v_{iRl}) + k_{pv} e_{vx} + \hat{f}_{ic1}$  and  $D_{i\theta d} = \ddot{z}_{id} + k_{iz} \dot{z}_{id} - v_{iRl} - g$  and

$$\phi_{id} = \frac{2\phi_{\max}}{\pi} a \tan\left(\frac{N_{i\phi d}}{D_{i\phi d}}\right) \quad (48)$$

where  $N_{i\phi d} = s_{i\phi d} (\ddot{x}_{id} + k_{ipx} \dot{x}_{id} - v_{iR1}) - c_{i\phi d} (\ddot{y}_{id} + k_{ipy} \dot{y}_{id} - v_{iR2}) - k_{iv2} e_{yy} + \hat{f}_{ic12}$ ,  $D_{\phi d} = c_{\bar{\theta}id} (\ddot{z}_{id} + k_{ipz} \dot{z}_{id} - v_{iR3} - g) + s_{\bar{\theta}id} c_{\psi id} (\ddot{x}_{id} + k_{ipx} \dot{x}_{id} - v_{iR1}) + s_{\bar{\theta}id} s_{\psi id} (\ddot{y}_{id} + k_{ipy} \dot{y}_{id} - v_{iR2})$  and  $v_{iR} = [v_{iR1} \ v_{iR2} \ v_{iR3}]^T = K_{ip} R_i v_i$  and  $\hat{f}_{ic1} = [\hat{f}_{ic11} \ \hat{f}_{ic12} \ \hat{f}_{ic13}]^T \in \Re^3$  is a NN estimate of the unknown function  $f_{ic1}(x_{ic1})$  (Dierks and Jagannathan, 2008). The desired angular velocity as well as the NN virtual controller for the formation leader is defined similarly to (23) and (28) and (29), respectively, and finally, the thrust and rotation torque vector are found to be

$$u_{il} = m_i c_{\bar{\theta}id} c_{\bar{\phi}id} (\ddot{z}_{id} + k_{ipz} \dot{z}_{id} - v_{iR3} - g) + m_i (c_{\bar{\phi}id} s_{\bar{\theta}id} s_{\bar{\psi}id} - s_{\bar{\phi}id} c_{\bar{\psi}id}) (\ddot{y}_{id} + k_{ipy} \dot{y}_{id} - v_{iR2}) + m_i k_{iv3} e_{ivz} \quad (49)$$

$$+ m_i (c_{\bar{\phi}id} s_{\bar{\theta}id} c_{\bar{\psi}id} + s_{\bar{\phi}id} s_{\bar{\psi}id}) (\ddot{x}_{id} + k_{ipx} \dot{x}_{id} - v_{iR1}) + m_i \hat{f}_{ic13}$$

$$u_{i2} = \hat{f}_{ic2} + K_{i\omega} \hat{e}_{i\omega} \quad (50)$$

where  $\hat{f}_{ic2} \in \Re^3$  is a NN estimate of the unknown function  $f_{ic2}(x_{ic2})$  and  $\hat{e}_{i\omega} = \hat{\omega}_{id} - \omega_i$ . The closed loop orientation, virtual control, and velocity tracking error dynamics for the formation leader are found to take a form similar to (25), (30) and (31), and (33) and (36), respectively (Dierks and Jagannathan, 2008).

Next, the stability of the entire formation is considered in the following theorem while considering the interconnection errors between the leader and its followers.

## Quadrotor UAV Formation Stability

Before proceeding, it is convenient to define the following augmented error systems consisting of the position and translational velocity tracking errors of leader i and N follower UAV's as

$$e_\rho = \left[ e_{i\rho}^T \ e_{j\rho}^T \Big|_{j=1} \dots e_{j\rho}^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)}$$

$$e_v = \left[ e_{iv}^T \ e_{jv}^T \Big|_{j=1} \dots e_{jv}^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)}.$$

Next, the transformation matrix (2) is augmented as

$$R_F = \text{diag} \left\{ R_i, R_j \Big|_{j=1}, \dots, R_j \Big|_{j=N} \right\} \in \Re^{3(N+1) \times 3(N+1)} \quad (51)$$

while the NN weights for the translational velocity error system are augmented as

$$\hat{W}_{cl} = \text{diag} \left\{ \hat{W}_{icl}, \hat{W}_{jcl} \Big|_{j=1}, \dots, \hat{W}_{jcl} \Big|_{j=N} \right\} \in \Re^{(N \cdot N_{cl} + N_{icl}) \times 3(N+1)}$$

$$\hat{\sigma}_{cl} = \left[ \hat{\sigma}_{icl}^T \ \hat{\sigma}_{jcl}^T \Big|_{j=1}, \dots, \hat{\sigma}_{jcl}^T \Big|_{j=N} \right]^T \in \Re^{(N \cdot N_{cl} + N_{icl})}$$

Now, using the augmented variables above, the augmented closed loop position and translational velocity error dynamics for the entire formation are written as

$$\dot{e}_\rho = -K_\rho e_\rho + (I - G_F) R_F e_v \quad (52)$$

$$\dot{e}_v = -K_v e_v + A_{cf} \tilde{W}_{cl}^T \hat{\sigma}_{cl} - K_\rho G_F R_F e_v + \xi_{cl} \quad (53)$$

where  $A_{cf} = \text{diag} \left\{ A_{ic}, A_{jc} \Big|_{i=1}, \dots, A_{jc} \Big|_{i=N} \right\}$  with  $A_{ic}$  defined similarly to  $A_{jc}$  in terms of  $\bar{\Theta}_{id}$ ,  $K_\rho = \text{diag} \left\{ K_{ip}, K_{jp} \Big|_{j=1}, \dots, K_{jp} \Big|_{j=N} \right\}$ ,  $K_v = \text{diag} \left\{ K_{iv}, K_{jv} \Big|_{j=1}, \dots, K_{jv} \Big|_{j=N} \right\}$ , GF is a constant matrix relating to the formation interconnection errors defined as

$$G_F = [0 \ 0; F_T \ 0] \in \Re^{(N+1) \times (N+1)} \quad (54)$$

and  $F_T \in \Re^{N \times N}$  is constant and dependent on the specific formation topology. For instance, in a string formation where each follower follows

the UAV directly in front of it, follower 1 tracks leader i, follower 2 tracks follower 1, etc., and  $F_T$  becomes the identity matrix.

In a similar manner, we define augmented error systems for the virtual controller, orientation, and angular velocity tracking systems as

$$\begin{aligned}\tilde{\Theta}_d^b &= \left[ (\tilde{\Theta}_{id}^b)^T, (\tilde{\Theta}_{jd}^b)^T \Big|_{j=1} \dots (\tilde{\Theta}_{jd}^b)^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)}, \quad \tilde{\omega}_d = \left[ \tilde{\omega}_{id}^T, \tilde{\omega}_{jd}^T \Big|_{j=1} \dots \tilde{\omega}_{jd}^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)}, \\ e_\Theta &= \left[ e_{i\Theta}^T \ e_{j\Theta}^T \Big|_{j=1} \dots e_{j\Theta}^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)}, \quad e_\omega = \left[ e_{i\omega}^T \ e_{j\omega}^T \Big|_{j=1} \dots e_{j\omega}^T \Big|_{j=N} \right]^T \in \Re^{3(N+1)},\end{aligned}\quad (55)$$

respectively. It is straight forward to verify that the error dynamics of the augmented variables (55) takes the form of (30), (31), (25), and (36), respectively, but written in terms of the augmented variables (55).

*Theorem 3.3.1:* (UAV Formation Stability) Given the leader-follower criterion of (8) with 1 leader and N followers, let the hypotheses of Theorem 3.1.1 hold. Let the virtual control system for the leader i be defined similarly to (28) and (29) with the virtual control NN update law defined similarly to (37). Let control velocity and desire pitch and roll for the leader be given by (45), (47), and (48), respectively, along with the thrust and rotation torque vector defined by (49) and (50), respectively, and let the control NN update law be defined identically to (38). Then, the position, orientation, and velocity tracking errors, the virtual control estimation errors, and the NN weights for each NN for the entire formation are all SGUUB.

*Proof:* Consider the following positive definite Lyapunov candidate

$$V = V_\Omega + V_c, \quad (56)$$

where

$$\begin{aligned}V_\Omega &= \frac{1}{2} \tilde{\Theta}_d^b T \bar{K}_\Omega \tilde{\Theta}_d^b + \frac{1}{2} \tilde{\omega}_d^T \tilde{\omega}_d + \frac{1}{2} \text{tr}\{\tilde{W}_\Omega^T F_\Omega^{-1} \tilde{W}_\Omega\} \\ V_c &= \frac{1}{2} e_\rho^T e_\rho + \frac{1}{2} e_\Theta^T e_\Theta + \frac{1}{2} e_v^T e_v + \frac{1}{2} e_\omega^T J e_\omega + \frac{1}{2} \text{tr}\{\tilde{W}_c^T F_c^{-1} \tilde{W}_c\}\end{aligned}$$

with  $\bar{K}_\Omega = \text{diag}\{\bar{K}_{\Omega I}, \bar{K}_{\Omega I} \Big|_{j=1} \dots \bar{K}_{\Omega I} \Big|_{j=N}\}$ ,  $\tilde{W}_\Omega = \text{diag}\{\tilde{W}_{\Omega I}, \tilde{W}_{\Omega I} \Big|_{j=1} \dots \tilde{W}_{\Omega I} \Big|_{j=N}\}$ ,  $J = \text{diag}\{J_i, J_i \Big|_{j=1} \dots J_i \Big|_{j=N}\}$ , and

$\tilde{W}_c = \text{diag}\{\tilde{W}_{ic}, \tilde{W}_{jc} \Big|_{i=1} \dots \tilde{W}_{jc} \Big|_{i=N}\}$ . The first derivative of (56) with respect to time is given by  $\dot{V} = \dot{V}_\Omega + \dot{V}_c$  and performing similar steps as those used in (40)-(41) reveals

$$\dot{V}_\Omega \leq -\left( \bar{K}_{\Omega 2\min} - \frac{N_\Omega}{\kappa_{\Omega \min}} \right) \|\tilde{\Theta}_d^b\|^2 - \left( \frac{K_{\Omega 3\min}}{2} - \frac{N_\Omega}{\kappa_{\Omega \min}} \right) \|\tilde{\omega}_d\|^2 - \frac{\kappa_\Omega}{4} \|\tilde{W}_\Omega\|_F^2 + \eta_\Omega' \quad (57)$$

$$\dot{V}_c \leq -\frac{K_{\rho \min}}{2} \|e_\rho\|^2 - \frac{K_{\Theta \min}}{2} \|e_\Theta\|^2 - \left( \frac{K_{v \min}}{2} - \frac{\eta_1^2}{2K_{\rho \min}} - \eta_2 \right) \|e_{j_v}\|^2 - \frac{K_{c \min}}{3} \|\tilde{W}_c\|_F^2 - \left( \frac{K_{\omega \min}}{3} - \frac{(N+1)T_{\max}^2}{2K_{\Theta \min}} \right) \|e_\omega\|^2 + \frac{3K_{\omega \min}}{4} \|\tilde{\omega}_d\|^2 + \frac{3N_c}{4K_{c \min}} \|\tilde{\omega}_d\|^2 + \eta_c \quad (58)$$

where  $\bar{K}_{\Omega 2 \min}$  is the minimum singular value of  $\bar{K}_{2\Omega}$ ,  $K_{\Omega \min}$  is the minimum singular value of  $K_\Omega = \text{diag}\{K_{\Omega I}, K_{\Omega I}|_{j=1} \dots K_{\Omega I}|_{j=N}\}$  with  $I$  being the identity matrix,  $K_{\Omega 3 \min}$  is the minimum singular value of  $K_{\Omega 3} = \text{diag}\{K_{\Omega 3 I}, K_{\Omega 3 I}|_{j=1} \dots K_{\Omega 3 I}|_{j=N}\}$ ,  $N_\Omega$  is the number of hidden layer neurons in the augmented virtual control system, and  $\eta_\Omega$  is a computable constant based on  $\eta_{i\Omega}$  and  $\eta_{j\Omega}$ ,  $j = 1, \dots, N$ . Similarly,  $K_{\rho \min}, K_{\Theta \min}, K_{v \min}, K_{\omega \min}$ , and  $K_{c \min}$  are the minimum singular values of the augmented gain matrices  $K_\rho, K_\Theta, K_v, K_\omega$ , and  $K_c$  respectively, where  $\eta_1 = R_{F \max} \sqrt{1+2N}$ ,  $\eta_2 = K_{\rho \min} R_{F \max} \sqrt{N}$  are known computable constants and  $\eta_c$  is a computable constant dependent on  $\eta_{ic}$  and  $\eta_{jc}$ ,  $j = 1, \dots, N$ . Now, using (57) and (58), an upper bound for  $\dot{V}$  is found to be

$$\dot{V}_\Omega \leq -\left( \bar{K}_{\Omega 2 \min} - \frac{N_\Omega}{K_{\Omega \min}} \right) \|\tilde{\omega}_d\|^2 - \left( \frac{K_{\Omega 3 \min}}{2} - \frac{N_\Omega}{K_{\Omega \min}} - \frac{3K_{\omega \min}}{4} - \frac{3N_c}{4K_{c \min}} \right) \|\tilde{\omega}_d\|^2 - \frac{K_\Omega}{4} \|\tilde{W}_\Omega\|_F^2 - \frac{K_{c \min}}{3} \|\tilde{W}_c\|_F^2 - \frac{K_{\rho \min}}{2} \|e_\rho\|^2 - \frac{K_{\Theta \min}}{2} \|e_\Theta\|^2 - \left( \frac{K_{v \min}}{2} - \frac{\eta_1^2}{2K_{\rho \min}} - \eta_2 \right) \|e_{j_v}\|^2 - \left( \frac{K_{\omega \min}}{3} - \frac{(N+1)T_{\max}^2}{2K_{\Theta \min}} \right) \|e_\omega\|^2 + \eta_c + \eta_\Omega \quad (59)$$

Finally, (59) is less than zero provided

$$\bar{K}_{\Omega 2 \min} > \frac{N_\Omega}{K_{\Omega \min}}, \quad K_{\Omega 3 \min} > \frac{2N_\Omega}{K_{\Omega \min}} + \frac{3K_{\omega \min}}{2} + \frac{3N_c}{2K_{c \min}}, \quad K_{v \min} > \frac{\eta_1^2}{K_{\rho \min}} + 2\eta_2, \quad K_{\omega \min} > \frac{3(N+1)T_{\max}^2}{2K_{\Theta \min}} \quad (60)$$

and the following inequalities hold:

$$\begin{aligned} \|\tilde{\omega}_d\| &> \sqrt{\frac{\eta_\Omega + \eta_c}{\bar{K}_{\Omega 2 \min} - N_\Omega / K_{\Omega \min}}} \quad \text{or} \quad \|\tilde{W}_c\|_F &> \sqrt{\frac{3(\eta_\Omega + \eta_c)}{K_{c \min}}} \quad \text{or} \quad \|e_{j_v}\| &> \sqrt{\frac{2(\eta_\Omega + \eta_c)}{K_{v \min} - \eta_1^2 / K_{\rho \min} - 2\eta_2}} \\ \text{or} \quad \|e_\rho\| &> \sqrt{\frac{2(\eta_\Omega + \eta_c)}{K_{\rho \min}}} \quad \text{or} \quad \|e_\Theta\| &> \sqrt{\frac{2(\eta_\Omega + \eta_c)}{K_{\Theta \min}}} \quad \text{or} \quad \|\tilde{W}_\Omega\|_F &> \sqrt{\frac{4(\eta_\Omega + \eta_c)}{K_{\Omega \min}}} \\ \text{or} \quad \|e_\omega\| &> \sqrt{\frac{\eta_\Omega + \eta_c}{K_{\omega \min} / 3 - (N+1)T_{\max}^2 / (2K_{\Theta \min})}} \quad \text{or} \quad \|\tilde{\omega}_d\| &> \sqrt{\frac{\eta_\Omega + \eta_c}{\frac{K_{\Omega 3 \min}}{2} - \frac{N_\Omega}{K_{\Omega \min}} - \frac{3K_{\omega \min}}{4} - \frac{3N_c}{4K_{c \min}}}} \end{aligned} \quad (61)$$

Therefore, it can be concluded using standard extensions of Lyapunov theory (Lewis et al., 1999) that  $V$  is less than zero outside of a compact set, revealing the position, orientation, and velocity tracking errors, the virtual

control estimation errors, and the NN weights for each NN for the entire formation are all SGUUB.

*Remark 3:* The conclusions of Theorem 3.3.1 are independent of any specific formation topology, and the Lyapunov candidate (56) represents the most general form required show the stability of the entire formation. Examining (60) and (61), the minimum controller gains and error bounds are observed to increase with the number of follower UAV's,  $N$ . These results are not surprising since increasing number of UAV's increases the sources of errors which can be propagated throughout the formation.

*Remark 4:* Once a specific formation as been decided and the form of  $F_T$  is set, the results of Theorem 3.3.1 can be reformulated more precisely. For this case, the stability of the formation is proven using the sum of the individual Lyapunov candidates of each UAV as opposed to using the augmented error systems (51)-(55).

## **OPTIMIZED ENERGY-DELAY SUB-NETWORK ROUTING (OEDSR) PROTOCOL FOR UAV LOCALI- ZATION AND DISCOVERY**

In the previous section, a group of UAV's was modeled as a nonlinear interconnected system. We have shown that the basic formation is stable and each follower achieves its separation, angle of incidence, and bearing relative to its leader with a bounded error. The controller assignments for the UAV's can be represented as a graph where a directed edge from the leader to the followers denotes a controller for the followers while the leader is trying to track a desired trajectory. The shape vector consists of separations and orientations which in turn determines the relative positions of the follower UAV's with respect to its leader.

Then, a group of  $N$  UAV's is built on two networks: a physical network that captures the constraints on the dynamics of the lead UAV and a sensing and communication network, preferably wireless, that describes information flow, sensing and computational aspects across the group. The design of the graph is based on the task in hand. In this graph, nodes and edges represent UAV's and control policies, respectively. Any such graph can be described by its adjacency matrix (Das et al., 2002).

In order to solve the leader-follower criterion (8), ad hoc networks are formed between the leader(s) and the follower(s), and the position of each UAV in the formation must be determined on-line. This network is

dependent upon the sensing and communication aspects. As a first step, a leader is elected similar to the case of multi-robot formations (Das et al., 2002) followed by the discovery process in which the sensory information and physical networks are used to establish a wireless network. The outcome of the leader election process must be communicated to the followers in order to construct an appropriate shape. To complete the leader-follower formation control task (8), the controllers developed in the previous section require only a single-hop protocol; however, a multi-hop architecture is useful to relay information throughout the entire formation like the outcome of the leader election process, changing tasks, changing formations, as well alerting the UAV's of approaching moving obstacles that appear in a sudden manner.

The optimal energy-delay sub-network routing (OEDSR) protocol (Jagannathan, 2007) allows the UAV's to communicate information throughout the formation wirelessly using a multi-hop manner where each UAV in the formation is treated as a hop. The energy-delay routing protocol can guarantee information transfer while minimizing energy and delay for real-time control purposes even for mobile ad hoc networks such as the case of UAV formation flying.

We envision four steps to establish the wireless ad hoc network. As mentioned earlier, leader election process is the first step. The discovery process is used as the second step where sensory information and the physical network are used to establish a spanning tree. Since this is a multi-hop routing protocol, the communication network is created on-demand unlike in the literature where a spanning tree is utilized. This on-demand nature would allow the UAV's to be silent when they are not being used in communication and generate a communication path when required. The silent aspect will reduce any inference to others. Once a formation becomes stable, then a tree can be constructed until the shape changes. The third step will be assignment of the controllers online to each UAV based on the location of the UAV. Using the wireless network, localization is used to combine local sensory information along with information obtained via routing from other UAV's in order to calculate relative positions and orientations. Alternatively, range sensors provide relative separations, angles of incidence, and bearings. Finally cooperative control allows the graph obtained from the network to be refined. Using this graph theoretic formulation, a group is modeled by a tuple  $\Gamma = (\vartheta, P, H)$ , where  $\vartheta$  is the reference trajectory of the robot,  $P$  represents the shape vectors describing

the relative positions of each vehicle with respect to the formation reference frame (leader), and  $H$  is the control policy represented as a graph where nodes represent UAV and edges represent the control assignments. Next, we describe the OEDSR routing protocol where each UAV will be referred to as a “node.”

In OEDSR, sub-networks are formed around a group of nodes due to an activity, and nodes wake up in the sub-networks while the nodes elsewhere in the network are in sleep mode. An appropriate percentage of nodes in the sub-network are elected as cluster heads (CHs) based on a metric composed of available energy and relative location to an event (Jagannathan, 2007) in each sub-network. Once the CHs are identified and the nodes are clustered relative to the distance from the CHs, the routing towards the formation leader (FL) is initiated. First, the CH checks if the FL is within the communication range. In such case, the data is sent directly to the FL. Otherwise, the data from the CHs in the sub-network are sent over a multi-hop route to the FL. The proposed routing algorithm is fully distributed since it requires only local information for constructing routes, and is proactive adapting to changes in the network. The FL is assumed to have sufficient power supply, allowing a high power beacon from the FL to be sent such that all the nodes in the network have knowledge of the distance to the FL. It is assumed that all UAV's in the network can calculate or measure the relative distance to the FL at any time instant using the formation graph information or local sensory data. Though the OEDSR protocol borrows the idea of an energy-delay metric from OEDR (Jagannathan, 2007), selection of relay nodes (RN) does not maximize the number of two hop neighbors. Here, any UAV can be selected as a RN, and the selection of a relay node is set to maximize the link cost factor which includes distance from the FL to the RN.

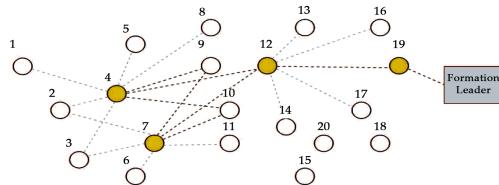
### **Selection of an Optimum Relay-Node-Based link cost factor**

Knowing the distance information at each node will allow the UAV to calculate the Link Cost Factor (LCF). The link cost factor from a given node to the next hop node ‘ $k$ ’ is given by (62) where  $D_k$  represent the delay that will be incurred to reach the next hop node in range, the distance between the next hop node to the FL is denoted by  $k \Delta x_k$ , and the remaining energy,  $E_k$ , at the next hop node are used in calculation of the link cost as

$$LCF_k = \frac{E_k}{D_k \cdot \Delta x_k} \quad (62)$$

In equation (62), checking the remaining energy at the next hop node increases network lifetime; the distance to the FL from the next hop node reduces the number of hops and end-to-end delay; and the delay incurred to reach the next hop node minimizes any channel problems. When multiple RNs are available for routing of the information, the optimal RN is selected based on the highest LCF. These clearly show that the proposed OEDSR protocol is an on demand routing protocol. For detailed discussion of OEDSR refer to (Jagannathan, 2007). The route selection process is illustrated through the following example. This represents sensor data collected by a follower UAV for the task at hand that is to be transmitted to the FL.

## Routing Algorithm through an Example



**Figure 2.** Relay node selection

Consider the formation topology shown in Figure 2. The link cost factors are taken into consideration to route data to the FL. The following steps are implemented to route data using the OEDSR protocol:

1. Start with an empty relay list for source UAV  $n$ ;  $\text{Relay}(n) = \{\}$ . Here UAV  $n_4$  and  $n_7$  are CHs.
2. First, CH  $n_4$  checks with which nodes it is in range with. In this case, CH  $n_4$  is in range with nodes  $n_1, n_2, n_3, n_5, n_8, n_9, n_{12}$ , and  $n_{10}$ .
3. The nodes  $n_1, n_2$ , and  $n_3$  are eliminated as potential RNs because the distance from them to the FL is greater than the distance from CH  $n_4$  to the FL.
4. Now, all the nodes that are in range with CH  $n_4$  transmit RESPONSE packets and CH  $n_4$  makes a list of possible RNs, which in this case are  $n_5, n_8, n_9, n_{12}$ , and  $n_{10}$ .
5. CH  $n_4$  sends this list to CH  $n_7$ . CH  $n_7$  checks if it is range with any of the nodes in the list.
6. Nodes  $n_9, n_{10}$ , and  $n_{12}$  are the nodes that are in range with both CH  $n_4$  and  $n_7$ . They are selected as the potential common RNs.

7. The link cost factors for  $n_9$ ,  $n_{10}$ , and  $n_{12}$  are calculated.
8. The node with the maximum value of LCF is selected as the RN and assigned to Relay(n). In this case,  $\text{Relay}(n)=\{n_{12}\}$ .
9. Now UAV  $n_{12}$  checks if it is in direct range with the FL, and if it is, then it directly routes the information to the FL.
10. Otherwise,  $n_{12}$  is assigned as the RN, and all the nodes that are in range with node  $n_{12}$  and whose distance to the FL is less than its distance to the FL are taken into consideration. Therefore, UAV's  $n_{13}$ ,  $n_{16}$ ,  $n_{19}$ , and  $n_{17}$  are taken into consideration.
11. The LCF is calculated for  $n_{13}$ ,  $n_{16}$ ,  $n_{19}$ ,  $n_{14}$ , and  $n_{17}$ . The node with the maximum LCF is selected as the next RN. In this case  $\text{Relay}(n)=\{n_{19}\}$ .
12. Next the RN  $n_{19}$  checks if it is in range with the FL. If it is, then it directly routes the information to the FL. In this case,  $n_{19}$  is in direct range, so the information is sent to the FL directly.

## Optimality Analysis for OEDSR

To prove that the proposed route created by OEDSR protocol is optimal in all cases, it is essential to show it analytically.

*Assumption 1:* It is assumed that all UAV's in the network can calculate or measure the relative distance to the FL at any time instant using the formation graph information or local sensory data.

*Theorem 4.3.1:* The link cost factor-based routing generates viable RNs to the FL.

*Proof:* Consider the following two cases

**Case I:** When the CHs are one hop away from the FL, the CH selects the FL directly. In this case, there is only one path from the CH to the FL. Hence, OEDSR algorithm does not need to be used.

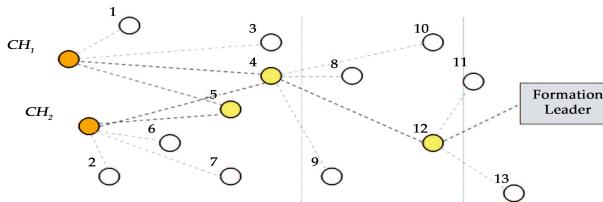
**Case II:** When the CHs have more than one node to relay information, the OEDSR algorithm selection criteria are taken into account. In Figure 3, there are two CHs, CH1 and CH2. Each CH sends signals to all the other nodes in the network that are in range. Here, CH1 first sends out signals to  $n_1$ ,  $n_3$ ,  $n_4$ , and  $n_5$  and makes a list of information about the potential RN. The list is then forwarded to CH2. CH2 checks if it is in range with any of the nodes in the list. Here,  $n_4$  and  $n_5$  are selected as potential common RNs. A single node must be selected from both  $n_4$  and  $n_5$  based on the OEDSR link cost factor. The cost to reach  $n$  from CH is given by (2). So based on the

OEDSR link cost factor,  $n_4$  is selected as the RN for the first hop. Next,  $n_4$  sends signals to all the nodes it has in range, and selects a node as RN using the link cost factor. The same procedure is carried on till the data is sent to the FL. Lemma 4.3.2: The intermediate UAV's on the optimal path are selected as RNs by the previous nodes on the path. Proof: A UAV is selected as a RN only if it has the highest link cost factor and is in range with the previous node on the path. Since OEDSR maximizes the link cost factor, intermediate nodes that satisfy the metric on the optimal path are selected as RNs.

*Lemma 4.3.3:* A UAV can correctly compute the optimal path (with lower end to end delay and maximum available energy) for the entire network topology.

*Proof:* When selecting the candidate RNs to the CHs, it is ensured that the distance from the candidate RN to the FL is less than the distance from the CH to the FL. When calculating the link cost factor, available energy is divided by distance and average end-to-end delay to ensure that the selected nodes are in range with the CHs and close to the FL. This helps minimize the number of multi-point RNs in the network.

*Theorem 4.3.4:* OEDSR protocol results in an optimal route (the path with the maximum energy, minimum average end-to-end delay and minimum distance from the FL) between the CHs and any source destination.



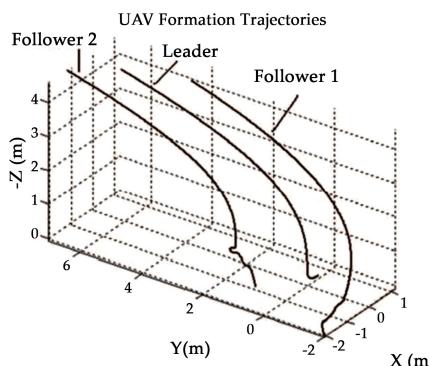
**Figure 3.** Link cost calculation

## SIMULATION RESULTS

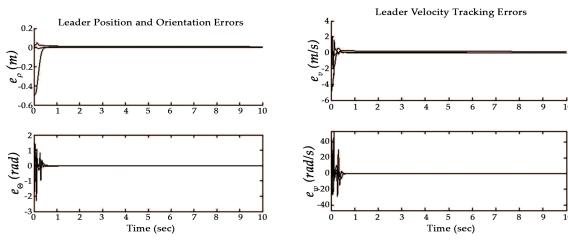
A wedge formation of three identical quadrotor UAV's is now considered in MATLAB with the formation leader located at the apex of the wedge. In the simulation, follower 1 should track the leader at a desired separation  $s_{\text{jid}} = 2$  m , desired angle of incidence ( $0$  rad)  $\alpha_{\text{jid}} =$  , and desired bearing ( $3$  rad)  $\beta_{\text{jid}} = \pi$  while follower 2 tracks the leader at a desired separation  $s_{\text{jid}} = 2$  m desired

angle of incidence, ( $10 \text{ rad}$ )  $\alpha_{\text{jid}} = \pi$ , and desired bearing ( $3 \text{ rad}$ )  $\beta_{\text{jid}} = \pi$ . The desired yaw angle for the leader and thus the formation is selected to be  $\sin(3.0 t) \psi_d = \pi$ . The inertial parameters of the UAV's are taken to be as  $m = 9.0 \text{ kg}$  and  $2 J = \text{diag}\{63.0, 42.0, 32.0\} \text{ kg m}^2$ , and aerodynamic friction is modeled as in (Dierks and Jagannathan, 2008). The parameters outlined in Section 3.3 are communicated from the leader to its followers using single hop communication whereas results for OEDSR in a mobile environment can be found in (Jagannathan, 2007). Each NN employs 5 hidden layer neurons, and for the leader and each follower, the control gains are selected to be,  $K_{\Omega_1} = .23$   $K_{\Omega_2} = .80$   $K_{\Omega_3} = 20$ ,  $K = \text{diag}\{30, 10, 10\} \rho$ ,  $k_{v1} = .10$   $k_{v2} = .10$   $k_{v3} = 30$ ,  $K = \text{diag}\{30, 30, 30\} \Theta$ , and  $K = \text{diag}\{25, 25, 25\} \omega$ . The NN parameters are selected as,  $F_\Omega = .10$   $\kappa_\Omega = 1$ , and  $F_c = .10$   $\kappa_c = 1.0$ , and the maximum desired pitch and roll values are both selected as  $2\pi/5$ .

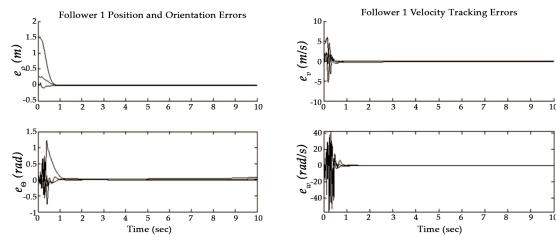
Figure 4 displays the quadrotor UAV formation trajectories while Figures 5-7 show the kinematic and dynamic tracking errors for the leader and its followers. Examining the trajectories in Figure 4, it is important to recall that the bearing angle,  $\beta_{ji}$ , is measured in the inertial reference frame of the follower rotated about its yaw angle. Examining the tracking errors for the leader and its followers in Figures 5-7, it is clear that all states track their desired values with small bounded errors as the results of Theorem 3.3.1 suggest. Initially, errors are observed in each state for each UAV, but these errors quickly vanish as the virtual control NN and the NN in the actual control law learns the nonlinear UAV dynamics. Additionally, the tracking performance of the underactuated states  $x_v$  and  $y_v$  implies that the desired pitch and roll, respectively, as well as the desired angular velocities generated by the virtual control system are satisfactory.



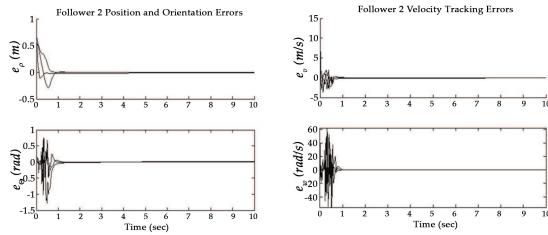
**Figure 4.** Quadrotor UAV formation trajectories



**Figure 5.** Tracking errors for the formation leader



**Figure 6.** Tracking errors for follower 1



**Figure 7.** Tracking errors for follower 2

## CONCLUSIONS

A new framework for quadrotor UAV leader-follower formation control was presented along with a novel NN formation control law which allows each follower to track its leader without the knowledge of dynamics. All six DOF are successfully tracked using only four control inputs while in the presence of unmodeled dynamics and bounded disturbances. Additionally, a discovery and localization scheme based on graph theory and ad hoc networks was presented which guarantees optimal use of the UAV's communication links. Lyapunov analysis guarantees SGUUB of the entire formation, and numerical results confirm the theoretical conjectures.

## REFERENCES

1. Das, A.; Spletzer, J.; Kumar, V.; & Taylor, C. (2002). Ad hoc networks for localization and control of mobile robots, Proceedings of IEEE Conference on Decision and Control, pp. 2978-2983, Philadelphia, PA, December 2002
2. Desai, J.; Ostrowski, J. P.; & Kumar, V. (1998). Controlling formations of multiple mobile robots, Proceedings of IEEE International Conference on Robotics and Automation, pp. 2964-2869, Leuven, Belgium, May 1998
3. Dierks, T. & Jagannathan, S. (2008). Neural network output feedback control of a quadrotor UAV, Proceedings of IEEE Conference on Decision and Control, To Appear, Cancun Mexico, December 2008
4. Fierro, R.; Belta, C.; Desai, J.; & Kumar, V. (2001). On controlling aircraft formations, Proceedings of IEEE Conference on Decision and Control, pp. 1065-1079, Orlando, Florida, December 2001
5. Galzi, D. & Shtessel, Y. (2006). UAV formations control using high order sliding modes, Proceedings of IEEE American Control Conference, pp. 4249-4254, Minneapolis, Minnesota, June 2006
6. Gu, Y.; Seanor, B.; Campa, G.; Napolitano M.; Rowe, L.; Gururajan, S.; & Wan, S. (2006). Design and flight testing evaluation of formation control laws. *IEEE Transactions on Control Systems Technology*, Vol. 14, No. 6, (November 2006) page numbers (1105- 1112)
7. Jagannathan, S. (2007). *Wireless Ad Hoc and Sensor Networks*, Taylor & Francis, ISBN 0-8247- 2675-8, Boca Raton, FL
8. Lewis, F.L.; Jagannathan, S.; & Yesilderek, A. (1999). *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, ISBN 0-7484-0596-8, Philadelphia, PA
9. Neff, A.E.; DongBin, L.; Chitrakaran, V.K.; Dawson, D.M.; & Burg, T.C. (2007). Velocity control for a quad-rotor uav fly-by-camera interface, Proceedings of the IEEE Southeastern Conference, pp. 273-278, Richmond, VA, March 2007
10. Saffarian, M. & Fahimi, F. (2008). Control of helicopters' formation using non-iterative nonlinear model predictive approach, Proceedings of IEEE American Control Conference, pp. 3707-3712, Seattle, Washington, June 2008
11. Xie, F.; Zhang, X.; Fierro, R; & Motter, M. (2005). Autopilot based nonlinear UAV formation controller with extremum-seeking, Proceedings of IEEE Conference on Decision and Control, pp 4933-4938, Seville, Spain, December 2005

# CHAPTER

# 10

## Dubins Waypoint, Navigation of Small-Class Unmanned Aerial Vehicles

---

**Larry M. Silverberg, Dahan Xu**

Mechanical and Aerospace Engineering, North Carolina State University,  
Raleigh, USA

### ABSTRACT

This paper considers a variation on the Dubins path problem and proposes an improved waypoint navigation (WN) algorithm called Dubins waypoint navigation (DWN). Based on the Dubins path problem, an algorithm is developed that is updated in real-time with a horizon of three waypoints. The purpose of DWN is to overcome a problem that we find in existing WN for

---

**Citation:** Silverberg, L. and Xu, D. (2019), "Dubins Waypoint, Navigation of Small-Class Unmanned Aerial Vehicles". *Open Journal of Optimization*, **8**, 59-72.  
doi: 10.4236/ojop.2019.82006.

**Copyright:** © 2019 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

small-class fixed-wing unmanned aerial vehicles (UAV) of not accurately reaching waypoints. This problem results at times in high overshoot and, in the presence of wind disturbances, it can cause a vehicle to miss the waypoint and swirl around it. To prevent this, the DWN creates “new waypoints” that are in the background, called turning points. Examples illustrate the improvement of the performance of WN achieved using the DWN algorithm in terms of the targeting of waypoints while reducing fuel and time.

**Keywords:** Dubins Path, Waypoint Navigation, Unmanned Aerial Vehicles, Autonomy, Shortest Path, Fuel, Optimization

## INTRODUCTION

The user community of unmanned aerial vehicle (UAV) systems has been growing significantly; the commercial market net worth reached a reported \$8.3 billion in 2018 [1] , the largest growth in the commercial markets being in small-class (under 55 pounds) vehicles. This paper develops an algorithm for a subset of this class of UAV, in particular, for fixed-wing vehicles. Users of small-class UAV are presently navigating autonomously by open-source algorithms such as Mission Planner, Cape, and Pix4D. These navigational systems employ waypoint navigation (WN), wherein the user enters waypoints, whether a priori (static) or not (dynamic). The practice of WN in this community is a constraint to which development work adheres. The community is growing, and with that, is expected to demand higher accuracy. In RC flight, the racing community already requires reaching targets accurately and the growth in autonomous flight with its varied missions will create more demand to reach waypoints accurately.

The shortest path between two waypoints with a limited turning radius is called a Dubins path [2] . WN employing Dubins paths has been studied considerably. Ariff and Go conducted a thorough examination of the dynamics involved in WN with a focus on dynamic soaring algorithms [3] . Goerzen, Kong, and Mettler made rigorous comparisons between different path planning algorithms for unmanned vehicle navigation and identified advantages and disadvantages [4] . Manyam, et al. expanded the Dubins path problem from two waypoints to three waypoints where all three points and their headings are constrained [5] . Wolek and Woolsey implemented the Dubins path problem with estimates of unsteady velocity disturbances [6] . McGee and Hedrick considered the situations in which the disturbance is less significant and more predictable [7] . The vehicle was placed in a moving frame as a means of finding a piecewise continuous optimal path.

Milutinovic et al. applied the characteristics of a Dubins vehicle to the circumnavigation algorithm, together with a feedback controller to improve the vehicle's performance [8]. Ketema and Zhao formulated the optimization problem of path planning under wind disturbances and identified situations in which the target points may be not reachable [9]. Meyer, Isaiah and Shima examined and proposed a solution for a vehicle to intercept a moving target with a Dubins path [10]. Criteria were given whether a target can or cannot be intercepted and an analytical solution was obtained by a path elongation algorithm. Wang et al. proposed a solution using a Dubins path to solve the formation rendezvous problem of multiple UAVs [11]. Medeiros and Urrutia optimized both path length and direction changes for a Dubins path problem, by dividing the path planning process into several stages, bringing in the algorithm of the travelling salesman's problem [12].

This paper considers a variation on the Dubins path problem and proposes an improved navigation algorithm called Dubins waypoint navigation (DWN). The proposed DWN algorithm considers a current vehicle state (position and velocity) along with the next two prescribed waypoints to plan the vehicle's path. Compared with traditional WN, one more waypoint is involved in the determination of the path, enabling consideration of the manner in which the vehicle should circle around its previous waypoint. The objective is similar to the Dubins path problem in that a shortest path and a minimum fuel path between waypoints consists of straight-line segments and arcs. The major difference is that in DWN the heading at the waypoints are not specified and we consider three points instead of two. Also, for simplicity, the paper only considers the case in which the vehicle's turning radius is constant.

The interest here lies in an algorithm that is updated in real-time with a horizon of three waypoints. The proposed DWN algorithm employs a current vehicle position and heading for the first waypoint, uses a prescribed vehicle position for the second waypoint, not specifying its heading, and uses a third waypoint to determine how the vehicle should circle around the second waypoint.

The purpose of DWN is to overcome a problem that we find in existing WN for small-class fixed-wing vehicles of not accurately reaching waypoints. This problem results from the present use of a waypoint radius as a threshold that determines when to move on to a next waypoint. Presently, a vehicle approaches a first waypoint and switches to navigating to a second waypoint when the vehicle is within a waypoint radius around the first waypoint,

thereby not accurately reaching the first waypoint. This leads the operator to desire a smaller waypoint radius. However, the smaller waypoint radius causes several problems. First, it causes high overshoot. Secondly, in the presence of wind disturbances, it can cause a vehicle to miss the waypoint and swirl around it.

When correcting this problem, it became important to preserve the simplicity of prescribing waypoints without adding operator complexity. Therefore, the DWN does not have to change anything in the foreground, including the operator's specified waypoints. Instead, the DWN creates "new waypoints" that can be in the background, called turning points, as explained in the method section.

Section 2 describes the DWN algorithm. Section 3 compares performances of WN and DWN with and without unknown wind disturbances, and the paper finishes in Section 4 with a summary and conclusions.

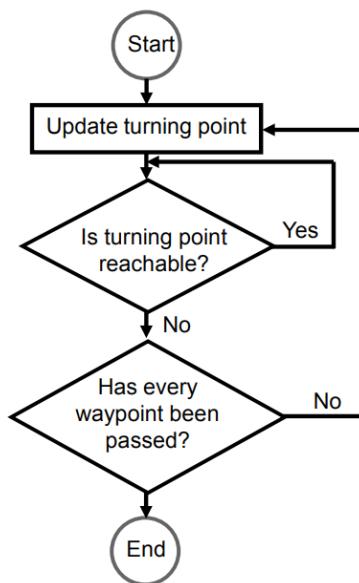
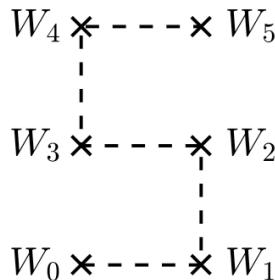
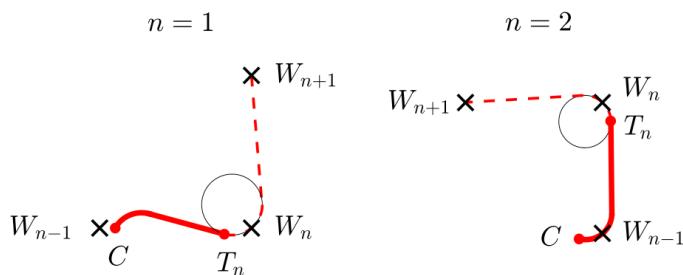
## METHOD

### Flow Chart

This section develops the DWN. The DWN eliminates waypoint circles and, instead, employs the new concept of turning points along with the new criterion for updating turning points based on whether or not a turning point is reachable. See Figure 1 showing the DWN flow chart. As shown, the DWN approaches navigation as an optimization problem with in a local optimization space that consists of a current point and two future waypoints. At any instant, the vehicle seeks to reach a turning point that is determined by minimizing fuel along the path up to the second future waypoint constituting the horizon. The trajectory, and hence the optimization problem, is updated once the turning point is unreachable. The reachability condition is determined from the vehicle's position, heading, and turning radius. For the purposes of real-time implementation, the trajectory may be updated more frequently than when the turning point is updated (like under the conditions of high winds). The stated optimization problem yields desired paths that are determined in closed-form.

### Finding Turning Points

Consider the illustrative WN problem shown in Figure 2 and, in particular, the navigation to the second and third waypoints shown in Figure 3.

**Figure 1.** Flow chart of the DWN.**Figure 2.** Illustrative WN problem.**Figure 3.** Navigating to the 2<sup>nd</sup> and 3<sup>rd</sup> waypoints.

The vehicle starts at  $W_0$  and heads to  $W_1$ . At this point, the DWN considers waypoints  $W_0$ ,  $W_1$ , and  $W_2$ . The other points are beyond the horizon.

Figure 3 shows the paths from  $W_0$  to  $W_2$  (left drawing) and from  $W_1$  to  $W_3$  (right drawing).

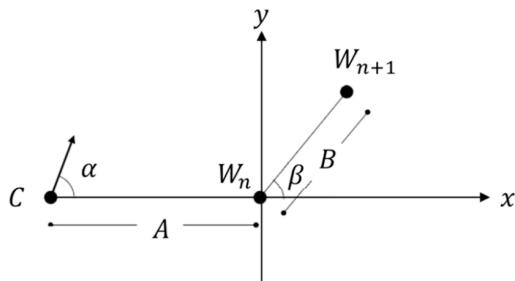
As shown,  $T_1$  denotes the turning point associated with  $W_1$ . It lies on the line tangent to a turning circle (not to be confused with the waypoint circle in classical WN) that has a turning radius  $R$ . The turning radius, specified by the user, is the desirable radius of the turns around waypoints. An optimization problem minimizes the fuel from  $W_0$  around and touching  $W_1$  to  $W_2$ . The orientation of the turning circle is free to rotate about  $W_1$  so the optimization problem is a minimization problem of fuel expressed in terms of the turning circle's orientation. The fuel consumed during a turn is greater than the fuel consumed while flying in a straight line over the same distance. Even though the minimization is over the distance extending to  $W_2$ , the DWN updates the vehicle's path once it reaches point  $T_1$ . Thus, the vehicle does not follow the section of the path from  $T_1$  to  $W_2$  (In Figure 3, the paths followed are solid lines and the paths not followed are dashed lines.) The DWN recalculates that not-followed section of the path in the next iteration. Under ideal conditions, DWN updates the planned path when the vehicle crosses  $T_1$ , at which point  $T_1$  is determined to be unreachable. Under real conditions, disturbances can result in the vehicle not reaching point  $T_1$  accurately. Point  $T_1$  is determined to be unreachable at a point  $C$  that is different but nearpoint  $T_1$ . Referring to the right drawing, the new horizon is set to  $W_3$  and the new turning point  $T_2$  is determined from the vehicle's current point  $C$  and current direction of flight and the locations of points  $W_2$  and  $W_3$ . As shown, the optimization problem now minimizes the fuel from  $C$  around and touching  $W_2$  to  $W_3$ . The optimization problem is now a minimization problem of fuel from  $C$  to  $W_3$  expressed in terms of the new turning circle's orientation. The DWN updates the iterations until the vehicle reaches its last waypoint.

## Key Parameters

Figure 4 shows the  $n^{\text{th}}$  iteration. The vehicle is heading to  $W_n$ . A coordinate system was set up so its x-axis points from  $C$  to  $W_n$  and its y-axis is perpendicular to x in the direction of  $W_{n+1}$ . The data is scaled such that the x and y coordinates represent non-dimensional lengths of distance divided by turning radius  $R$ ; equivalently  $R = 1$ . We perform all of the calculations after the geometric parameters are normalized with respect to  $R$ .

<sup>1</sup>The x axis, because it is along the line through C and  $W_n$ , can cause  $\beta$  to exceed  $90^\circ$ .

As shown,  $\alpha$  is the heading angle (between  $-180^\circ$  and  $180^\circ$ ) and  $\beta$  is the turn angle ( $\beta$  is between  $0^\circ$  and  $90^\circ$ )<sup>1</sup>. We assume that there are no sharp turns between specified waypoints (the angle between any two lines connecting three consecutive waypoints is never greater than  $90^\circ$ ). At point C, the vehicle turns counter-clockwise (CCW), or clockwise (CW). Likewise, it turns around  $W_n$  CCW or CW. In total, there are four cases: CCW-CCW, CCW-CW, CW-CCW, and CW-CW.



**Figure 4.** Set up of the coordinate system.

The interest lies in finding closed-form expressions for the paths as functions of A, B,  $\alpha$  and  $\beta$  by curve fitting. Toward this end, note that the minimum fuel paths are continuous with respect to these parameters for each case, individually. Discontinuities occur when transitioning from one case to another. Therefore, the curve-fitting needs to be performed for each case and the case that consumes the smallest amount of fuel yields the true minimum (See Figure 5).

## Optimization Problem

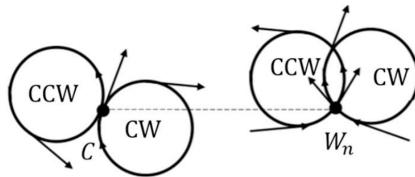
Figure 6 shows the geometry of the CW-CCW case. The fuel from C to  $W_{n+1}$  is a function of the orientation angle  $\theta$  of the turning circle. (The other cases, not shown, are similar.) The fuel is

$$Fu = w_a L_a + w_s L_s \quad (1)$$

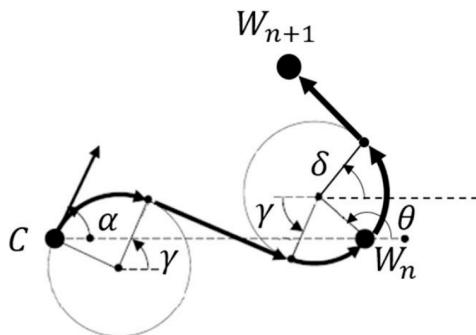
where  $L_a$  is the total length of the two arcs (around the first and second turning circles),  $L_s$  is the length of the two straight segments (after the first and second turning circles), and  $w_a$  and  $w_s$  are corresponding weights (in units of fuel per length). Letting  $w_a = w_s = 1$  (in which the weights have units of 1), yields the shortest path problem:

$$L = L_a + L_s \quad (2)$$

In the results section, we will show that the shortest path and minimum fuel solutions are nearly indistinguishable under a broad range of conditions, allowing the shortest path solution to approximate the minimum fuel solution. This is important because the shortest path solution is independent of the vehicle, increasing ease of implementation and the versatility of DWN.



**Figure 5.** Different directions of turning.



**Figure 6.** Geometry for the CW-CCW case.

Referring again to Figure 6, the first straight line segment starts at tangential point  $(x_1, y_1)$  and ends at turning point  $T_n = (x_2, y_2)$ . The second line segment starts at tangential point  $(x_3, y_4)$ . Note that we define the first circle by the current position C and the vehicle's heading; it is either CCW or CW. The location of the second circle depends on the orientation angle  $\theta$ . The second line segment is tangent to the second circle and passes through  $W_{n+1}$  and therefore is uniquely expressed in terms of the angle  $\delta$  shown, which, in turn, depends on  $\theta$ .

The coordinates  $x_1, y_1, x_2, y_2, x_3$ , and  $y_4$  for each of the four cases are given in Table 1 and Table 2.

By appropriately manipulating the geometric relationships, the path length (the objective function) becomes a function of the orientation angle

$\theta$  of the turning circle, and the solution of this 3-point optimization problem becomes a function of  $\theta$ . This part of the iteration can be solved off-line and curve-fit to determine  $T_n$ . There are two ways to fit  $T_n$ . One way is to directly fit  $T_n$  to A, B,  $\alpha$ , and  $\beta$ . The second way is to fit  $\theta$  to A, B,  $\alpha$ , and  $\beta$  followed by determining  $T_n$  from  $\theta$ . The second way of curve fitting the data will inherently lead to a more accurate curve fit than the first way because the relationship between  $T_n$  and  $\theta$  is rather complicated and its analytical relationship is available, too [13].

**Table 1.** Location of the end points of the first line segment.

|        | $x_1$                            | $y_1$   | $x_2$                       | $y_2$                       |
|--------|----------------------------------|---|-----------------------------|-----------------------------|
|        | $-A + \sin \alpha + \cos \gamma$ | $-\cos \alpha + \sin \gamma$  | $\cos \theta + \cos \gamma$ | $\sin \theta + \sin \gamma$ |
| CW-CW  |                                  | $\gamma = \arctan \frac{\sin \theta + \cos \alpha}{\cos \theta + A - \sin \alpha} + \frac{\pi}{2}$  |                             |                             |
|        | $-A + \sin \alpha + \cos \gamma$ | $-\cos \alpha + \sin \gamma$  | $\cos \theta + \cos \gamma$ | $\sin \theta + \sin \gamma$ |
| CW-CCW |                                  | $\gamma = \arctan \frac{\sin \theta + \cos \alpha}{\cos \theta + A - \sin \alpha} + \arccos \frac{2}{\sqrt{(\sin \theta + \cos \alpha)^2 + (\cos \theta + A - \sin \alpha)^2}}$ |                             |                             |
|        | $-A - \sin \alpha + \cos \gamma$ | $\cos \alpha + \sin \gamma$   | $\cos \theta - \cos \gamma$ | $\sin \theta + \sin \gamma$ |
| CCW-CW |                                  | $\gamma = \arctan \frac{\sin \theta - \cos \alpha}{\cos \theta + A + \sin \alpha} - \arccos \frac{2}{\sqrt{(\sin \theta - \cos \alpha)^2 + (\cos \theta + A + \sin \alpha)^2}}$ |                             |                             |
|        | $-A - \sin \alpha + \cos \gamma$ | $\cos \alpha + \sin \gamma$   | $\cos \theta - \cos \gamma$ | $\sin \theta - \sin \gamma$ |
| CW-CCW |                                  | $\gamma = \arctan \frac{\sin \theta - \cos \alpha}{\cos \theta + A + \sin \alpha} - \frac{\pi}{2}$  |                             |                             |

**Table 2.** Location of second line segment.

|                  | $x_3$                       | $y_3$   |
|------------------|-----------------------------|---|
|                  | $\cos \theta + \cos \delta$ | $\sin \theta + \sin \delta$   |
| CW-CW & CCW-CW   |                             | $\delta = \arctan \frac{B \cos \beta - \cos \theta}{B \sin \beta - \sin \theta} + \arccos \frac{1}{\sqrt{(B \cos \beta - \cos \theta)^2 + (B \sin \beta - \sin \theta)^2}}$ |
|                  | $\cos \theta + \cos \delta$ | $\sin \theta + \sin \delta$   |
| CW-CCW & CCW-CCW |                             | $\delta = \arctan \frac{B \cos \beta - \cos \theta}{B \sin \beta - \sin \theta} - \arccos \frac{1}{\sqrt{(B \cos \beta - \cos \theta)^2 + (B \sin \beta - \sin \theta)^2}}$ |

## Updating the Turning Point

As described earlier, the DWN guides a vehicle to a waypoint until it becomes

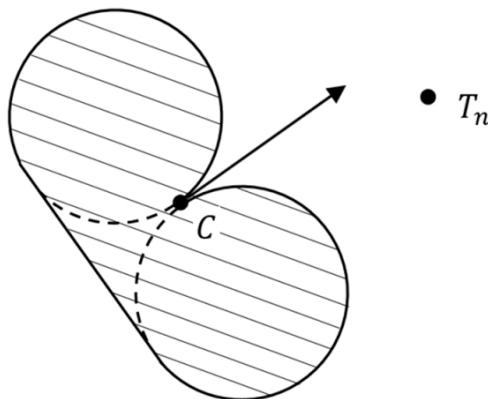
unreachable, at which point it updates the turning point. Figure 7 shows the reachability condition. As shown, a reachability zone consists of CCW and CW circles “enclosed” on the rear by a tangent line. The reachability zone is fixed to the vehicle. The vehicle is trying to reach waypoint  $T_n$ . At a given instant of time,  $T_n$  could be either inside or outside the reachability zone. (In the figure,  $T_n$  is outside of the reachability zone.). As soon as it is inside the reachability area,  $T_n$  becomes unreachable, and the DWN updates the turning point to  $T_{n+1}$ .

Note that the purpose of “enclosing” the CCW and CW with the line segment was to prevent the possibility of point C passing  $T_n$  undetected, which would otherwise be possible because the detection is not truly performed continuously but only at discrete instances of time.

Comparing the reachability zone used in DWN with the waypoint circle used in WN, first notice that the reachability zone, in the absence of disturbances, allows the vehicle to accurately reach a first waypoint, as opposed to beginning its turn to a second waypoint before reaching the first waypoint. Secondly, in WN, the operator can easily be fooled into the natural desire to select a waypoint radius that is smaller than the vehicle’s turning radius toward the goal of more closely reaching a waypoint. However, when doing so in the presence of a wind disturbance, this can result in missing the waypoint, finding it difficult to reach, and causing the vehicle to swirl around the waypoint. In the DWN, no waypoint radius is considered.

## Parameterization

We formulated the 3-point horizon optimization problem to keep the computational effort minimal for real-time implementation. To further reduce computational effort and for robustness, we parameterized the solution, effectively reducing it to a look-up table. Toward this end, we determined the orientation angle  $\theta$  of a turning circle as a function of the four parameters  $A$ ,  $B$ ,  $\alpha$ , and  $\beta$ . We parameterized the orientation angle by smoothly fitting the data to the four parameters separately for each case (CCW-CCW, CCW-CW, CW-CCW, and CW-CW). Toward distinguishing between the different cases, we needed to parameterize the boundaries of each of the cases in terms of the four parameters. In particular,  $\alpha$  and  $\beta$  transition from case to case due to changes in  $A$  and  $B$ . For example, consider the graph of  $\alpha$  versus  $\beta$  shown in Figure 8 for particular parameters ( $A = B = 4$ ,  $w_a = w_s = 1$ ).



**Figure 7. Examples of reachable and unreachable points.**

As shown, there are three boundaries and three interior regions. (It turns out that the CW-CW case never produces an optimal orientation angle.) We express the boundary of the turn angle  $\beta$  as a function of the heading angle  $\alpha$  in which its coefficients are expressed as a function of the distances  $A$  and  $B$  as follows:

$$\beta = \begin{cases} a_0 + a_1\delta + a_2\delta^2 & \alpha \geq -a_1/2a_2 \\ \frac{4a_0a_2 - a_1^2}{4a_2} & \alpha \leq -a_1/2a_2 \end{cases} \quad (3)$$

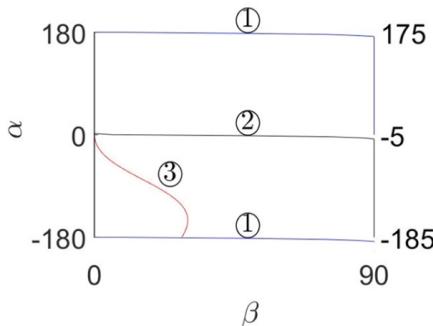
$$a_i = b_{0i} + b_{1i}A + b_{2i}A^2 + b_{3i}B + b_{4i}B^2 + b_{5i}AB \quad (i = 1, 2, 3) \quad (4)$$

We determined the coefficients  $b_{0i}$  through  $b_{6i}$  ( $i = 1, 2, 3$ ) separately for each of the cases. The real-time procedure of updating the turning points is as follows:

- 1) Calculate  $A$ ,  $B$ ,  $\alpha$ , and  $\beta$  at an instant of time from a current state of the vehicle.
- 2) Calculate the coefficients associated with the boundaries between the interior regions (See [13]).
- 3) Determine the interior region in which the vehicle lies.
- 4) Calculate the orientation angle  $\theta$  of the next turning circle (See [13]).
- 5) Calculate the next turning point  $T_n$  (expressed in terms of  $\theta$  analytically).

## RESULTS

Let us continue with the 6-waypoint example and compare WN and DWN. In WN, navigation performance depends on the minimum turning radius and the waypoint radius. In DWN, navigation performance depends on minimum turning radius alone. The feedback control algorithms are the same for WN and DWN. However, the resulting overshoots differ. An illustrative feedback control algorithm described below accounts for turning toward an endpoint limited by the vehicle's minimum turning radius  $R$  and serves as a basis for the comparison between WN and DWN.



**Figure 8.** Distributions of different turning conditions.

## An Illustrative Feedback Control Algorithm

Consider Figure 9 showing a reference path with endpoints that are waypoints in the case of WN and that are turning points in the case of DWN. A vehicle at time  $t_i$  heads in a direction  $\phi_{Ti}$  (not shown) a distance  $s_i$  from the reference path. The vehicle travels a distance  $v\Delta t$  over a time step. At time  $t_{i+1}$ , the vehicle heads in a direction  $\phi_{i+1} = \phi_i + \psi_i$  in which  $\psi_i$  is the turn angle over one iteration, given by

$$\psi_i = \begin{cases} \psi_{\max} & \phi_{Ti} + \phi_{Ri} > \psi_{\max} \\ -\psi_{\max} & \phi_{Ti} + \phi_{Ri} < -\psi_{\max} \\ \phi_{Ti} + \phi_{Ri} & |\phi_{Ti} + \phi_{Ri}| \leq \psi_{\max} \end{cases} \quad (5)$$

$$\phi_{Ri} = |gs_i + h\dot{s}_i| \frac{\phi_{Ti}}{|\phi_{Ti}|} \quad \psi_{\max} = \frac{v\Delta t}{R}$$

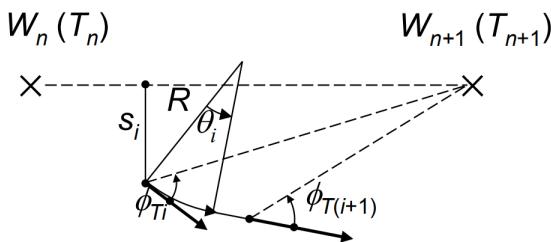
Above,  $\phi_{Ti}$  is a target angle and  $\phi_{Ri}$  is a reference angle (all of the angles are positive counter-clockwise),  $g$  and  $h$  are control gains, and  $\phi_{Ti}/|\phi_{Ti}|$  determines whether  $\phi_{Ri}$  is counter-clockwise or clockwise.

The parameters used in the results, both with and without wind disturbances,

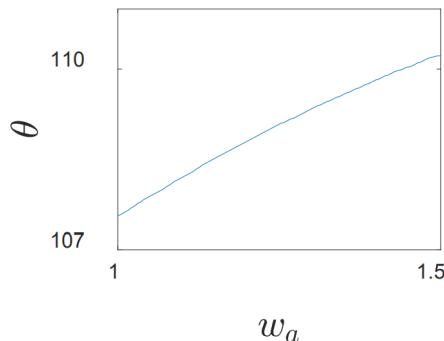
are as follows: The minimum turning radius was  $R = 0.7$ , the time step was 0.02, the vehicle speed was  $v = 0.8$ , and the control gains were  $g = 0.1$  and  $h = 5.25$ . With these parameters the turning radius was 0.7.

## The Shortest Path Approximation

Let us compare the minimum fuel and the shortest path solutions, the latter determined when  $w_a = w_s = 1$ . In practice, the distance between waypoints is at least four times larger than the turning radius  $R$ . Therefore, we will only consider distances  $A$  and  $B$  that are more than or equal to 4 times greater than  $R$ . When the distances  $A$  and  $B$  are exactly 4 times greater than  $R$ , the vehicle is flying in a straight line the least amount of the time and, for this case, we expect the difference between the minimum fuel solution and the shortest path solution to be the greatest. Figure 10 shows minimum fuel solutions in which  $w_a$  varies from 1—the shortest path case—to  $w_a = 1.5$ , for which the fuel cost during turning flight is 50% greater than the fuel cost during straight flight (In small-class fixed-wing vehicles the fuel penalty of turning is less than 50%). As shown, the orientation angle of the turning circle does not change more than 4°. The minimum-fuel path is very close to the shortest path and therefore the shortest path approximates the minimum fuel path. The use of the shortest path would therefore be acceptable in the vast majority of small-class UAV problems. This finding is significant because the shortest path solutions apply to any vehicle whereas the minimum fuel solutions need to consider vehicle parameters. In the results below, we employ the shortest path solutions.



**Figure 9.** Parameters for the controller.



**Figure 10.** Example of a CCW-CCW condition ( $A = B = 4$ ,  $\alpha = -45^\circ$ ,  $\beta = 30^\circ$ , CCW-CCW).

## Navigation in the Absence of an Unknown Wind Disturbance

Figure 11 compares WN and DWN in the absence of an unknown wind disturbance. Indeed, when the wind disturbance is completely accommodated for (by subtracting it out), DWN and WN exhibit the performance shown in Figure 11. The minimum turning radius  $R$  of the vehicle is the same in each case. The three cases consider WN waypoint radii of  $0.1R$ ,  $0.5R$ , and  $R$ , respectively.

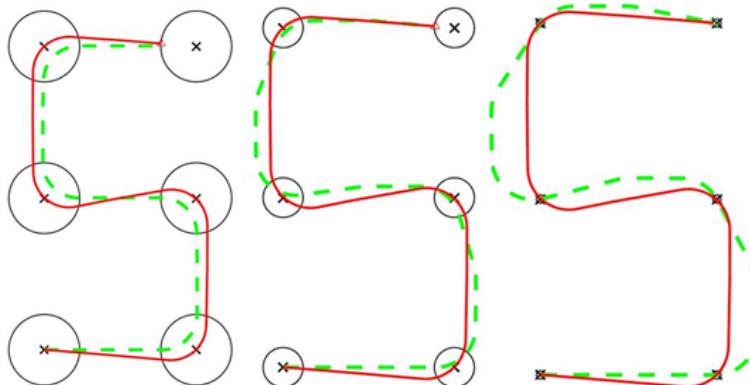
First consider DWN (solid lines). In each case, the vehicle passes through waypoints exactly and follows the minimum fuel paths—by design. Next, consider WN. In the first case, the vehicle passes close to the waypoints but with significant overshoot. In the other two cases, we see as the waypoint radius increases that the vehicle passes farther from the waypoints. In the third case, the vehicle undershoots the desired path. Note that in the WN and DWN cases given above, the feedback control had very little effect on the response because there was no unknown wind disturbance.

## Navigation in the Presence of an Unknown Wind Disturbance

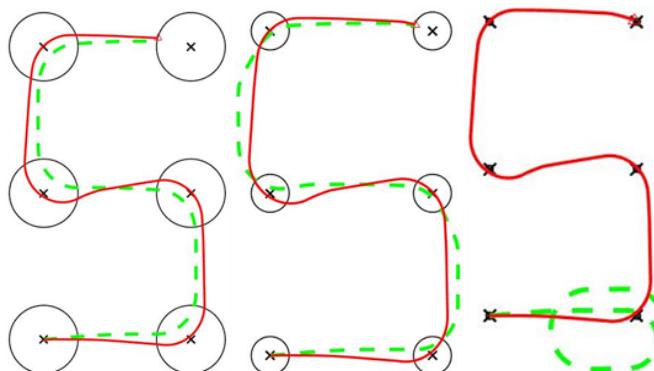
Figure 12 shows WN and DWN responses in the presence of a wind disturbance. The wind velocity is constant and unknown, set to 10% of the drone's speed directed from the bottom to the top of the figures (in the horizontal plane) and not accommodated for.

In the left figure, the waypoint radius was set to be small, intending to achieve higher accuracy. However, the wind exceeds the capability of the feedback controller and the vehicle just misses the second waypoint

sending it into a swirl. DWN also misses the turning point paired with the second waypoint but moves on to complete its journey. It can do this because it recognizes the waypoint to be unreachable. In the middle and the right figures, the waypoint radii are larger and WN manages to navigate the vehicles. It is worth noticing that errors exist for both WN and DWN. The DWN errors result from using the turning points for navigation instead of waypoints. It is clear that the DWN errors are smaller than the WN errors due to its features already discussed, hence one still recognizes that DWN improves the performance of the WN navigation.



**Figure 11.** WN and DWN ( $R_{wp} = 0.1R$ ,  $0.5R$ , and  $R$ ).



**Figure 12.** WN with and without the DWN in the presence of a disturbance ( $R_{wp} = 0.1R$ ,  $0.5R$ , and  $R$ ).

## CONCLUSION

WN suffers from an existing trade-off between minimum turning radius

and waypoint radius that prevents vehicles from reaching waypoints and closely following desired paths. This paper developed an algorithm, called Dubins waypoint navigation (DWN) that remedies this problem by not using waypoint circles. Instead, we introduced a reachability zone and turning points. The DWN significantly reduces undershoot, overshoot, and swirl. Furthermore, the algorithm remedies the problem in a way that can be hidden from the operator to avoid confusion. The paper showed, by establishing a horizon that includes two future waypoints, how to improve the performance of WN in terms of the targeting of waypoints while reducing fuel and time. Pertaining to real-time implementation, we also reduced the computational effort, essentially eliminating it, by parameterizing the shortest path solution. We also compared WN and DWN tracking performance (in the absence of an unknown disturbance) and regulation performance (in the presence of an unknown disturbance). The comparisons are illustrative of the improvements in path following (tracking and regulation) obtained by DWN.

## ACKNOWLEDGEMENTS

The authors gratefully recognize the Namibia Wildlife Aerial Observatory (WAO) for its support of this work.

## NOMENCLATURE

A = distance between current position and the following waypoint

a = coefficient for curve fitting  $\beta$  with respect to  $\delta$

B = distance between the following two successive waypoints

b = coefficient for curve fitting a with respect to A and B

C = current position

$F_u$  = total fuel cost

g = proportional term for PID controller

h = derivative term for PID controller

$L_a$  = total length of the arcs

$L_s$  = total length of the straight segments

R = turning radius of the vehicle

T = target point

v = speed of the vehicle

W = waypoint

$w_a$  = fuel cost per unit length travelled on an arc

$w_s$  = fuel cost per unit length travelled on a straight line

$\alpha$  = angle of heading in local system

$\beta$  = angle between the following two successive waypoints in local system

$\gamma$  = angular position of the first tangential point in local system

$\delta$  = angular position of the last tangential point in local system

$\theta$  = angular position of the center of the turning circle

$\phi_i$  = angle that denotes the heading in global frame at the  $i$ th time instance

$\phi_R$  = angle of turning resulted from the feedback controller

$\phi_T$  = angle between heading and the target

$\psi$  = angle of turn in one time step

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest regarding the publication of this paper.

## REFERENCES

1. Drubin, C. (2013) UAV Market Worth \$8.3 B by 2018. *Microwave Journal*, 56, 37.
2. Tsourdos, A., White, B. and Shanmugavel, M. (2011) Path Planning in Two Dimensions. In: Tsourdos, A., White, B. and Shanmugavel, M., Eds., *Cooperative Planning of Unmanned Aerial Vehicles*, Wiley, Chichester, 30. <https://doi.org/10.1002/9780470974636>
3. Ariff, O. and Go, T. (2011) Waypoint Navigation of Small-Scale UAV Incorporating Dynamic Soaring. *The Journal of Navigation*, 64, 29-44. <https://doi.org/10.1017/S0373463310000378>
4. Goerzen, C., Kong, Z. and Mettler, B. (2010) A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57, 65-100. <https://doi.org/10.1007/s10846-009-9383-1>
5. Manyam, S., Rathinam, S., Casbeer, D. and Garcia, E. (2017) Tightly Bounding the Shortest Dubins Paths through a Sequence of Points. *Journal of Intelligent & Robotic Systems*, 88, 495-511. <https://doi.org/10.1007/s10846-016-0459-4>
6. Wolek, A. and Woolsey, C. (2015) Feasible Dubins Paths in Presence of Unknown, Unsteady 5 Velocity Disturbances. *Journal of Guidance Control and Dynamics*, 38, 782-786. <https://doi.org/10.2514/1.G000629>
7. McGee, T. and Hedrick, J. (2007) Optimal Path Planning with a Kinematic Airplane Model. *Journal of Guidance, Control, and Dynamics*, 30, 629-633. <https://doi.org/10.2514/1.25042>
8. Milutinovic, D., Casbeerm, D., Cao, Y. and Kingston, D. (2017) Coordinate Frame Free Dubins Vehicle Circumnavigation Using Only Range-Based Measurements. *International Journal of Robust and Nonlinear Control*, 27, 2937-2960. <https://doi.org/10.1002/rnc.3718>
9. Ketema, Y. and Zhao, Y. (2010) Micro Air Vehicle Trajectory Planning in Winds. *Journal of Aircraft*, 47, 1460-1463. <https://doi.org/10.2514/1.C000247>
10. Meyer, Y., Isaiah, P. and Shima, T. (2015) On Dubins Paths to Intercept a Moving Target. *Automatica*, 53, 256-263. <https://doi.org/10.1016/j.automatica.2014.12.039>
11. Wang, Z., Lium L., Long, T. and Xu, G. (2018) Efficient Unmanned Aerial Vehicle Formation Rendezvous Trajectory Planningf Using

Dubins Path and Sequential Convex Programming. *Engineering Optimization*, 51, 1412-1429. <https://doi.org/10.1080/0305215X.2018.1524461>

12. Medeiros, A. and Urrutia, S. (2010) Discrete Optimization Methods to Determine Trajectories for Dubins' Vehicles. *Electronic Notes in Discrete Mathematics*, 36, 17-24. <https://doi.org/10.1016/j.endm.2010.05.003>
13. Silverberg, L. and Xu, D. (2018) Minimum-Fuel Hidden Layer Heuristic for SmallClass UAV. Master of Science Thesis, Mechanical Engineering, North Carolina State University, Raleigh.



# **CHAPTER**

# **11**

## **Modelling Oil-Spill Detection with Swarm Drones**

---

**F. Aznar,<sup>1</sup> M. Sempere,<sup>1</sup> M. Pujol,<sup>1</sup> R. Rizo,<sup>1</sup> and M. J. Pujol<sup>2</sup>**

<sup>1</sup>Department of Computer Science and Artificial Intelligence, University of Alicante, 03090 Alicante, Spain

<sup>2</sup>Department of Applied Mathematics, University of Alicante, 03090 Alicante, Spain

### **ABSTRACT**

Nowadays, swarm robotics research is having a great increase due to the benefits derived from its use, such as robustness, parallelism, and flexibility. Unlike distributed robotic systems, swarm robotics emphasizes a large number of robots, and promotes scalability. Among the multiple applications of such

---

**Citation:** Chen-Charpentier, Benito, Aznar, F., Sempere, M., Pujol, M., Rizo, R., “Modelling Oil-Spill Detection with Swarm Drones”, Hindawi Journal on Analysis and Models in Interdisciplinary Mathematics, Volume 2014, Article ID 949407, 14 pages, <https://doi.org/10.1155/2014/949407>.

**Copyright:** © 2014 F. Aznar et al. This is an open access article distributed under the Creative Commons Attribution License.

systems we could find are exploring unstructured environments, resource monitoring, or distributed sensing. Two of these applications, monitoring, and perimeter/area detection of a given resource, have several ecological uses. One of them is the detection and monitoring of pollutants to delimit their perimeter and area accurately. Maritime activity has been increasing gradually in recent years. Many ships carry products such as oil that can adversely affect the environment. Such products can produce high levels of pollution in case of being spilled into sea. In this paper we will present a distributed system which monitors, covers, and surrounds a resource by using a swarm of homogeneous low cost drones. These drones only use their local sensory information and do not require any direct communication between them. Taking into account the properties of this kind of oil spills we will present a microscopic model for a swarm of drones, capable of monitoring these spills properly. Furthermore, we will analyse the proper macroscopic operation of the swarm. The analytical and experimental results presented here show the proper evolution of our system.

## INTRODUCTION

Maritime activity has been increasing gradually in recent years. For example, around 42000 vessels (excluding the fishing ones) pass throughout the North Sea, carrying products that can adversely affect the environment, such as oil, which can produce high levels of pollution in case of being spilled into sea. Moreover, many pollutants are accidentally spilled from ships during “normal” operations. These spills are probably small but become significant due to the large number of ships. Dramatic incidences of marine pollution, such as the Prestige oil spill off the Spanish north coast [1–3], have highlighted the potential for human-caused environmental damage. In attempting to mitigate or avoid future damage to valuable natural resources caused by marine pollution, research has been undertaken by the scientific community to study the processes affecting the fate and distribution of marine pollution and especially to model and simulate these processes. Furthermore, active systems, able to detect and track such kind of spills, are an invaluable tool to help to locate and clean the affected resources [4–6].

Moreover, swarm robotics is an approach to solve problems inspired by the collective behaviour of social animals and it is focused on the interaction of multiple robots. It is a different approach to classical artificial intelligence, where the main goal is usually to develop behaviours that mimic human brain function. Swarm robotics is based on the metaphor of social insects and

emphasizes aspects like decentralized control, limited communication between agents, local information, emergence of a global behaviour, and robustness [7–10]. As it has been shown [10] it is different from other multirobotic systems, because, for example, the swarm robots are relatively simple.

However, the distributed nature of these systems makes it difficult to develop an architecture that correctly models a swarm and coordinates it to perform complex tasks. In order to design a swarm behaviour, a mathematical model must be provided for both the individual agents and the whole swarm [11, 12]. These models will be tested to evaluate the performance of the swarm before its deployment in real UAVs. This is mainly because this kind of systems has emergent properties that make it difficult to predict the overall operation of the swarm when only the local behaviour of the agents is being analysed.

In this paper, we will present a swarm behaviour to be used with robotic drones in order to cover the perimeter or the area of an oil spill. We will use GNOME [13], a very realistic environmental simulation system for marine pollutants. With the data provided by this simulation, using real maritime information and air currents predictions, we will create a virtual oil spill in the Spanish coast. We will show that our behaviour is able to both detect and track the spill for a given period of time. Moreover, we will also show, through the use of the macroscopic model derived by the local rules of the robots, that the swarm will track the oil pollutants. A Fokker-Plank equation will be derived from the microscopic behaviour in order to predict the probability that a drone stays in a given position at time  $t$ .

## POLLUTANT DISPERSION MODEL

Whether the trajectory model is accurate, adequate, or correct is often questioned. The accuracy and adequacy of a model are associated with the data used for the calculations and modelled physical processes. The modelling of an oil spill is not a simple task, mainly due to the number of factors influencing the trajectories of pollutants: sea currents, winds, and even the gravitational force or surface tension of water. There are several applications that model and simulate pollutant discharges into the sea. These applications are based on the modelling of the most important forces that interact on an oil slick. In this paper, one of these applications has been used to obtain realistic data of an oil spill off Spanish coast. As it will be seen in subsequent sections, we use both ocean currents and wind data. These data will be used for testing a realistic oil spill using a swarm of flying robots (drones).

More specifically, we have used GNOME, an interactive environmental simulation system designed for the rapid modelling of pollutant trajectories in the marine environment. GNOME simulates oil movement due to winds, currents, tides, and spreading. It was developed by the Hazardous Materials Response Division (HAZMAT) of the National Oceanic and Atmospheric Administration Office of Response and Restoration (NOAA OR&R). HAZMAT uses GNOME during spill events to calculate the spill trajectory and the associated uncertainty in that trajectory. As it has been mentioned in [14], GNOME can be used for predicting how winds, currents, and other processes might move and spread oil on the water, for studying how predicted oil trajectories are affected by inexactness (uncertainty) in current and wind observations and forecasts, and for predicting the chemical and physical changes of the oil spill during the time that it remains on the water surface.

One of the benefits of this tool is the possibility of using external files containing information on tides, currents, and so forth. If this information is added, then GNOME uses it to make the trajectory prediction of the spill (even using uncertainty during simulation) in the specified region. The output from the model consists of graphics, movies, and data files for postprocessing in a geographic information system (GIS) or NOAA Emergency Response Division's (ERD1) in-house model GNOME Analyst.

The model used by this tool is general and applies to trajectory problems. It is an Eulerian/Lagrangian model that is two-dimensional (2-D) in space, but vertically isolated layered systems can be modelled. Shoreline maps are inputs to the model, so any area can be modelled. The model automatically handles either hemisphere or east or west longitude.

More specifically, random spreading (diffusion) is calculated by a simple random walk with a square unit probability. The random walk is based on the diffusion value,  $D$ , that represents the horizontal turbulent diffusion of the spill in the water. During a spill, the value is calibrated as based on overflight data. In this way diffusion and spreading are treated as stochastic processes. Gravitational and surface tension effects are ignored, as these are only important during the first moments of a spill. Complex representation of subgrid diffusion and spreading effects are ignored.

The main diffusion equation used in GNOME is presented in [13], where  $Dx$  and  $Dy$  are the scalar diffusion coefficients in  $x$  and  $y$  directions and  $C$  is the pollutant concentration:

$$\frac{\partial C}{\partial t} = D_x \cdot \frac{\partial^2 C}{\partial x^2} + D_y \cdot \frac{\partial^2 C}{\partial y^2}. \quad (1)$$

GNOME simulates this diffusion with a random walk with any distribution, with the resulting diffusion coefficient being one-half the variance of the distribution of each step divided by the time step:

$$D_x = \frac{1}{2} \cdot \frac{\sigma^2}{\Delta t}, \quad (2)$$

where  $\sigma^2$  is the variance of the distribution of diffused points and  $\Delta t$  is the time elapsed between time steps.

Evaporation in GNOME is modelled by a simplistic 3-phase evaporation algorithm where the pollutant is treated as a three-component substance with independent half-lives [15]:

$$\begin{aligned} X_{\text{prob}} = & \left( P_1 \cdot \left( 2^{-t_i/H_1} - 2^{(t_{i-1}-2t_i)/H_1} \right) \right. \\ & + P_2 \cdot \left( 2^{-t_i/H_2} - 2^{(t_{i-1}-2t_i)/H_2} \right) \\ & \left. + P_3 \cdot \left( 2^{-t_i/H_3} - 2^{(t_{i-1}-2t_i)/H_3} \right) \right) \\ & \times \left( P_1 \cdot 2^{-t_i/H_1} + P_2 \cdot 2^{-t_i/H_2} + P_3 \cdot 2^{-t_i/H_3} \right)^{-1}, \end{aligned} \quad (3)$$

where  $t_i$  and  $t_{i-1}$  are the time elapsed (age; in hours) at time step  $i$  and the previous time step  $i-1$ , respectively, since the LEs release.  $H_1$ ,  $H_2$ , and  $H_3$  are the half-lives of each constituent (in hours) for the pollutant, and  $P_1$ ,  $P_2$ , and  $P_3$  are the percentages of each constituent (as decimals) for the pollutant.

Spilled substances are modelled as point masses (up to 10,000<sup>4</sup>) called LEs (Lagrangian elements) or “splots” (from “spill dots”). Spills can be initialized as one-time or continuous releases and as point or line sources or evenly spaced in a grid on the map for diagnostic purposes.

Once GNOME executes a simulation, the solution is produced in the form of a trajectory. GNOME provides two solutions to an oil spill scenario, the best estimate trajectory and the uncertainty trajectory. The best estimate solution shows the model result with all of the input data assumed to be correct. The uncertainty solution allows the model to predict other possible trajectories that are less likely to occur, but which may have higher associated risks. In this paper we will use the uncertainty solution of pollutant particles (represented by its LEs) for generating a continuous pollutant map. More details of this mathematical model could be obtained in [13].

## SWARM DESIGN

In this section we will analyse the features that must have a robotic swarm to detect and follow an oil spill in effective way. We will also propose a specific microscopic behaviour for this task. As mentioned above, the modelling of this kind of pollutants is a complex task because of the interaction of many factors. However, it is easy to extract a set of characteristics that a system needs to locate this kind of spills.

On the one hand, the swarm must be able to detect and follow pollutants that can change and move in time mainly by the action of advection and diffusion. Depending on the application, two situations can be found: the origin of the spill is known or the swarm initially makes the detection process (in this work it is assumed that detection is one of the tasks to be performed). On the other hand, the appearance of several polluting slicks is very probable due, among other factors, to the transport and deposit of sediments on the coast while the oil slick disperses and evaporates.

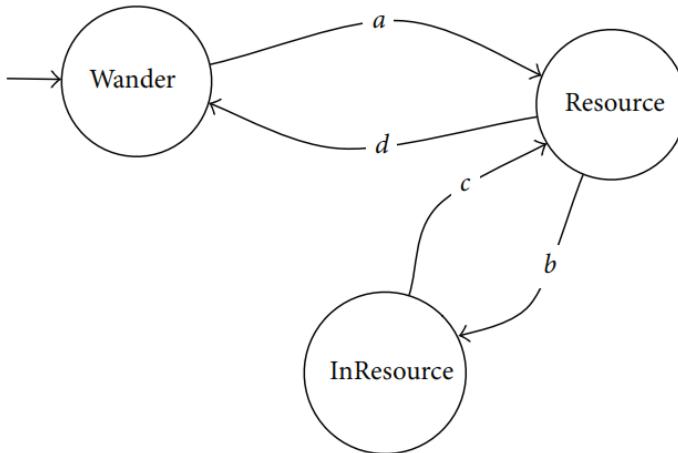
The behaviour of the swarm must be highly robust and tolerant to failures and should be totally distributed. Therefore, all agents must have the same importance when performing the behaviour, without the existence of any agent more prominent than another. Finally, the behaviour should be highly scalable for two reasons: robust issues and performance of the behaviour, since as a first step it may be beneficial to use a reduced number of agents until they find any evidence of a particular discharge.

Although in this paper behaviours will be analysed in a simulated way, the features of agents are directly based on flying robots (our intention is to use these algorithms in drones). These drones will have a video camera that will use a vision algorithm to determine the existence of any oil slick in its visual field (as presented in [4, 16]). For security reasons, we assume that drones will fly at different heights, so that collisions in a position ( $x$ ,  $y$ ) are avoided. We also assume that due to flight altitude (about 500 m above sea level) the differences in the visual field caused by different flying altitudes are negligible. All tests, unless otherwise specified, will be carried out by a swarm of 200 drones. It is a medium-sized swarm, appropriate for simulation.

Our main goal, once the behaviour is designed, is to determine the ability of the swarm to locate, converge, and follow an oil spill. Therefore, a macroscopic model to predict the global behaviour of the swarm and to verify its performance will be specified. More specifically, we propose a homogeneous behaviour, executed by all agents, consisting of three states.

Initially, drones look for any trace of the spill in the environment. Once the spill is detected, the drone will head to it. Finally, the drone will try to stay inside it (to cover the slick) or in its perimeter (to mark it).

Figure 1 shows the finite state machine (FSM) that governs the behaviour of each robot. The initial state is *Wander*, since we initially assume that the position of the spill is unknown. The transition from *Wander* state is performed when the agent's visual sensor detects an oil slick. In this case, the new state will be *Resource*. This state (*Resource*) has two possible transitions: the spill is not detected (transition *d*) so the system returns to *Wander* state or the amount of oil detected is  $>80\%$  of the image (transition *b*), so, the system switches to *InResource* state. The agent remains in this state if the detected oil percentage is  $>80\%$ ; otherwise, the system returns to *Resource* state.



**Figure 1:** Finite state machine (FSM) that governs the operation of each agent. The initial state is *Wander*. The transition *a* is triggered when the agent's visual sensor detects an oil slick. Transition *b* occurs when the amount of oil detected is  $>80\%$  of the image. Transition *c* is triggered when the amount of oil is  $\leq 80\%$  of the image. Finally, transition *d* is triggered if no oil is detected.

Next, each behaviour specified in the FSM will be described in more detail.

At the beginning of the execution agents start at *Wander* state. At that time they have no idea where to find an oil spill. Therefore, all directions are equally valid and they will move randomly. The velocity of the drone at time *t* is defined as

$$\mathbf{v}_w(t) = \mathbf{v}_w(t-1) + \mathbf{rand} \cdot \mu_1, \quad (4)$$

where rand is a random uniform vector defined within the interval of the maximum and minimum velocity of the drone, and  $\mu_1$  is the coefficient of variability on the current velocity. With values close to 1, the robot will move in a totally random way.

If a drone detects the resource, it heads to the resource to position itself at the perimeter or over it, depending on the desired behaviour. The velocity of the agent is defined by three factors:

$$\mathbf{v} = \alpha_1 \times \mathbf{v}_c + \alpha_2 \times \mathbf{v}_o + \alpha_3 \times \mathbf{v}_r, \quad (5)$$

where  $\alpha_1 + \alpha_2 + \alpha_3 = 1$  and these values define the intensity of each factor.

More specifically,  $\mathbf{v}_c$  specifies the direction of the robot. This direction is determined by the area with the larger intensity resource average:

$$\mathbf{v}_c = \gamma(S) \times \left\| \sum_{s \in S} ((\mathbf{pos}(s) - \mathbf{pos}(\text{rob})) \cdot s) \right\|, \quad (6)$$

where  $S$  is sensor readings that detect the resource at a given time,  $\mathbf{pos}(s)$  is the position vector of a reading, and  $\mathbf{pos}(\text{rob})$  is the position of the robot. We assume that the intensity of the resource is in the range  $[0, 1]$ , where 0 is the complete lack of resource and 1 is the unambiguous detection of it.

$\gamma$  determines the direction of the velocity vector, depending on whether the robot is outside or inside de resource. The aim of this variable is, therefore, to keep the robot on the perimeter of the resource:

$$\gamma(S) = \begin{cases} 1 & \frac{\sum_{s \in S} s}{|S|} \leq \eta \\ -1 & \frac{\sum_{s \in S} s}{|S|} > \eta, \end{cases} \quad (7)$$

where  $\eta$  is a threshold that determines, from the quantity of resource detected (0 means no resource and 1 maximum quantity of resource), if the agent is located on the perimeter. If the main objective of the system is that drones cover the pollute slick, then  $(S)$  will be defined as 1 for any set of readings  $S$  at a given time.

$\mathbf{v}_o$  specifies an avoidance vector with respect to all robots detected at a given moment:

$$\mathbf{v}_o = \left\| \sum_{i=1}^{|R|} (\mathbf{pos}(r_i) - \mathbf{pos}(\text{rob})) \right\|, \quad (8)$$

where  $R$  is the set of detected robots,  $\mathbf{pos}(r_i)$  is the position of the detected robot  $i$ , and  $\mathbf{pos}(\text{rob})$  is the position of the current robot.

Moreover, we will take into account the accuracy of the transmitted locations: there are several factors that could make these locations not to be optimal. We will include, therefore, a random component to model this uncertainty in the movement of the robot:  $\mathbf{kr}(t) = \mathbf{kr}(t-1) + \mathbf{rand} \cdot \mu_2$ , where  $\mu_2$  is the coefficient of variability on the velocity

Finally, when drones are located inside the spill and, therefore, the borders of the resource are not detected, we assume that drones will develop a random wander behaviour until they find water again because there is no other information about which direction is better to follow:

$$\mathbf{v}_s(t) = \mathbf{v}_s(t-1) + \mathbf{rand} \cdot \mu_3, \quad (9)$$

where  $\mu_3$  is the coefficient of variability on the velocity.

In this section we have defined a microscopic behaviour for the detection and tracking of an oil spill. Each agent of the swarm executes this behaviour locally. It is a simple behaviour that can be run in low processing capacity systems. This behaviour does not require any communication between agents or a global positioning system.

## EXPERIMENTATION

We will now present a set of experiments in order to test the operation of the proposed microscopic behaviour. As we will see, these experiments use simulations of oil spills based on real data.

We have simulated an oil spill in the coastal area of “Sant Antoni de Portmany,” on Ibiza island, using an area of approximately  $350 \text{ km}^2$ . This area has been chosen because of the various currents that affect it and its proximity to the peninsula. Real weather data have been used for both sea and air currents, considering the start date of the spill on April 10, 2013, at 0:00 h. We use these real data to run a simulation for seven days from the start of the spill, simulating a ship loaded with 100000 barrels of oil. Figure 7 shows the map of the main area affected and the initial place of impact. In order to work with data generated by GNOME we use the average of the points of discharge generated by the application (More specifically,  $(t)' =$

$G(M(t) \oplus \text{sshape})$  where  $M(t)$  is the map obtained with GNOME at time  $t$ ,  $G$  is a Gaussian filter with  $\sigma = 10|S|$ , and  $\oplus$  is a morphological binary operator (splots in GNOME), making uniform the continuity between nearby contaminated areas and simplifying the simulation. This allows us to reduce the number of splots needed in the simulation:

$$\text{Robots} = 200$$

$$\mu_1 = 0.3$$

$$\mu_2 = 0.3$$

$$\mu_3 = 0.3$$

$$\eta = 0.8$$

$$\alpha_1 = 0.6$$

$$\alpha_2 = 0.2$$

$$\alpha_3 = 0.2. \quad (10)$$

The previous equation shows default parameters used in simulations for the microscopic model. These parameters have been adjusted taking into account the microscopic behaviour experiments.

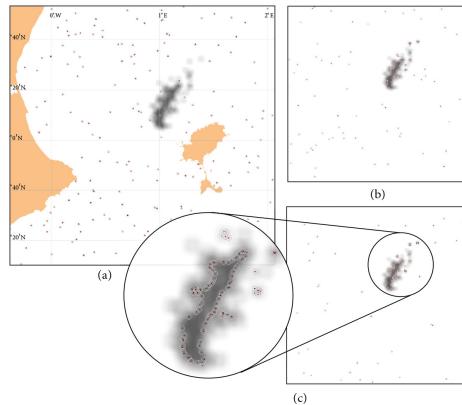
The simulation of the microscopic model has been developed using the software MASON [17]. We use a swarm of 200 agents randomly distributed by the environment that moves uniformly at 60 km/h. The simulation uses small size drones ( $< 3 \text{ m}^2$ ) that are able to visually detect an area of 1 km<sup>2</sup>.

Using these parameters, several tests will be performed to verify the correct operation of our model at local level. Initially, first tests check the convergence of the swarm for a single static slick. Using the same example, will be modified to cover the slick instead of marking its perimeter. Next tests will verify the convergence of the swarm choosing an instant with several active slicks. Finally, we will check the tracking of the movement of the spill.

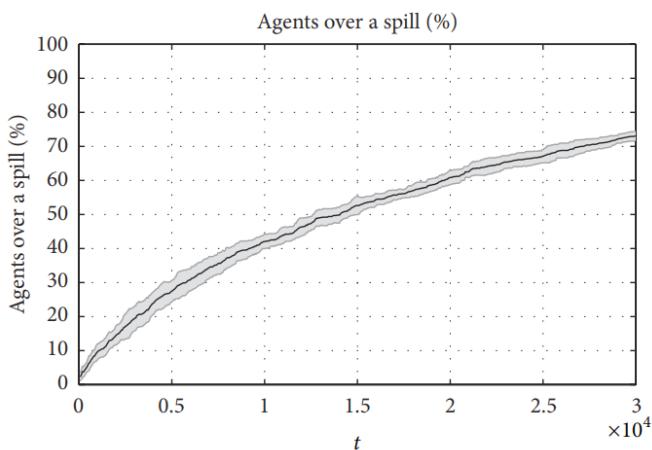
## Spill Detection

First tests have been developed to detect and monitor a static slick. On the one hand, the swarm will mark the perimeter of the slick. Figure 2 shows

the initial distribution of agents with their geographical location (a) and the position of agents at time  $t = 15000$  s and  $t = 30000$  s ((b) and (c)). At time  $t = 30000$  s the swarm completely surrounds the oil slick. Figure 3 presents the percentage of agents over an oil slick for ten independent simulations. As it can be appreciated, this percentage increases progressively.

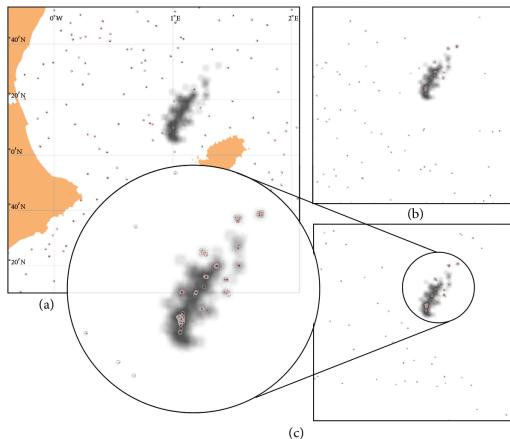


**Figure 2:** Distribution of agents with respect to time (in seconds) for the task of detecting the perimeter of an oil spill: (a) geographical location of the spill and initial position of agents, (b) distribution of agents at time  $t = 1500$  s, and (c) distribution of agents at time  $t = 30000$  s.



**Figure 3:** Percentage of agents on an oil slick with respect to time (in seconds). Ten different experiments have been carried out, showing the average and variance.

Moreover, as it has been discussed in the definition of the microscopic model of the swarm, when  $(S) = 1$  the swarm will cover the slick instead of marking its perimeter. Figure 4 shows the results of the swarm for this case, where 10 independent simulations have been carried out.



**Figure 4:** Distribution of agents for the task of covering the spill ( $(S) = 1$ ): (a) geographical location of the slick and initial position of agents, (b) position of agents at time  $t = 15000$  s, (c) position of agents at time  $t = 30000$  s.

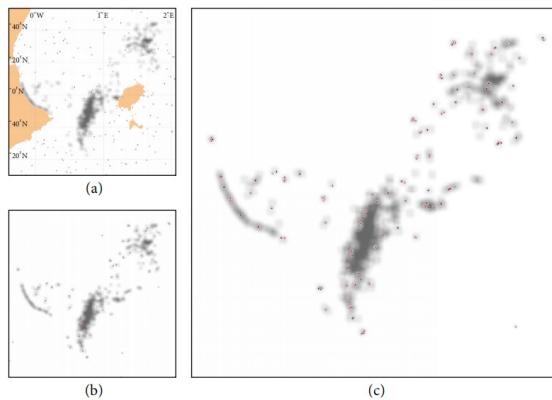
The experiments in Figure 2 show that the operation of the swarm is correct: 50% of agents are able to detect the spill in less than 4 hours since the beginning of the execution, marking correctly the perimeter of the spill. However, in a real environment the appearance of several slicks is common. This case will be analysed in the next section.

## Several Spills

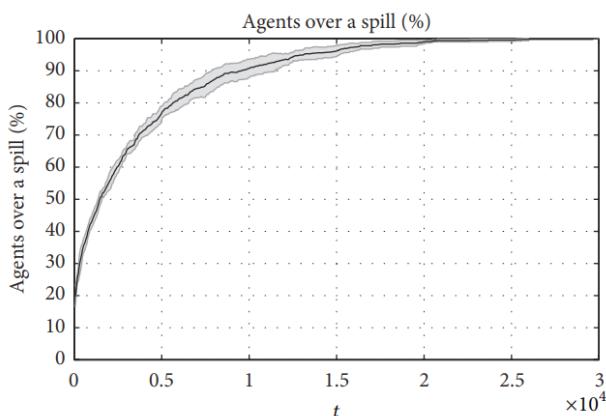
When an oil spill occurs, the oil may spread across kilometres generating several oil slicks. For this reason, it is very important to verify the correct operation of the swarm with different slicks that can spread out over a wide area. In the simulation, at early hours, several areas containing large proportions of pollutant are produced. For example, underwater currents can move oil slick close to the coast, while the ship continues spilling oil into the sea.

In order to check the operation of the microscopic model in these cases, we have chosen an instant of the simulation where the slick has a complex structure. Figure 5 shows the evolution of the distribution of robots. As it

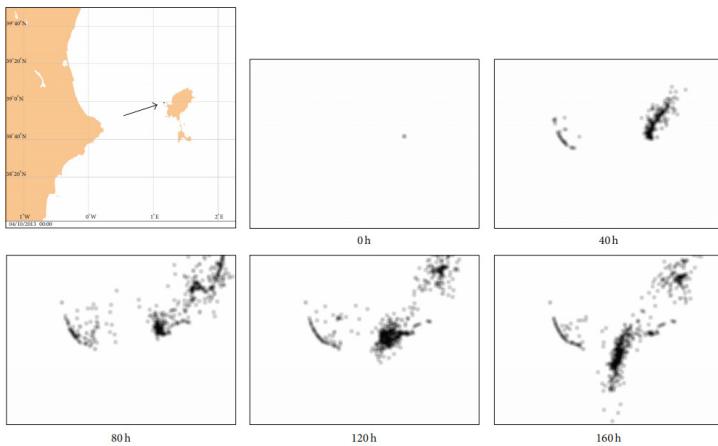
can be seen in this figure, even in this complex case and without previous data that indicates the spill trend, our model is able to locate and mark the perimeter of the slicks. More specifically, Figure 6 presents the percentage of agents that are located on a polluted area. In this figure it can be seen that the number of agents increases with time, therefore distributing uniformly on the slick.



**Figure 5:** Distribution of agents for the task of detecting the perimeter of an oil spill on a map with several slicks: (a) geographical location of the spill and initial position of agents, (b) distribution of agents at time  $t = 15000$  s, and (c) distribution of agents at time  $t = 30000$  s.



**Figure 6:** Percentage of agents on an oil slick with respect to time (in seconds) for a complex slick. Ten different experiments have been carried out, showing the average and variance.



**Figure 7:** Origin and temporal evolution of the spill. Several snapshots that show the position of the spill at different instants, measured in hours from the beginning of the spill, are shown.

## Spill Movement

Figure 5 tests have shown that the swarm is able to detect and mark the perimeter/area with complex slicks.

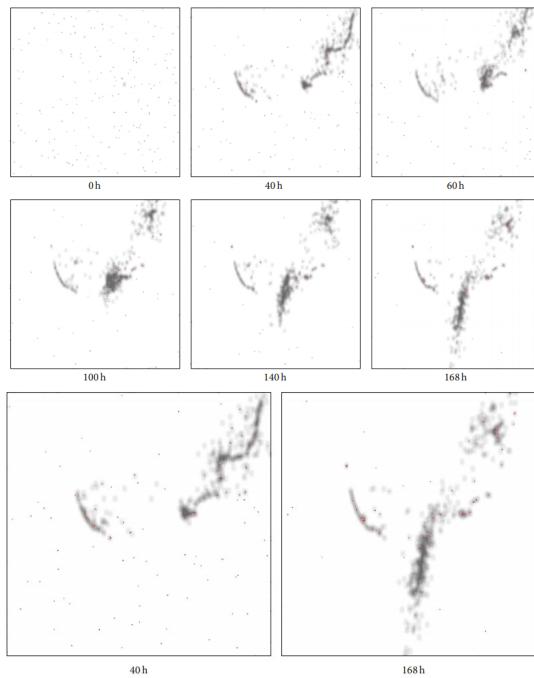
However, in a real spill, finding the location of the spill is as important as effective monitoring is. In this section we will analyse the behaviour of the swarm taking into account that the spill will spread as time progresses.

A simulation of 168 hours of the evolution of the spill has been performed, starting April 10, 2013, at 0:00 h.

In order to simplify the simulation, we capture an image of the state of the spill every 4.5 hours. Although the spill will experience changes in this period, the swarm can deal with these changes.

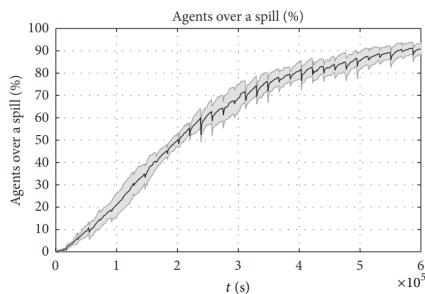
Figure 7 shows the origin point of the spill and some captures of its evolution.

As in previous experiments, the swarm is distributed randomly by the environment. In this case, in the simulation process, the location of the resource obtained by GNOME is updated every 4.5 hours. Figure 8 shows the evolution of the swarm with respect to the state of the oil spill.



**Figure 8:** Evolution of the swarm with respect to the oil slick. Multiple snapshots at different instants in time (measured in hours) are shown.

Figure 9 shows the percentage of drones that are on top of a resource at a given time. In order to produce this graph, 10 different simulations have been carried out. The graph shows the average and variance of these simulations. In this graph a series of steps produced artificially by the simulator (because of the sudden change from one state to the next) can be seen.



**Figure 9:** Percentage of agents over an oil slick with respect to time (in seconds). Ten different experiments have been carried out, showing the average and variance.

Nonetheless, the evolution of the swarm is correct. The swarm is able to track the spill and mark each of the slicks.

## MACROSCOPIC MODEL

Once the microscopic behaviour has been described, it is interesting to see the global behaviour of the swarm. There are several techniques used to analyse this behaviour [18] such as the use of recurrence equations, generated from a microscopic behaviour defined by a PFSM or the definition of differential equations. However, most of these methods allow only the analysis of the evolution of the state transitions globally.

In this work, we consider the framework proposed in [19] in order to obtain the probability distribution of the swarm position for any time  $t$ . This will enable us to predict, in great detail, the behaviour of the overall system. As described by [19], once the microscopic behaviour has been defined, the global behaviour of the system can be calculated using the Fokker-Planck equation:

$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t} = -\nabla (\mathbf{A}(\mathbf{r}, t) \rho(\mathbf{r}, t)) + \frac{1}{2} Q \nabla^2 (B^2(\mathbf{r}, t) \rho(\mathbf{r}, t)), \quad (11)$$

where  $Q$  is the displacement by a collision.  $\rho(\mathbf{r}, t) dr_x dr_y$  is the probability of encountering a robot at position  $r$  within the rectangle defined by  $dr_x$  and  $dr_y$  at time  $t$

This equation provides a method to statistically model a swarm of robots based on modelling techniques of multiparticle systems from the field of quantum physics. From a Langevin equation that represents the behaviour of a single particle, the Fokker-Planck equation is derived for all the system.

As we have already seen in [20], the Fokker-Planck equation implements the necessary abstraction of microscopic details as described above and treats rapidly changing parameters such as noise. The equation is still exact if this noise is generated by a Gaussian process, that is, if it is fully determined by the first two moments. It gives the temporal evolution of the probability density describing the positions of the agents.

Initially, the swarm designer must specify the functions  $A$  and  $B$  of (11), in accordance with the desired microscopic behaviour. Function  $A$  is a direction and describes the deterministic motion based on information provided by the environment and the information indirectly provided by other robots via the environment. Function  $B$  describes the random component of the motion.  $A$  and  $B$  are characterized by the underlying control algorithm.

$B$  typically incorporates influences by other robots that disturb the robot, for example, by the need of collision avoidance.  $A$  might incorporate an external influence such as a light gradient.

The most difficult task is the definition of these functions. This design is not unique and it requires finding two functions that correctly describe the behaviour of the swarm. These functions will be analytically presented and their performance will be tested in following sections.

A function determines the displacement of the swarm.  $A$  depends primarily on a vector representing the directional information. A potential field  $P$  is commonly used to define it. In our case, we need to establish a function that takes into account the following things based on the proposed microscopic model: the random motion states of the robot, the probability that a movement of an agent fails in its execution (e.g., due to a collision), and the potential field where the robots move. Although it is possible to model a probability distribution for each state, as our microscopic model has no interaction between agents (except purely physical, as collisions) and the behaviour of states is relatively simple, the macroscopic behaviour of the swarm can be comprised in a single distribution:

$$\begin{aligned} A(\mathbf{r}, t) = & (1 - b_r) \cdot (1 - \rho(\mathbf{r}, t))^{\mu_4} \\ & \cdot (\Gamma(P(\mathbf{r}, t)) \cdot K(\nabla P(\mathbf{r}, t))), \end{aligned} \quad (12)$$

where  $P$  is directly related to the sensor readings in point  $\mathbf{r}$  at time  $t$  and  $\mu_5$  is a normalization term:

$$P(\mathbf{r}, t) = \mu_5 \cdot s(\mathbf{r}, t). \quad (13)$$

In (12), a function  $A$  that takes into account previous aspects is proposed. When the density of agents increases, the probability of collision also increases and, therefore, this situation reduces the rate at which robots are directed by vector specified by  $A$ . More specifically, consider the following.

$b_r$  is the probability that comprises the influences of all states that develop a random or pseudorandom state.  $P(\mathbf{r}, t)$  as commented before, is defined as the probability of encountering a robot at position  $\mathbf{r}$  at time  $t$ .  $\Gamma$  is a function applied to the potential field that produces, for each position of  $P$ , the direction to be taken by drones, as a single agent would do using the function  $\gamma$ . In our case, we use a “slidingneighbourhood” filter as commented, for example, in [21] to perform the same calculation on  $P$  as on the microscopic model, changing the sign of the displacement when a ratio greater than 80% of pollutant is detected.  $K$  is a convolution operator. Being  $G$  a square gaussian

kernel of size  $d$ , generated with  $\sigma = 4 \cdot |S|$ , and assuming two-dimensional coordinates so that  $\mathbf{r} = (i, j)$ , the sum of the components of this kernel is  $\sum_{i=1}^d \sum_{j=1}^d (i, j)$ .  $F$  is defined as this sum if it is different from 0; otherwise, it will be defined as 1. In this way,

$$K_{i,j} = \frac{1}{F} \cdot \sum_{i=1}^d \sum_{j=1}^d G(i, j) \cdot \nabla P((i, j), t). \quad (14)$$

Function  $B$  describes the nondeterministic motion and, therefore, it takes into account the random motion of agents. Two forces, which must be considered, take part in the microscopic behaviour. On the one hand, some influences derived from agents that are on Wander and InResource states. These states have a random motion, depending on the intensity of parameters  $\mu_1, \mu_3$ . On the other hand, the behaviour itself causes that the environment has areas with a higher density of agents. In these areas the probability of collision can be increased depending on the density of agents at a given time:

$$B(\mathbf{r}, t) = b_r \cdot \rho(\mathbf{r}, t)^{\mu_4}. \quad (15)$$

Thus, in (15) two terms can be observed:  $b_r$  comprises the influences of all states that develop a random or pseudorandom state, as previously commented, and  $\rho(\mathbf{r}, t)^{\mu_4}$  is a term that defines the connection between the density of robots and their probability of collision.

## Spill Detection

The operation of the microscopic behaviour has been analysed in the previous section. It has been shown that the swarm is able to locate and mark the perimeter of an oil spill with relatively simple rules. Several tests have been developed to verify the validity of the presented model in various cases.

In addition to the tests above, it is possible to establish, for a given discharge, the areas of the map with the highest probability of containing a robot, independently of the number of agents (on condition that you use enough of them), by using the macroscopic model. This provides a more accurate visualization of the behaviour for large swarms without being limited by the number of agents to be simulated.

This section, using the previously presented macroscopic definition, presents how the swarm is able to locate and mark, in a given time, an oil spill:

Robots = 200

$$\mu_4 = 5$$

$$\mu_5 = 0.95$$

$$b_r = 0.5. \quad (16)$$

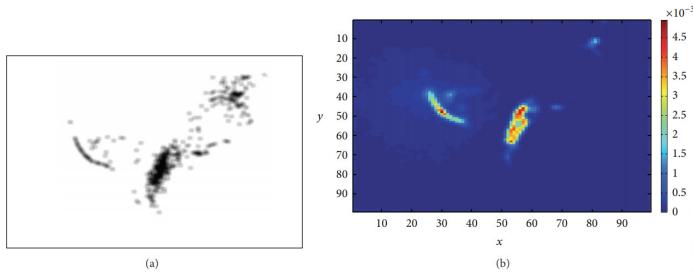
The previous equation shows default parameters used in simulations for the macroscopic model. These parameters have been adjusted taking into account the microscopic behaviour experiments. We will present the analysis of the macroscopic behaviour of the swarm for two time instants  $t = (140 \text{ h}, 168 \text{ h})$ .

The simulation process is simple, once the functions  $A$  and  $B$  of Fokker-Planck equation have been defined. Initially, an origin point for the swarm must be established. Although in microscopic simulations it is possible to establish different origin points (as setting randomly the position of agents), Fokker-Planck equation requires a single point. Nonetheless, several simulations for different origin points have been performed, observing that the results for large values of  $t$  are very similar: only slight variations can be seen if the origin point is located just on an oil slick. In this case, the probability of this slick in relation to the rest of the spill can be higher.

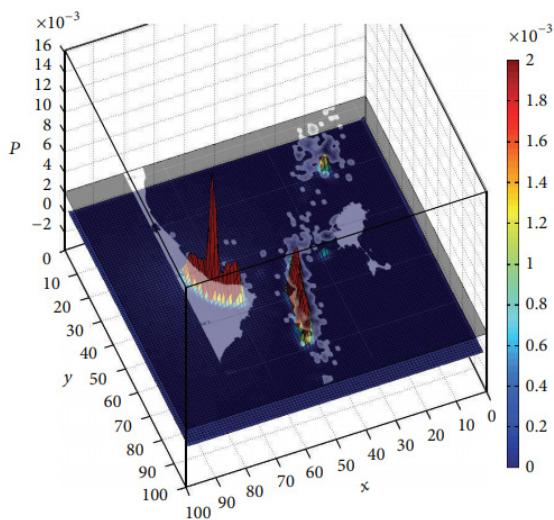
The same origin point has been used for all tests, discretizing the simulation area in  $100 \times 100$  units. The origin point is established in Fokker-Planck equation as  $(x, y) = (40, 25)$ .

Once the Fokker-Planck equation has been defined, the probability distribution that a certain agent is in a position of the environment at a given time can be obtained in an iterative way. This distribution can be calculated iterating for each instant of time  $t$  all the positions of the map, updating for each position the new probability described by the equation.

The macroscopic state of the swarm is presented in Figure 10 for  $t = 140 \text{ h}$ . A clear convergence in the perimeter of the spill can be observed. A three-dimensional representation of the probability distribution that an agent is in a certain position of the environment at  $t = 168 \text{ h}$  is presented in Figure 11. As it can be observed, the macroscopic model correctly predicts the behaviour presented at the microscopic level of the swarm.



**Figure 10:** (a) Map ( $M$ ) generated from GNOME data for  $t = 140$  h. (b) Probability that a robot is in a certain position of the space at  $t = 140$  h. Sampled using the macroscopic model of the swarm with the Fokker-Planck equation.

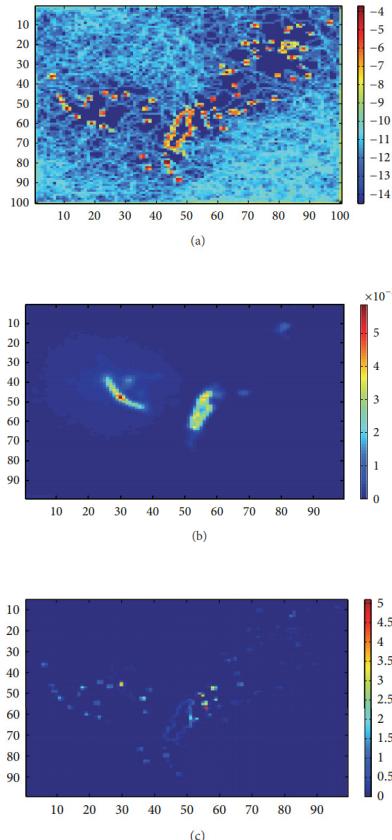


**Figure 11:** Three-dimensional representation of the probability that a robot is in a particular position at  $t = 168$  h. Sampled using the macroscopic model of the swarm with the Fokker-Planck equation. A map for the probability 0.002 where the environment and the state of the spill at  $t = 168$  h are represented is added.

## Model Comparison

In the previous section the behaviour of the macroscopic model and how this model predicts the overall behaviour of the swarm have been presented. Now, we will compare predictions of microscopic model and macroscopic model for a specific case.

Figure 12(a) shows a probability distribution obtained from the microscopic model. In order to do this, we have discretized the simulation environment and, with the same parameters used in previous sections, we have developed a simulation with 200 agents at  $t = 140$  h. Along the simulation, we save the number of agents that pass on a discrete cell (in order to compare it with the macroscopic model that we have also discretized in  $100 \times 100$  units) and then we calculate the probability that an agent is in this cell at a given time.



**Figure 12:** Comparison of the microscopic prediction and macroscopic model for the map  $M$  at  $t = 140$  h. (a) ( $\log(P_{\text{micro}})$ ) Logarithm of the probability distribution obtained simulating 200 agents for 30,000 seconds. Logarithmic distribution is used to highlight the states with low probability. (b) ( $P_{\text{macro}}$ ) Probability distribution obtained with Fokker-Planck equation. (c) Product of both distributions, decreasing the importance of high values in the microscopic simulation (specifically  $P_{\text{macro}} \times P_{\text{micro}}^{1/2}$ ).

In the same way, Figure 12(b) shows the probability distribution that predicts the macroscopic model. In this figure it can be seen that the area of interest is covered in both models. There are minor differences within the models, due to the deficiencies of microscopic simulation that, among other things, depends on the number of agents used in the simulation.

Nevertheless, we can compare both approaches by multiplying both distributions. In this way only high probabilities remain and, therefore, it is easier to observe if the areas of the spill are correctly identified in both models. Bearing this in mind, we have slightly rectified the microscopic distribution. When we use a limited number of agents in the simulation, we discover, in some cases, that high probabilities hide important information in the distribution.

In order to avoid this loss of information, we have used the square root of the distribution in order to compare both models.

The product of macroscopic distribution and the square root of microscopic distribution are presented in Figure 12(c). This figure shows how the most important parts of the spill are detected with both distributions, predicting the macroscopic model the same perimeter areas detected by the microscopic model.

## DISCUSSION

This paper describes a microscopic model that is able to locate and mark the perimeter of an oil spill. The microscopic behaviour presented does not require direct communication between agents. This limitation can cause that the convergence of the swarm on the spill takes more time, depending on the number of drones and the size of the spill.

However, this behaviour is versatile and easy to implement and develop, even in areas without GPS coverage. It is important to highlight that a swarm system, which requires direct communication between agents, is a limited system because of the maximum range of each agent and the saturation of the radiofrequency space if the system needs a large number of agents.

Moreover, we have demonstrated that the process of locating and marking the perimeter of the spill without communication is robust and efficient. We have shown that the swarm system is able to completely delimit the spill if the number of agents is sufficient. In order to achieve this task, an agent must be able to detect drones that are nearby. There are several ways, as, for example, using a camera or transmitting the GPS position.

We propose the use of signal intensity (at a given frequency) for obstacle avoidance tasks. This strategy may show some problems (we have implemented it by using a reactive behaviour); however, it has several advantages. Many countries require that drones broadcast a continuous signal indicating their position. Europe uses 433 MHz frequency for this purpose. The intensity of the signal in a particular area can be detected by using the same infrastructure. If the intensity of the signal grows with the movement of the agent, this agent must change its direction. We emphasize that, as a swarm approach, this is not a communication between agents but simply a *beacon* that we can use, if necessary, to know the position of drones.

The proposed macroscopic model demonstrates that the tendency of the swarm, for a sufficient number of drones, is the same that can be perceived in the microscopic model. The connection of both models has been tested for a complex spill, generated with GNOME. These experiments have shown that the fundamental characteristics of the behaviour (detection and monitoring) are reflected in both models. It is advisable not to forget the differences between the two models.

The microscopic model defines the individual behaviour. Because of this it is easy to understand at local level. However, this model does not define the behaviour of all the swarm. In order to analyse the global behaviour, a set of tests can be defined for a large number of agents. However, these tests can be expensive and difficult and are not exempt from problems.

The macroscopic model defines the global behaviour of the swarm. It allows us to verify the emergent behaviour from the interaction between all agents that run the microscopic model. The macroscopic model demonstrates the tendency of the swarm for a large number of agents. The analysis of this model is complex, because of the use of differential equations that, for example, force us to choose a single point to start the simulation. Even so, this model has remarkable advantages [20], for example, continuous analysis for any point of the environment, time of the probability that an agent is located in a given location, and simulation time negligible compared to microscopic model.

We are currently working on the implementation of this system in a real swarm of drones. Our immediate future research focuses on this real swarm, since it allows us to adjust the algorithms for a real system.

We are already in the testing phase for small swarms (5 drones), obtaining satisfactory results in our preliminary tests. We are using low-cost, custom-developed hexacopters to test this behaviour. The low computational needs

required by this behaviour make it possible to use cheap Arduino control boards to control the UAVs. We have developed several tests for our swarm behaviors using artificial color landmarks. While flying our agents can localize (by wandering) a landmark and follow it (if moving) in real time.

## **CONFLICT OF INTERESTS**

The authors declare that there is no conflict of interests regarding the publication of this paper.

## **ACKNOWLEDGMENT**

This work has been supported by the Spanish Ministerio de Ciencia e Innovación, Project TIN2009-10581.

## REFERENCES

1. J. I. Medina-Bellver, P. Marín, A. Delgado et al., “Evidence for in situ crude oil biodegradation after the prestige oil spill,” *Environmental Microbiology*, vol. 7, no. 6, pp. 773–779, 2005.
2. S. Díez, J. Sabaté, M. Viñas, J. M. Bayona, A. M. Solanas, and J. Albaigés, “The prestige oil spill. I. Biodegradation of a heavy fuel oil under simulated conditions,” *Environmental Toxicology and Chemistry*, vol. 24, no. 9, pp. 2203–2217, 2005.
3. J. J. González, L. Viñas, M. A. Franco et al., “Spatial and temporal distribution of dissolved/dispersed aromatic hydrocarbons in seawater in the area affected by the prestige oil spill,” *Marine Pollution Bulletin*, vol. 53, no. 5–7, pp. 250–259, 2006.
4. F. Nirchio, M. Sorgente, A. Giancaspro et al., “Automatic detection of oil spills from sar images,” *International Journal of Remote Sensing*, vol. 26, no. 6, pp. 1157–1174, 2005.
5. M. N. Jha, J. Levy, and Y. Gao, “Advances in remote sensing for oil spill disaster management: state-of-the-art sensors technology for oil spill surveillance,” *Sensors*, vol. 8, no. 1, pp. 236–255, 2008.
6. C. Brekke and A. H. S. Solberg, “Oil spill detection by satellite remote sensing,” *Remote Sensing of Environment*, vol. 95, no. 1, pp. 1–13, 2005.
7. M. Dorigo, E. Tuci, R. Groß et al., “The swarm-bot project,” in *Kunstliche Intelligenz*, pp. 31–44, Springer, Berlin, Germany, 2005.
8. M. Dorigo and E. Şahin, “Autonomous Robots: guest editorial,” *Autonomous Robots*, vol. 17, no. 2-3, pp. 111–113, 2004.
9. G. Beni, “From swarm intelligence to swarm robotics,” in *Swarm Robotics*, pp. 1–9, Springer, Berlin, Germany, 2005.
10. E. Şahin, “Swarm robotics: from sources of inspiration to domains of application,” in *Swarm Robotics*, vol. 3342 of *Lecture Notes in Computer Science*, pp. 10–20, 2005.
11. O. Soysal and E. Şahin, “A macroscopic model for self-organized aggregation in swarm robotic systems,” in *Swarm Robotics*, pp. 27–42, Springer, Berlin, Germany, 2007.
12. W. Liu, A. F. T. Winfield, and J. Sa, “Modelling swarm robotic systems: a case study in collective foraging,” in *Proceedings of the Towards Autonomous Robotic Systems (TAROS '07)*, pp. 25–32, 2007.

13. United States. National Ocean Service. Office of Response and Restoration, *General Noaa Operational Modeling Environment (Gnome) Technical Documentation*, U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, National Ocean Service, Office of Response & Restoration, 2012.
14. K. Wojtaszek, *Application of transport model for building contingency maps of oil spills on the North Sea [M.S. thesis]*, Delft University of Technology, 2003, A-1, [http://risk2.ewi.tudelft.nl/research-and-publications/doc\\_download/70-katarzynawojtaszekthesispdf](http://risk2.ewi.tudelft.nl/research-and-publications/doc_download/70-katarzynawojtaszekthesispdf).
15. P. D. Boehm, D. L. Flest, D. Mackay, and S. Paterson, “Physical-chemical weathering of petroleum hydrocarbons from the ixtoc I blowout: chemical measurements and a weathering model,” *Environmental Science and Technology*, vol. 16, no. 8, pp. 498–505, 1982.
16. K. Karantzalos and D. Argialas, “Automatic detection and tracking of oil spills in SAR imagery with level set segmentation,” *International Journal of Remote Sensing*, vol. 29, no. 21, pp. 6281–6296, 2008.
17. S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan, “Mason: a new multi-agent simulation toolkit,” in *Proceedings of the Swarm Fest Workshop*, vol. 8, 2004.
18. L. Bayindir and E. Şahin, “A review of studies in swarm robotics,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 2, pp. 115–147, 2007.
19. H. Hamann and H. Wörn, “A framework of space-time continuous models for algorithm design in swarm robotics,” *Swarm Intelligence*, vol. 2, no. 2–4, pp. 209–239, 2008.
20. H. Hamann, *Space-Time Continuous Models of Swarm Robotic Systems: Supporting Global-to-Local Programming*, vol. 9, Springer, Berlin, Germany, 2010.
21. J. L. Semmlow, *Biosignal and Medical Image Processing*, Signal Processing and Communications, Taylor & Francis, 2004.

# **CHAPTER**

# **12**

## **A Survey of Modelling and Identification of Quadrotor Robot**

---

**Xiaodong Zhang,<sup>1</sup> Xiaoli Li,<sup>2</sup> Kang Wang,<sup>2</sup> and Yanjun Lu<sup>1</sup>**

<sup>1</sup>School of Automation, Shenyang Aerospace University, Shenyang 110136, China

<sup>2</sup>School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China

### **ABSTRACT**

A quadrotor is a rotorcraft capable of hover, forward flight, and VTOL and is emerging as a fundamental research and application platform at present with flexibility, adaptability, and ease of construction. Since a quadrotor is basically considered an unstable system with the characteristics of dynamics

---

**Citation:** Xiaodong Zhang, Xiaoli Li, Kang Wang, and Yanjun Lu, “A Survey of Modelling and Identification of Quadrotor Robot”, Hindawi Journal on Abstract and Applied Analysis, Volume 2014 |Article ID 320526 | 16 pages | <https://doi.org/10.1155/2014/320526>.

**Copyright:** © 2014 Xiaodong Zhang et al. This is an open access article distributed under the Creative Commons Attribution License.

such as being intensively nonlinear, multivariable, strongly coupled, and underactuated, a precise and practical model is critical to control the vehicle which seems to be simple to operate. As a rotorcraft, the dynamics of a quadrotor is mainly dominated by the complicated aerodynamic effects of the rotors. This paper gives a tutorial of the platform configuration, methodology of modeling, comprehensive nonlinear model, the aerodynamic effects, and model identification for a quadrotor.

## INTRODUCTION

A quadrotor is agile to attain the full range of motion propelled by four rotors symmetrically across its center with smaller dimension and simple fabrication, unlike a conventional helicopter with complicated mechanism. Generally, it should be classified as a rotary-wing aircraft according to its capability of hover, horizontal flight, and vertical take-off and landing (VTOL) [1]. In 1920s, the prototypes of manned quadrotors were introduced for the first time [1, 2]; however, the development of this new type of air vehicle is interrupted for several decades due to various reasons such as mechanical complexity, large size and weight, and difficulties in control especially. Only in recent years a great deal of interests and efforts have been attracted on it; a quadrotor has even become a more optional vehicle for practical application, such as search-and-rescue and emergency response amazingly. As a small, unmanned aerial vehicle (UAV), it has versatile forms from 0.3 to 4 kg. Up to now, some large quadrotors already have sufficient payload and flight endurance to undertake a number of indoor and outdoor applications, like Bell Boeing Quad TiltRotor and so forth [3]. With the improvements of high energy lithium battery, MEMS sensor and other technologies, especially, the scope for commercial opportunities is rapidly increasing [4]. As a quadrotor is inexpensive and easy to be designed and assembled, as well as the complex dynamics, such a rotorcraft is emerging as a fundamental research platform for aerial robotics research for the problems related to three-dimensional mobility and perception [5]. Furthermore, a quadrotor's design and implementation have even become a Multidisciplinary Engineering Undergraduate Course nowadays for the aim to teach students to cope with the challenges, for instance, fast and unstable dynamics and tight integration of control electronics and sensors [6].

For the specific purposes including academic research, commercial usage, and even military aim, many research groups or institutions have fabricated various quadrotors, such as the X4-flyer [7], OS4 [8], STARMAC

[9], and Pixhawk [10] which have become the shining stars mentioned on the network, magazines, and all kinds of academic journals. It is worthy to note that the Draganflyer X4, Asctec Hummingbird, Gauí Quad flyer, and DJI Wookong have been introduced and developed in the comprehensive commercial market. For the powerful operation, some new types of quadrotors with tilting propellers or a new configuration, have been constructed in [10–15] in order to address the issues such as underactuated system. In addition, a number of OSPs (open-source projects) for quadrotors have emerged with contributions from RC hobbyists, universities, and corporations [16].

A quadrotor helicopter is a highly nonlinear, multivariable, strongly coupled, underactuated, and basically an unstable system (6 DOF with only 4 actuators), which acts as preliminary foundation for design of control strategy. Many controllers have been presented to overcome the complexity of the control resulting from the variable nature of the aerodynamic forces in different conditions of flight [17]. Many works have been published on control issues about quadrotors, such as PID controllers [18–21], linear quadratic LQR algorithm [22, 23],  $H\infty$  loop forming method [24], sliding mode variable structure control, feedback linearization [25, 26], backstepping [27, 28], and even intelligent control [29, 30]. In those works above, the linearization of the nonlinear model around hover flight regime is conducted and used to construct controller to stabilize the quadrotor's attitude under small roll and pitch angles. The treatments to the vehicle dynamics, based on some simplistic assumptions, have often ignored known aerodynamic effects of rotorcraft vehicles. In the case of hovering and forward flight with slow velocity, those assumptions are approximately reasonable.

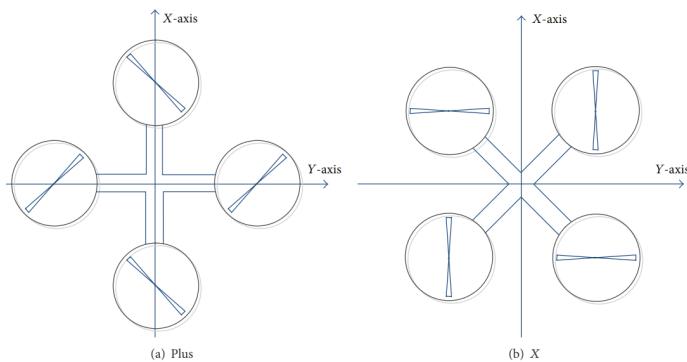
As the quadrotor research shifts to new research areas (i.e., mobile manipulation, aerobatic moves, etc.) [31, 32], the need for an elaborate mathematical model arises, and the simplistic assumption is no more suitable. When aggressive maneuvers such as fast forward and heave flight actions, VTOL, and the ground effect appear, the dynamics of quadrotors could be influenced significantly under these aerodynamic force and moment. It is shown in [33] that existing techniques of modeling and control are inadequate for accurate trajectory tracking at higher speed and in uncertain environments if aerodynamic influence is ignored. The model incorporated with a full spectrum of aerodynamic effects that impact on the quadrotor in faster climb, heave, and forward flight has become an area of active research with considerable effort focusing on strategies for generating sequences of controllers to stabilize the robot to a desired state.

Traditionally, first principle assumptions and measurements of vehicle parameters can be used to derive nonlinear models, from which linear models can be obtained. As an alternative, system identification is also powerful to derive dynamic models directly from flight test data to overcome the challenges such as hardly obtained parameters of the underlying physics of the vehicle. Even though there exists a large volume of multirotor research, there is very little research into system ID of multirotors [34]. The reason for this absence is partially due to the unstable system dynamics of the quadrotor, which makes open-loop identification nonpractical.

This paper provides a tutorial introduction to configuration, modeling, aerodynamics effects analysis, and model identification for quadrotor. The paper's basic structure is as follows: above all, the characteristics and configuration of quadrotor are introduced, then two formulations for the model of quadrotor are compared, and a comprehensive nonlinear equation characterizing the dynamics of quadrotor is derived. Thereupon, aerodynamic effects that impact on the quadrotor in aggressive maneuvers are revealed. At last, several methods about identification are reviewed.

## CHARACTERISTICS OF QUADROTOR

Typically, the structure of a quadrotor is simple enough, which comprises four rotors attached at the ends of arms under a symmetric frame. The dominating forces and moments acting on the quadrotor are given by rotors driven with motors, especially BLDC motors. According to the orientation of the blades, relative to the body coordinate system, there are two basic types of quadrotor configurations: plus and cross-configurations shown in Figure 1.

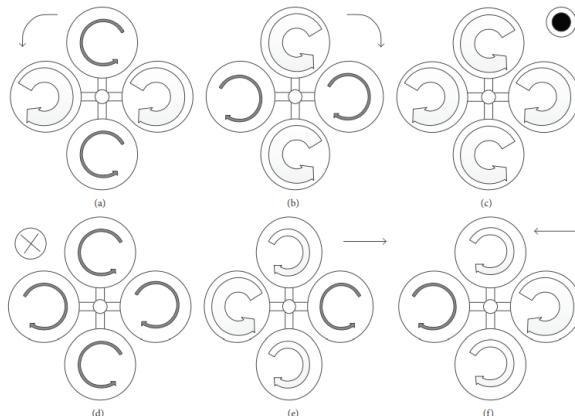


**Figure 1:** Plus and X quadrotor configurations.

In the plus configuration selected by most of the quadrotors, a pair of blades, spinning in the same clockwise or counter-clockwise direction, are fabricated on x and y coordinates of the body frame coordinate system, such as the assemble of the Draganflyer XPro. On the contrary, a different cross-configuration is adopted by some other quadrotors, such as the Convertawings model A, the Piasecki PA-39, or the Curtiss-Wright VZ-7AP, in which there is no rotor at the front or the rear but instead two rotors are on the right side and two on the left.

In contrast with the plus configuration, for the same desired motion, the cross-style provides higher momentum which can increase the maneuverability performance as each move requires all four blades to vary their rotation speed [35]. However, the attitude control is basically analogous [36].

It is the configuration of a quadrotor that shows the inherent characteristics. Basic control sequences of cross-configuration are shown in Figure 2. The quadrotor's translational motion depends on the tilting of rotorcraft platform towards the desired orientation. Hence, it should be noted that the translational and rotational motion are tightly coupled because the change of rotating speed of one rotor causes a motion in three degrees of freedom. This is the reason that allows the quadrotor with six degrees of freedom (DOF) to be controlled by four inputs; therefore the quadrotor is an underactuated system.



**Figure 2:** Quadrotor dynamics: (a) and (b) difference in torque to manipulate the yaw angle ( $\Psi$ ); (c) and (d) hovering motion and vertical propulsion due to balanced torques; (e) and (f) difference in thrust to manipulate the pitch angle ( $\theta$ ) and the roll angle ( $\phi$ ).

In principle, a quadrotor is dynamically unstable and therefore proper control is necessary to make it stable. Despite of unstable dynamics, it is good for agility. The instability comes from the changing rotorcraft parameters and the environmental disturbances such as wind [37]. In addition, the lack of damping and the cross-coupling between degrees of freedom make it very sensitive to disturbances.

## 6-DOF AIRFRAME DYNAMICS

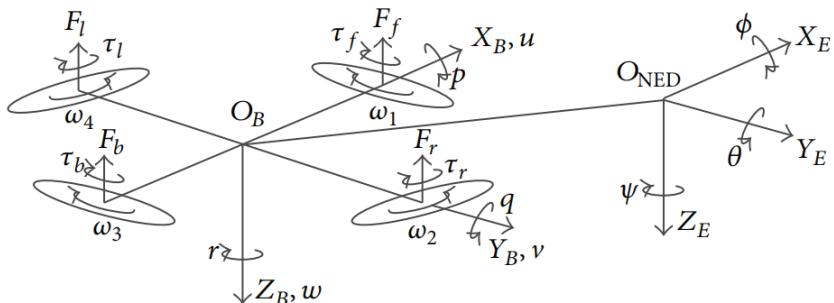
Dominating methods as Euler-Lagrange formalism and Newton-Euler formalism are applied to model the dynamics for an aircraft [38–44]. It has been noted that the Newton-Euler method is easy to be understood and accepted physically despite of the compact formulation and generalization shown by Euler-Lagrange formalism. Nevertheless, two methods are consistent for the description of dynamics. That is to say, it is indicated that after a speed transform matrix the Lagrange equation is an expression form of the second Newton Law [45].

### Euler-Lagrange Formalism

The generalized coordinates of the rotorcraft are given in [46]:

$$\mathbf{q} = (x, y, z, \psi, \theta, \phi) \in R^6, \quad (1)$$

where  $(x, y, z) = \xi \in R^3$  denotes the position of the mass center of the quadrotor relative to the inertial frame and  $(\psi, \theta, \phi) = \eta \in R^3$  are the three Euler angles (resp., yaw, pitch, and roll), under the conditions  $(-\pi \leq \psi \leq \pi)$  for yaw,  $(-\pi/2 \leq \theta \leq \pi/2)$  for pitch, and  $(-\pi/2 \leq \phi \leq \pi/2)$  for roll, which represent the orientation of the rotorcraft (see Figure 3).



**Figure 3:** Quadrotor's body-fixed and inertial coordinate systems.

Naturally, translational and rotational coordinates are obtained from the model

$$\xi = (x, y, z) \in R^3, \quad \eta = (\psi, \theta, \phi) \in R^3. \quad (2)$$

The translational and the rotational kinetic energy of the rotorcraft are

$$T_{\text{trans}} \triangleq \frac{m}{2} \dot{\xi}^T \dot{\xi}, \quad T_{\text{rot}} \triangleq \frac{1}{2} \dot{\eta}^T J \dot{\eta}, \quad (3)$$

where  $m$  denotes the mass of the quadrotor.  $J = W^T I W$  is the moment of inertia matrix in the inertial coordinate system after being transformed from the body frame, by matrix  $W$ :

$$W = \begin{bmatrix} -\sin(\theta) & 0 & 1 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \end{bmatrix}. \quad (4)$$

The only potential energy to be considered is the gravitational potential given by  $U = mgz_E$ .

The Lagrangian of the rotorcraft is

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T_{\text{trans}} + T_{\text{rot}} - U,$$

$$\dot{\xi}^T \dot{\xi} + \frac{1}{2} \dot{\eta}^T J \dot{\eta} - mgz_E = \frac{m}{2}. \quad (5)$$

The full rotorcraft dynamics model is derived from the Euler-Lagrange equations under external generalized forces:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = (F_\xi, \tau), \quad (6)$$

where  $F_\xi = RF$  is the translational force applied to the quadrotor due to the throttle control input,  $\tau \in R^3$  represents the pitch, roll, and yaw moments and  $R$  denotes the rotational matrix  $R(\psi, \theta, \phi) \in SO(3)$ , which represents the orientation of the rotorcraft relative to a fixed inertial frame.

Since the Lagrangian contains no cross-terms in the kinetic energy combining  $\xi$  and  $\eta$ , the Euler-Lagrange equation partitions into two parts. One obtains

$$m\dot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_\xi, \quad (7)$$

$$J\ddot{\eta} + J\dot{\eta} - \frac{1}{2} \frac{\partial}{\partial \eta} (\dot{\eta}^T J \dot{\eta}) = \tau. \quad (8)$$

Rewrite (8) as

$$J\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} = \tau, \quad (9)$$

where  $C(\eta, \dot{\eta})$  is referred to as the Coriolis terms and contains the gyroscopic and centrifugal terms.

## Newton-Euler Formalism

Typically, it is necessary to define two frames of reference, each with its defined righthanded coordinate system, as shown in Figure 3.  $X$ ,  $Y$ , and  $Z$  are orthogonal axes of the body-fixed frame with its correspondent body linear velocity vector  $V' = [u \ v \ \omega]^T$  and angular rate vector  $\Omega' = [p \ q \ r]^T$ . Another one is an Earth-fixed inertial (also known as navigation) coordinate system  $E = (X_E, Y_E, Z_E)$  with which initially the bodyfixed coincides. The attitude of the quadrotor, expressed in terms of the Euler angles  $\phi$  (roll),  $\theta$  (pitch), and  $\psi$  (yaw), is evaluated via sequent rotations around each one of the inertial axes. Herein, a reference frame by  $O_{NED}$  (North-EastDown) denotes an inertial reference frame and  $OB$  a bodyfixed reference frame.

Generally, a quadrotor is considered as a rigid body in a three-dimensional space. The motion equations of a quadrotor subject to external force  $F \in R^3$  and torque  $\tau \in R^3$  are given by the following Newton-Euler equations with respect to the body coordinate frame  $B = (XB, YB, ZB)$ :

$$\begin{bmatrix} mI_{3 \times 3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times mV \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix}. \quad (10)$$

The rotorcraft orientation in space is presented by a rotation  $R$  from  $B$  to  $E$ , where  $R \in SO3$  is the rotation matrix. Here  $c\theta$  is for  $\cos(\theta)$  and  $s\theta$  is for  $\sin(\theta)$ :

$$R = \begin{pmatrix} c_\psi c_\theta & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\theta s_\psi & s_\phi s_\theta s_\psi + c_\theta c_\psi & c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}. \quad (11)$$

With the transformation  $R$ , the first equation assessing the translational

dynamics in (11) can be written in E:

$$m\ddot{\xi} = RF - mgZ_E. \quad (12)$$

Recall the kinematic relationship between the generalized velocities  $\dot{\eta} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$  and the angular velocity  $\Omega = W\eta$ ;  $W \in R^{3 \times 3}$ .

Defining a pseudoinertia matrix  $(\eta) = JW$  and a Coriolis vector  $C(\eta, \eta) = I^T \eta + W \eta \times I^T \eta$ , one can obtain

$$\begin{aligned} m\ddot{\xi} &= RF - mgZ_E, \\ I(\eta)\ddot{\eta} + C(\dot{\eta}, \eta) &= \tau. \end{aligned} \quad (13)$$

This model has the same structure as the one obtained by the Euler-Lagrange approach, in which the main difference is the expressions of  $I$  and  $C$ , which are more complex and more difficult to implement and to compute in the case of the Euler-Lagrange method. It is important to note that the model (13) is common for all aerial robots with six degrees of freedom.

## BASIC DYNAMIC MODEL OF A QUADROTOR

This section introduces the basic quadrotor dynamic modeling with rigid body dynamics and kinematics. This model, based on the first order approximation, has been successfully utilized in various quadrotor control designs so far.

In the first place, some assumptions are reasonable and essential shown as follows [44].

- i The structure is supposedly rigid.
- ii The structure is supposedly symmetrical.
- iii The CoG (center of gravity) and the body fixed frame origin are assumed to coincide.

### Dynamic Model of a Quadrotor

As we know, Newton second law is applied to the translational motion in inertial frames [47]. From the equation of Coriolis, one can obtain

$$m \frac{d\mathbf{v}}{dt_i} = m \left( \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} \right) = \mathbf{f}, \quad (14)$$

where  $m$  is the mass of the quadrotor,  $\mathbf{f}^b \triangleq (fx \ fy \ fz)^T$  is the total force applied to the quadrotor, and  $\mathbf{v}$  is the translational velocity.  $\boldsymbol{\omega}_{b/i}$  is

the angular velocity of the airframe with respect to the inertial frame. Since the control force is computed and applied in the body coordinate system, and since  $\omega$  is measured in body coordinates, (14) is expressed in body coordinates, where  $k^b \triangleq (u, V, \omega)$  and  $w^b_{b/i} \triangleq (p, q, r)^T$ .

For rotational motion, Newton's second law state is

$$\frac{dh}{dt_i} = \frac{dh}{dt_b} + \omega_{b/i} \times h = m, \quad (15)$$

where  $h$  is the angular momentum and  $m$  is the applied torque.  $h^b = Jw^b_{b/i}$ ;  $J$  is the constant inertia matrix. The quadrotor is essentially symmetric about all three axes, which implies that  $J = \text{diag}(J_x, J_y, J_z)$ .

Given  $m^b \triangleq (\tau\phi, \tau\theta, \tau\Psi)^T$ , which denote the rolling torque, the pitching torque, and the total yawing torque, are induced by the rotor thrust and rotor drag acting on the airframe.

The six-freedom-degree model for the quadrotor kinematics and dynamics can be summarized as follows:

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} &= \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\theta c_\psi \\ s_\theta & -s_\phi c_\theta & -c_\phi c_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \\ \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} &= \begin{pmatrix} rv - qw \\ p\omega - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}, \\ \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{pmatrix} &= \begin{pmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \\ \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\theta \\ \frac{1}{J_z} \tau_\psi \end{pmatrix}. \end{aligned} \quad (16)$$

Equation (16) is a full nonlinear model for a quadrotor, in which the complex dynamics is shown obviously, such as strong nonlinearity like the multiplication between system states, intensive coupling among the variables, and the multivariable features intuitively, that imposes the difficulties on the

controller design and, on the other hand, attracts great interest of research.

## Forces and Moments

The forces and torques that act on the quadrotor are primarily due to gravity and the four propellers shown in Figure 3. The steady-state thrust  $F_i$  generated by a hovering rotor (i.e., a rotor that is not translating horizontally or vertically) in free air coincides with  $-ZB$  axis. The total force acting on the quadrotor is given by

$$F = F_f + F_r + F_b + F_l. \quad (17)$$

The rolling torque, the pitching torque, and the total yawing torque are given by

$$\begin{aligned} \tau_\phi &= l(F_l - F_r), & \tau_\theta &= l(F_f - F_b), \\ \tau_\psi &= \tau_r + \tau_l + \tau_f + \tau_b. \end{aligned} \quad (18)$$

The gravity force acting on the center of mass is given by

$$f_g^b = R^T \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} -mgs_\theta \\ mgc_\theta s_\phi \\ mgc_\theta c_\phi \end{pmatrix}. \quad (19)$$

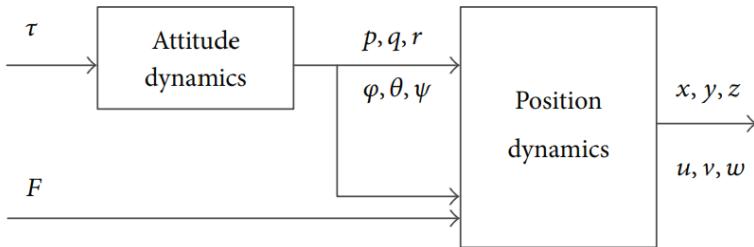
Equation (16) shows strong coupled dynamics [48]: the speed change of one rotor gives rise to motion in at least 3 degrees of freedom.

For instance, the speed decrease of the right rotor will roll the craft to the right under the imbalance between left and right lift forces, coupled with the rotorcraft's yaw to the right due to the imbalance in torque between clockwise and counter-clockwise, so the translation changes direction toward the front.

Nevertheless, in some cases that the rotating movement is slight, the Coriolis terms  $qr$ ,  $pr$ , and  $pq$  are small and can be neglected. So the dynamics of the quadrotor is simplified and given as [47]

$$\begin{aligned}
\ddot{x} &= \frac{(-c_\phi s_\theta c_\psi - s_\phi s_\psi) F}{m}, \\
\ddot{y} &= \frac{(-c_\phi s_\theta s_\psi + s_\phi c_\psi) F}{m}, \\
\ddot{z} &= \frac{g - c_\phi c_\theta F}{m}, \\
\ddot{\phi} &= \frac{1}{J_x} \tau_\phi, \\
\ddot{\theta} &= \frac{1}{J_y} \tau_\theta, \\
\ddot{\psi} &= \frac{1}{J_z} \tau_\psi.
\end{aligned} \tag{20}$$

This model is shown in Figure 4 to which two diagrams in [49, 50] are similar. Note that the attitude of quadrotor is changed, subject to the input  $\tau$  (moment) produced by each rotor. However, the position/altitude dynamics block is affected by  $Tj$  and angle variables.



**Figure 4:** Simplified block diagram of the quadrotor's dynamics.

Intuitively, Figure 4 gives the insight of the dynamic of the quadrotor that the angles and their time derivatives do not depend on translation components, whereas the translations depend on angle (and not on angular velocities) [50]. Based on the characteristics of the dynamics, a quadrotor control problem can be split into two distinct control problems, the inner attitude/altitude loop designed for stability and tracking of desired Euler angles and the outer  $X$ ,  $Y$ , and  $Z$  position loops for regulating the vehicle position [27].

State space equations are applied in the control design and system identification generally. Hence, the nonlinear system of a quadrotor is illustrated as the formulation, which is described in different manner in [3, 27, 51]:

$$\dot{x} = f(x) + g(x)U, \quad (21)$$

where

$$\begin{aligned} x &= [x, y, z, \psi, \theta, \phi, u, v, \omega, p, q, r]^T, \\ y &= [x, y, z, \psi]^T, \\ U &= [F, \tau_\phi, \tau_\theta, \tau_\psi]^T. \end{aligned} \quad (22)$$

Herein the output  $y$  is composed of  $x$ ,  $y$ ,  $z$  and  $\psi$  is for the trajectory track, but if for the hovering control,  $y = [\phi, \theta, \psi, Z]^T$  should be selected because in translation movement shown in (22), the three state variables,  $x$ ,  $y$ , and  $z$ , are subordinated to the same control parameter  $F$ ; hence only one state is controllable and the others are subjected to the controlled translation and angular motions.

## Gyroscopic Torques

At the normal attitude, namely, Euler angles are zero and the axes of the rotors with higher speeds spinning are coincident with the  $zB$  axis of the robot frame. However, while the quadrotor rolls or pitches, the direction of the angular momentum vectors of the four motors is forced to be changed. A gyroscopic torque will be imposed on the airframe that attempts to turn the spinning axis so that it aligns with the precession axis. It is noted that no gyroscopic torque occurs with rotation around the  $zB$  axis (yaw) because the spin and precession axes are already parallel [52].

The gyroscopic (inertial) moment is modeled in [53] as

$${}^b_G \vec{M}_J = \sum_{i=1}^4 J_r \left( \boldsymbol{\omega}_{b/i} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \Omega_i, \quad (23)$$

where  $J_r$  is the gyroscopic inertia, namely, that of the rotating part of the rotor and  $\Omega_i$  is the angular rate of the rotor  $i$  ( $i = 1, 2, 3, 4$ ).

## Quaternion Differential Equations

A problem, so called gimbal lock, will appear with the Euler angle  $\theta$  close to  $\theta = 90^\circ$ , and then the Roll angle  $\phi$  loses its meaning. To overcome this problem, the quaternion method, which offers a mathematical notation that allows the representation of three-dimensional rotations of objects in 4D space, is selected to be the alternative remedy. Reference [54] gives a quaternion dynamics description and [55] proposes a new quaternion-based feedback control scheme for the attitude stabilization of a quadrotor aircraft.

In fact, every parameterization fails to fully represent rigid body pose in every case. That is to say, Euler angles cannot globally represent rigid body pose owing to the gimbal lock, whereas quaternions cannot define it uniquely [56]. Although researchers proved the effectiveness of using quaternions to describe aircraft dynamics, Euler angles are still the most common way of representing rigid body pose.

## Linearized Model

The full nonlinear model is very useful, as it provides insight into the behavior of the vehicle. However, a linear model is used widely, which attributes to the abundance of well-studied tools available for the control system design. As we can see, most of the controllers are based on the nonlinear model with hover conditions and are stable and effective only around reasonably small angles.

Typically, the linearization of a nonlinear state space model  $x = (x) + g(x)U$  is executed at an equilibrium point of the model

$$(x^*, U^*) : f(x^*) + g(x^*)U^* = 0. \quad (24)$$

Then, the linear model is derived by

$$A = \frac{\partial f}{\partial x} \mid x = x^*, \quad B = g(x^*). \quad (25)$$

As the hovering is one of the most important regimes for a quadrotor, at this point, the condition of equilibrium of the quadrotor in terms of (24)-(25) is given as in [54]:

$$\begin{aligned} & U_1^* = mg \\ & U_{2,3,4}^* = 0 \\ (U^*, x^*) : & x_{3,5,6,7,8,9,10,11}^* = 0 \\ & x_{1,2,4,12}^* = \text{random constant.} \end{aligned} \quad (26)$$

While hovering, some assumptions are reasonable, such as the negligible influence of external forces and moments on the aircraft and small velocities and rotational velocities. By performing a Taylor series expansion and eliminating the higher order terms on (20), and using small angle approximations, a linear model is given [38, 57–59]:

$$\begin{aligned}\dot{\phi} &= p, \\ \dot{\theta} &= q, \\ \dot{\psi} &= r, \\ \dot{u} &= -g\theta, \\ \dot{v} &= g\phi, \\ \dot{w} &= g - \frac{F}{m}, \\ I_{xx}\dot{p} &= \tau_\phi, \\ I_{yy}\dot{q} &= \tau_\theta, \\ I_{zz}\dot{r} &= \tau_\psi.\end{aligned}\tag{27}$$

Then the state vector is  $x = [u, \vec{V}, w, p, q, r, \phi, \theta, \psi]$ . Here, the deviations from the trim value act as the states to be considered, and all further references to aircraft states are understood to refer to the perturbation states.

Distinctly different to the linearization at hovering regime, [60] presents a new scheme, which has never before been considered in quadrotor control, in which the linearizations at four points of equilibria are conducted. The four linearizations represent different operating modes in a quadrotor flight mission. These operating modes are

- i      hover;
- ii     vertical motion with a constant velocity;
- iii    horizontal translation with a constant pitch angle tilt;
- iv    horizontal translation with a constant roll angle tilt.

All four linearizations produce four linear time-invariant systems and four controllers accordingly that are simple, low-order, and decentralized

and have integral-action designed for the system stabilization despite of the issues of controller switch between two linear systems.

## AERODYNAMIC EFFECTS

In most of research projects, quadrotor dynamics has often ignored known aerodynamic effects of rotorcraft vehicles because only the stability while hovering is the aim, as stated before. At slow velocities, such as while hovering, this is indeed a reasonable assumption [61]. However, in case of demanding flight trajectories, such as fast forward and descent flight manoeuvres, as well as in the presence of the In Ground Effect, these aerodynamic phenomena could significantly influence quadrotor's dynamics [33], and the performance of control will be diminished if aerodynamic effects are not considered [62], especially in situations where the aircraft is operating at its limits (i.e., carrying heavy load, single engine breakdown, etc.).

Acting as a propulsion system, the aerodynamics of rotors plays the most important role on the movement of the quadrotor excepted with gravity and air drag with respect to the airframe. The kinematics and dynamics of the rotors are fairly complex, resulting from the combination of several types of motion, such as rotation, flapping, feathering, and lagging; normally the last two items are neglectable [3, 4]. The theoretical models based on the blade element theory (BET) combined with momentum theory (MT) show many advantages such as more flexible, more simple, and convenient in contrast with the empirical models based on empirical data typically obtained in the wind tunnel.

Note that the application of helicopter theory to a quadrotor is not straightforward for the reason of many important differences between conventional helicopter and quadrotor [1]. In order to address the issues, the specific research, with the aim at a quadrotor vehicle, is necessary to establish full model with complex dynamics subject to aerodynamic forces and moments. Many works [33, 40, 62–66] on rotor model have been done based on the results obtained for conventional helicopters [67].

Blade flapping is of significant importance in understanding the natural stability of quadrotors [4]. Since it induces forces in the x-y rotor plane of the quadrotor, the underactuated directions in the dynamics, high gain control cannot easily be implemented against the induced forces. On the other hand, the total thrust variation owing to the vertical maneuver also imposes nonignorable influence on the quadrotor behavior.

## Total Thrust

In case of simplifications in aerodynamic effects, the assumption that a rotor's thrust is proportional to the square of its angular velocity is the most common consideration. However, it is proved that this assumption about rotor's thrust is especially far from reality in the cases of nonhovering regime.

The helicopter literatures [67–69] give analysis about many effects on the total thrust in more detail, in which translation lift and change of angle of attack act as the two related effects. As a rotocraft flights across translation, the momentum of the airstream induces an increase in lift force, which is known as translational lift. The angle of attack (AOA) of the rotor with respect to the free-stream also influences the lift, with an increase in AOA increasing thrust, just like in aircraft wings.

Applying blade element theory to quadrotor construction, the expression for rotor thrust  $T$  is given in [56]:

$$T = \frac{N\rho acR^3\Omega^2}{4} \left[ \left( \frac{2}{3} + \mu^2 \right) \Theta_0 - (1 + \mu^2) \frac{\Theta_{tw}}{2} - \lambda_i - \lambda_c \right] \quad (28)$$

and a thrust coefficient  $C_T$  is given in

$$C_T = \left( \frac{2}{3} + \mu^2 \right) \Theta_0 - (1 + \mu^2) \frac{\Theta_{tw}}{2} - \lambda_i - \lambda_c, \quad (29)$$

where  $\mu$ ,  $\lambda_i$ , and  $\lambda_c$  are speed coefficients  $V_{xy}/R\Omega$ ,  $V_z/R\Omega$ , and  $V_i/R\Omega$ , respectively.  $V_{xy}$  and  $V_z$  are the horizontal and vertical components of the total air stream velocity, respectively, and  $V_i$  is the induced velocity, and  $\Omega$  is rotor angular speed. Herein, the other parameters and coefficients in the formulation above will not be described and refer to the literature [56].

Especially at hovering regime,  $\mu=0$  and  $\lambda_c = 0$  (i.e., static conditions) yield

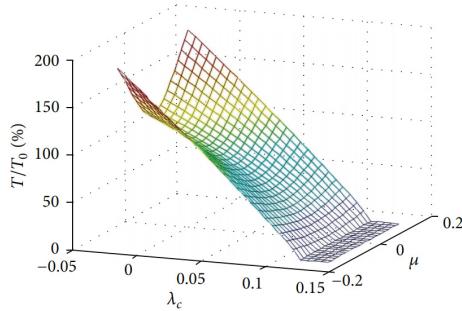
$$T = \frac{1}{2}\rho a\bar{c}R^3 (\Theta_{3/4} - \lambda_i) \Omega^2. \quad (30)$$

So one can obtain  $T \propto \Omega^2$  just like the relationship between  $T$  and  $\Omega$  used in the upwards context.

For the calculation of the aerodynamic coefficient  $CT$  it is crucial to know three airspeed coefficients  $\mu$ ,  $\lambda_c$ , and  $\lambda_i$ . Two of them,  $\mu$ ,  $\lambda_c$ , can easily be obtained from the available motion data  $V_{xy}$ ,  $V_z$ , and  $\Omega_R$ .  $\lambda_i$  however is very hard to know, because it is impossible to measure the induced velocity  $V_i$ .

One can solve this problem by means of calculating the induced velocity coefficient  $\lambda_i$  involved in the two aerodynamic principals, momentum, and blade element theories. In view of the fact that the macroscopic momentum equation and the microscopic blade element equation give the same rotor thrust formulation:

$$\begin{aligned} T &= \frac{N\rho\bar{a}\bar{R}^3\Omega^2}{4} \left[ \left( \frac{2}{3} + \mu^2 \right) \Theta_0 - (1 + \mu^2) \frac{\Theta_{tw}}{2} - \lambda_i - \lambda_c \right] \\ &= 2\rho R^2 \pi \lambda_i \sqrt{(\lambda_c + \lambda_i)^2 + \mu^2 + \frac{\lambda_c^2}{7.67}}. \end{aligned} \quad (31)$$



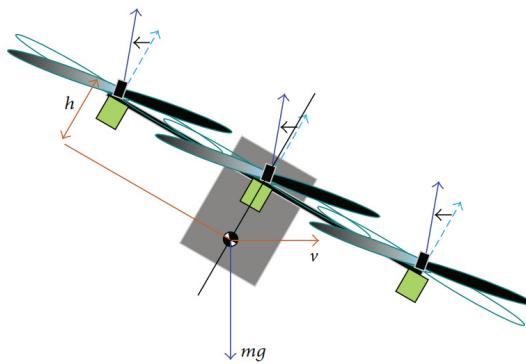
**Figure 5:**  $T/T_0$  ratio during  $x$ - $y$  plane movement [61].

The results of solving this equation can be shown in Figure 5.

The induced velocity decreases with an increase of airflow produced by quadrotor movement, which can be seen in Figure 5. Although both direction's movements in  $x$ - $y$  plane tend to increase induced velocity, only the vertical movement decreases the thrust coefficient. As a result, during takeoff the quadrotor loses rotor thrust, but during horizontal movement that same thrust is increased and enables more aggressive maneuvers [62, 70].

## Blade Flapping

When a rotor translates horizontally through the air, the advancing blade of the rotor has a higher velocity relative to the free-stream and will generate more lift than the retreating blade which sees a lower effective airspeed. This causes an imbalance in lift, inducing an up and down oscillation of the rotor blades, which is known as blade flapping [63, 64]. When a quadrotor is in steady state suffering the blade flapping, its rotor plane will tilt at some angle off of vertical, causing a deflection of the thrust vector illustrated in Figure 6.



**Figure 6:** Effect of blade flapping [33].

In fact, the rotor thrust is perpendicular to the rotor plane and not to the hub of the rotor. Thus, in the case of blade flapping the rotor disk tilts, the rotor thrust is also inclined with respect to the airframe and imposes a component in the x and y directions of the body-fixed frame. As a result, the flapping of the blades results in a variety of effects on the dynamics of the vehicle, and in particular affecting attitude control performance [65], owing to the reason that a moment is produced for the rotor plane aligned with the vehicle's center of gravity, and the effect increases with speed.

Note that the lateral flapping was neglected in above considerations, as due to quadrotor's symmetry and in-pair counter rotation of rotors; its net influence is negligibly small in all instances of forward flight. Hence, there are two moments that need to be considered. First of all, the moment  $M_{b,\text{lon}}$  is caused by the longitudinal thrust due to a deflection angle  $\alpha_{1S}$  between the rotor plane and rotorcraft platform:

$$M_{b,\text{lon}} = Th \sin \alpha_{1S}, \quad (32)$$

where  $h$  is the vertical distance from the rotor plane to the center of gravity of the vehicle and  $T$  is the thrust.

In addition, in the case of stiff rotors without hinges at the hub, there is also a moment  $M_b$ , generated directly at the rotor hub from the flapping of the blades:

$$M_{b,s} = k_\beta \alpha_{1S}, \quad (33)$$

where  $k_\beta$  is the stiffness of the rotor blade in Nm/rad. The total longitudinal moment created by blade flapping  $M_{bf}$  is the sum of these two moments:

$$M_{b,f} = M_{b,\text{lon}} + M_{b,s}. \quad (34)$$

Although a controller designed exactly is possibly successful to counteract small disturbances, it is difficult to reject the large systematic disturbances that result from the aerodynamic effects such as blade flapping. For the improvement of control performance, it is necessary to design a feed forward compensation block in order to cancel out moments and forces resulting from blade flapping and variations in total thrust [33].

## Ground Effect and Ceiling Effect

When a rotor operates near the ground (about at half rotor diameter), a phenomenon always appears that thrust augmentation pushes the vehicle away from the ground, which is related to a reduction of the induced airflow velocity.

This is called ground effect [71, 72]. Different from other approaches, an adaptive technique [73] is an option to deal with this effect. However, for the aim of improvement of the autonomous VTOL controller, a principal model of this effect is needed.

One proposed mathematical model of ground effect [74] is

$$\frac{T}{T_\infty} = \frac{1}{1 - (R/4Z)^2}. \quad (35)$$

Here  $R$  is the radius of the rotor,  $z$  is the vertical distance from the ground,  $T$  is the thrust produced by the propeller in ground effect, and  $T_\infty$  is the thrust produced at the same power outside of ground effect.

Note that for  $z/R = 2$  the predicted ratio between  $T$  and  $T_\infty$  is just 1.016. Therefore, this formula (35) predicts that ground effect is negligible when the rotor is more than one diameter off the ground, that is,  $z/R > 2$ .

Except for the ground effects, a “ceiling effect” is another issue needs to be researched, the reason is a quadrotor can flight indoor different from conventional helicopter.

In fact, so called “ceiling effect” means when the vehicle is close to an overhead plane, the ceiling effect pulls the vehicle towards the ceiling which can cause a crash in the worst case.

The effects have been proved by a set of experiments. Unfortunately, no formal formulation of a ceiling effect is presented so far.

## IDENTIFICATION OF A QUADROTOR MODEL

Model is the foundation and the first step of control and simulation. In general, system models are derived from first principles, physical insight and sometimes input and output data, and the last two items are classified as system identification. As for a Quadrotor, the first principle scheme is intuitive to obtain the dynamical model, perhaps from the inheritance of traditional flight mechanics, by which the relationship between the inputs and outputs, that is, the underlying dynamics is revealed distinctly. However, on the other hand, the mathematical formulation proposed is characterized by the unwelcome complexity and strong nonlinearity that is regarded as a nightmare for controller design.

As an alternative solution, system identification is effective to derive a model. System identification, as the art and science of building mathematical models of dynamic systems from observed input-output data, has developed for few decades, starting from the year 1965, and enormous methods are presented. However, there are many open problems [75, 76]: such as nonlinearity and closed-loop identification, which are just the characteristics shown in the quadrotor dynamics.

Significantly, the attention on the model and identification aspect is paid on the fixed-wing and helicopter [34], instead of quadrotor, or multirotor, and the reason may be the fact of less applications of quadrotor aircraft by now, as well as relative complicated dynamics which exhibit some distinctive features on the modeling and identification schemes, presented as follows [77].

- i Continuous-time model is preferable. In aerospace applications, a continuous-time model looks more popular than the discrete-time one for the reason of intuition.
- ii To support closed-loop identification, a quadrotor system is basically open-loop unstable, so that identification experiments have to be implemented in closed loop under automatic control, or with a human operator.
- iii To uncouple the cross-interaction among the operation axes, the cross-coupling in dynamic behavior imposes difficulty on the estimation of a model, so that isolation of different dynamic modes is necessary to alleviate this effect.

To the best of my knowledge, although not many, some schemes dedicated to the quadrotor model identification are proposed in the following.

Herein, by means of the classification in the literature [75], so called a whole palette of grey shades from white to black, these methods are introduced successively.

## **Off-White Models: Parameters Estimate Based on First Principle Model**

A set of parameter estimates in the nonlinear quadrotor model could be taken directly from measurements, a CAD model using model software like SOLIDWORKS to model all the parts of the quadrotor, or derive from experiments, along with the associated error [66, 70, 78]. The parameters listed in the following context can be obtained by a regular identification method [79–81].

- i Aerodynamic parameters: rotor, blade, and aerodynamic parameters are obtained through measurement, computation, simulation, or from references.
- ii Masses and displacements: component masses and distances are measured with respect to the rotor plane.
- iii Rotational inertia is obtained through measurement and computation [2].
- iv Motor constants: resorting to some experiments conducted, the motor model can be simplified to be a first order system and the constants could be extracted from the experimental data [82, 83].

The nonlinear identification problem is to estimate such parameters from the measured data. In general, this is a difficult problem that has not yet been treated in full generality. In [84], a Levenberg-Marquardt optimization method and a quadratic optimization method are applied, respectively, to obtain the z inertia and the rotor parameters.

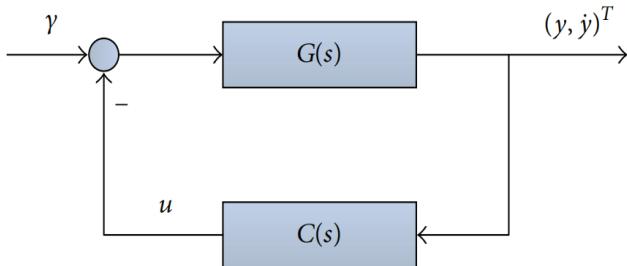
## **Steel-Grey Models: Local Linear Models Identification**

Nonlinear systems are often handled by linearization around a working point. The idea behind composite local models is to deal with the nonlinearities by developing local models, which are good approximations in different neighborhoods, and then compose a global model from these. Often, the local models are linear, so a common name for composite models is also local linear models [75], as described in Section 4.5, where the hover regime is acted as a working point. After the linearization at working point, the identification issue is simplified and easy to tackle with the help of linear

identification methods as follows. However, the linearization is a realistic simplification after all, which results in the bias, even model mismatch, and sometimes is unreasonable in the case of aggressive maneuvers. In addition, a set of models need to be derived in the situations that many working points exist, so the switch between two models in the model set should be paid enough attention to weaken the disturbance.

### **Parameter Identification**

Using the linearized system dynamics after some treatments such as neglecting the nonlinear coupling terms, a parameter identification [85] is performed to identify separately each quadrotor axis performed in closed loop. The generic scheme of the identification process is depicted in Figure 7.



**Figure 7:** Indirect closed-loop identification setup.

The controller ( $s$ ) used during identification is a simple stabilizing, hand-tuned PD-controller with known parameters. Using linearized system dynamics equation, the identification signal, that is, a pseudorandom binary sequence (PRBS) of full length, and the controller  $C$  and output data  $(y, y)^T$ , one can use nonlinear optimization to estimate that our parameter vectors  $\Delta\phi, \theta = (p, J, T_d)^T$  and  $\Delta\psi = (p, z, J, T_d)^T$  are conducted, respectively. The experimental results show a very good correlation with real data, which confirms the proposed approach in which an iterative parameter identification scheme is applied, the results of which can easily be reproduced and offers great accuracy. In general, a regular procedure is implemented for the parameter identification, in which a linear model is derived from the nonlinear one with some simplification and neglecting, and then a parameter identification problem is shown, at last a iterative algorithm is applied to obtain the estimated value of the parameters, just like what is described in this literature. Next, an adaptive controller could be designed based on the parameter identification.

## ***Time Domain Identification***

Reference [86] presents the estimation of a linear mathematical model for the dynamics of a quadrotor by time domain system identification.

At first, the model structure has to be determined by the begin with a large pool of potential candidate regressors, meaning states or control inputs, or some combination thereof, then calculate the potential correlation of each regressor with a state derivative using a linear least-squares method. The final step in the process is to retain those regressors which have a significant correlation with the state derivative in question. The level of correlation is determined by the user so as to capture as much of the measured behavior as possible with a minimal number of regressors. An example of such a pool of regressors is given as follows:

$$\dot{u} = X_u u + X_q q + X_\theta \theta + X_{\text{lon}} \delta_{\text{lon}}, \quad (36)$$

where the state derivative  $\dot{u}$  is linearly related to the regressors,  $u$ ,  $q$ ,  $\theta$ , and  $\delta_{\text{lon}}$ , by their corresponding parameters,  $X_u$ ,  $X_q$ ,  $X_\theta$ , and  $X_{\text{lon}}$ . This process of regressor pool correlation is repeated for each state derivative.

After the appropriate model structure has been determined, the next step is to determine the value and error of each parameter by a linear least squares method. These values form the dynamics and control matrices,  $A$  and  $B$ . At the same time, the error values are also adjusted to account for any remaining uncharacterized behavior, known as colored residuals.

The method herein is considered as the basic algorithm in the realm of system identification, which is used to address the model identification issue of linear system. It should be noted that the method is desirable for the SISO (single input and single output) system; therefore, such characteristics as cross-coupling must be mitigated in advance.

## ***Frequency-Domain Identification***

A frequency-domain system identification method is used to obtain a linear representation of the quadrotor dynamics [87]. Contrast to time domain analysis, frequency-domain identification can obtain a relative robust model with the treatment of cutting down the errors associated with bias effects and processing noise, resulting in a robust model.

In the algorithm, the frequency response data acquired is validated by evaluating its coherence, which is an indication of how well the output and input data are correlated. The definition of coherence is given as

$$\Upsilon_{xy}^2 \omega = \frac{|\widehat{G}_{xy}(f)|^2}{|\widehat{G}_{xx}(f)| |\widehat{G}_{yy}(f)|}, \quad (37)$$

where  $\widehat{G}_{xx}(f)$ ,  $\widehat{G}_{yy}(f)$ , and  $\widehat{G}_{xy}(f)$  represent the autospectral densities of the input, output, and cross-spectral density of the input and output, respectively, and are the frequency point. A perfect correlation between input and output would result in a coherence value of unity, while poor coherence typically falls below a value of 0.6.

It might also be noted that the data must be decoupled such that the inputs provided by off-axis commands are rejected from the output on the axis of interest, after the coherence of the data is validated. The multiple single output system estimation can be expressed in (38), where  $H$  is the system estimation:

$$\widehat{H}(f) = \widehat{G}_{xx}^{-1}(f) \widehat{G}_{xy}(f). \quad (38)$$

In the system identification process, the transfer functions of each axis will be acquired first, followed by state space representations and complete system analysis. The single input-single output (SISO) transfer function identification cost function can be defined as

$$J = \frac{20}{n_\omega} \sum_{\omega_1}^{\omega_{n_\omega}} W_g \left[ W_g \left( |\widehat{T}_c| - |T| \right)^2 + W_p \left( \angle \widehat{T}_c - \angle T \right)^2 \right], \quad (39)$$

where the parameters such as  $n_\omega$ ,  $\omega_1$  refer to [87].

As shown in [87], based on the rational experimental setup, a frequency-domain system identification method obtains a linear representation of the quadrotor dynamics. It might also be noted that the choice of the periodic excitation signal is to minimize leakage in the computation of frequency spectra, which is still an open problem in the area.

### **Subspace Model Identification**

A subspace model identification (SMI) method [77], which has been proved extremely successful in dealing with the estimation of statespace models for multiple-input multiple-output (MIMO) systems is used to the identification of a quadrotor flight dynamics. More precisely, the continuous-time predictor based subspace identification approach proposed is applied to flight data collected during dedicated identification experiments, and at hovering flight condition, a linear statespace model is derived. As an advantage over the

most identification techniques, this approach is feasible for the application in a closed-loop system as the correlation between  $u$  and  $w$ ,  $V$  is not required. The key ideas of the algorithm are provided in the following.

Consider the linear time-invariant continuous-time system:

$$dx(t) = Ax(t)dt + Bu(t)dt + dw(t), \quad x(0) = x_0,$$

$$dz(t) = Cx(t)dt + Du(t)dt + dv(t),$$

$$y(t)dt = dz(t), \quad (40)$$

where  $x \in R^n$ ,  $u \in R^m$ , and  $y \in R^p$  are, respectively, the state, input, and output vectors and  $w \in R^n$  and  $V \in R^p$  are the process and the measurement noise, respectively, modeled as Wiener processes with incremental covariance given by

$$E \left\{ \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix} \begin{bmatrix} dw(t) \\ dv(t) \end{bmatrix}^T \right\} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix}. \quad (41)$$

The system matrices  $A$ ,  $B$ ,  $C$ , and  $D$ , of appropriate dimensions, are such that  $(A, C)$  is observable and  $([B, Q^{1/2}], )$  is controllable. Assume that a data set  $(t_i, y(t_i))$ ,  $i[1, N]$  of sampled input/output data obtained from (41) is available. Then, the problem is to provide a consistent estimate of the state space matrices  $A$ ,  $B$ ,  $C$ , and  $D$  on the basis of the available data.

Note that both model order and the tuning parameters of the identification algorithm (i.e., the position of the Laguerre pole  $a$  and the parameters of the PBSID<sub>opt</sub> algorithm) need to be achieved at the head of the procedure; herein, a crossvalidation approach, explained in detail in the literature [77], is used to address the issue.

As can be observed from the experiments, in which the input signal adopted for identification experiments is the so-called 3211 piecewise constant sequence, the identified models capture the essential features of the response of the quadrotor along all the axes. As it is known that the SMI method is rapid and easy to use, however, the model deserved from the algorithm is not based on some kind of optimal criteria, so the model obtained is also not believed to be optimum.

### **UKF Method**

As we know, if the systems have severe nonlinearities, EKF can be hard

to tune and often gives unreliable estimation due to the linearization relied by the EKF in order to propagate the mean and covariance of the states. Therefore, UKF is applied for the identification of a quadrotor model [88].

For the quadrotor system with continuous-time system dynamics,

$$\dot{x} = Ax + Bu,$$

$$y = Cx. \quad (42)$$

Since the state vector of the quad-rotor

$$x_k = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})^T \quad (43)$$

the system state equation can be derived according to the full nonlinear equations. Finally, based on all of the system equations, the parameters to be estimated and identified are formulated as follows:

$$\Theta = (I_{xx}, I_{yy}, I_{zz}, I_R, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})^T. \quad (44)$$

Based the experiments, the error of the estimation for velocity at  $x$ -axes is less than 0.001, while the errors at both  $y$ - axes and  $z$ -axes are less than 0.0015. For estimation of angular velocities at  $x$ ,  $y$ , and  $z$ -axes, the errors are less than 0.0015. From the errors computed, it can be concluded that the UKF output matches with the measured output and the measured noise is well filtered by the UKF. It has good convergence time and relatively reliable for the estimations.

## Slate-Grey Models: RBF-ARX Model

The RBF-ARX model is a nonlinear time-varying model whose structure resembles the ARX model. Its independent variables are groups of signals indicating the nonlinear status of the system, and its model parameters can be promptly adjusted to the best by taking the advantages of the RBF neural network. Owing to the self-adjusting parameters, the RBF-ARX model not only has an outstanding approximation in local linear space but also has superior global performance. RBF-ARX basic model structure of a quadrotor is shown as follows [89, 90]:

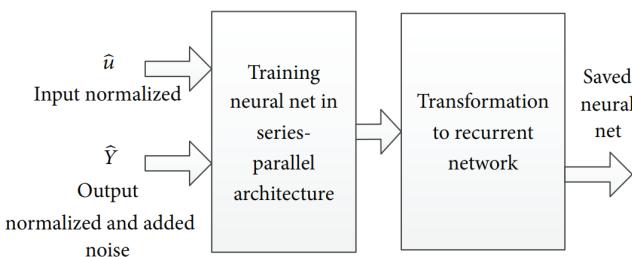
$$\begin{aligned}
y(t) &= c(\omega(t)) + \sum_{k=1}^{n_y} A_k(\omega(t)) Y(t-k) \\
&\quad + \sum_{k=d}^{n_u} B_k(\omega(t)) U(t-k) + e(t), \\
C(\omega(t)) &= [\phi_0^1(\omega(t)) \phi_0^2(\omega(t)) \phi_0^3(\omega(t))]^T, \\
\phi_0^i(\omega(t)) &= c_0^1 + \sum_{m=1}^h c_m^i \exp \left\{ -\|\omega(t) - z_{Y,m}\|_{\lambda_{r,m}}^2 \right\}, \tag{45}
\end{aligned}$$

where  $n_y$ ,  $n_u$ ,  $m$ , and  $h$  are the order of the system,  $d$  is delay factor of system, and  $e(t)$  is white noise. A quadrotor system is a nonlinear system with four inputs and 3 outputs, and the inputs are the input electric voltage of four rotors, that is,  $U(t) = [Vf(t) Vr(t) Vl(t) Vb(t)]^T$ , and the three outputs are pitch, roll, and yaw angles, respectively, that is,  $Y(t) = [p(t) r(t) y(t)]^T$  and  $w(t) = [p(t) r(t)]^T$ . A set of simulation tests show that the error of RBF-ARX model is most close to a normal distribution, which indicates that the good model is obtained. In addition, no matter how the state variable ( $t$ ) slides, the distribution of system pole does not go beyond the stable scope. So the RBF-ARX model is suitable for quadrotor.

## Black Models

### *Neural Network Model*

A black box model that uses a neural network to learn the dynamics of the quadrotor is attempted [79]. The nonlinear autoregressive network with exogenous inputs (NARX) architecture is setup which has 6 different nets, one each for the  $X$ ,  $Y$ , and  $Z$  velocities and roll, pitch, and yaw rates. The other variables are then integrated (position/angles) and derive (accelerations) from the calculated velocities/rates. The procedure, described in Figure 8, is designed to address the issue of the current prediction ( $y_t$ ) dependent on the predicted output for the previous time point ( $y_{t-1}$ ), in which a series-parallel architecture, ( $t-1$ ) is the correct output for time point  $t-1$ ,  $u(t)$  is the input vector for time point  $t$ , and  $y(t)$  is the prediction for time point  $t$ , is used for training before converting the net to the parallel architecture.



**Figure 8:** The flow for training the neural net.

Some tests result shows that the black box neural network model can predict both the roll and pitch with very good accuracy and however perform not better for the yaw rate, which will be improved by creating a larger net or by adding more variables to the state vector for the nets. Surprisingly, the model was tested on a manoeuvre for which it had not been trained; a successful result is obtained. This result shows that the black box neural network model learned not only the dynamics of the quadrotor but also the dynamics of the trends and the noise. Neural networks have proved to be a remarkable modeling scheme carried out in various applications; however, on the other hand, substantial amount of tests need to be taken. Obviously, the work is costly and time-consuming.

### **Data-Based Model**

The main purpose of data-based techniques is to take full advantage of the information acquired from huge amounts of process measurements. Without recourse to physical models obtained from first principles, a relatively overall perspective of system performance could be revealed via available measurements. Through deep insights of process measurements, information like system characteristics and regularity can be dug out for optimal modeling and decision making. The description [91] aforementioned reveals the insight and application of data-based model identification schemes.

It is noted that several typical data-based approaches, which only depend on process measurements, principal component analysis (PCA), partial least squares (PLS), and their variants, are successfully utilized in many areas [92–96]. In the realms of model and control, iterative learning control (ILC) scheme, and model free adaptive control (MFAC)—in essence, model free methods—show great advantages without a priori knowledge about the underlying processes, such as time delay and system order, despite their potential limit for processes with high complexity.

In addition, the recent developments on model-data integrated approaches, which also rely on available process measurements and a prior known limited knowledge about the processes for monitoring and control purposes, the iterative feedback tuning (IFT), and virtual reference feedback tuning (VRFT), have become the promising research topics. Although there are no reports of the application of the data-based model identification on the quadrotor so far, these approaches will soon find their utilizations.

*Remark.* Hover condition is the main status of the quadrotor, as quite a few tasks, such as surveillance, search, and rescue, are implemented in the condition.

Therefore, linear model is simplified on the complex nonlinear one derived from first principle model, in which the feasibility is proved by application result. However, aggressive maneuver shows the obvious nonlinear characteristics, so that the nonlinear model is needed, in which neural networks are an optional scheme despite the fact that a large amount of tests are indispensable.

The data-based approaches have shown the distinctive advantages in other application areas; the utilization on a quadrotor will be a commendable attempt. It is noted that these methods aforementioned not only are applicable in the quadrotor aspect but also is helpful to the field with the similar characteristics, such as active suspension systems of vehicle investigated in [97, 98].

## CONCLUSION AND PERSPECTIVE

By now, a quadrotor has been a preferable platform of aircraft design and autonomous control due to the distinct characteristics involved in the craft configuration and flight dynamics, and the expansion to the commercial and military application is underway.

The research results at present show that the static and dynamic characteristics of a quadrotor have been developed, and the aerodynamic effects on the craft flight performance have been revealed with the help of the conventional rotor theory.

As a base and preliminary for the next control and simulation work, the paper surveys the state of art of the modeling and identification of a quadrotor; afterwards some opinions are given as follows.

- 1) The full nonlinear equations have been obtained by classical Euler-Newtonian mechanics as well as Lagrange equation,

with the knowledge of different force imposed on, especially aerodynamic function. However, there is no specific research of the model of a quadrotor, in which the dynamics is described comprehensively and systematically.

- 2) The dynamics of quadrotor shows distinct characteristics for the VTOL, horizontal flight, aggressive maneuver, and hovering mode involved in the execution of flight task. Whereas at the present, the hovering mode attracts over much attention, probably owing to the fact that the hovering is the one of the important working modes. Obviously, more efforts of research should be paid on the other modes such as aggressive maneuver.
- 3) The method of multistatic status linearization model has been presented at the aim of control, which may be an effective way to address the issue of complex dynamics. However, a problem of undisturbed switching between the multimodels should be resolved.
- 4) In contrast with the other crafts, there is no more attention for the model identification of a quadrotor for the reason of the requirements mentioned in the paper. The effective transplantation of the identification methods used in the other respects is a practical choice.
- 5) A new configuration of quadrotor with tilt rotor can eliminate the deficiency such as underactuate feature. The relative research work has been in progress, which will be a focus hereafter.

## CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests regarding the publication of this paper.

## ACKNOWLEDGMENTS

This work was supported by the Fundamental Research Funds for the Central Universities (Grant no. FRF-TP-12-005B), Program for New Century Excellent Talents in Universities (Grant no. NCET-11-0578), Aeronautical Science Foundation of China (Grant no. 2012ZD54013), and Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20130006110008).

## REFERENCES

1. V. Martinez, *Modeling of the flight dynamics of a quadrotor Helicopter [M.S. thesis]*, Cranfield University, Cranfield, UK, 2007.
2. M. Y. Amir and V. Abbass, “Modeling of quadrotor helicopter dynamics,” in *Proceedings of the International Conference on Smart Manufacturing Application (ICSMA ‘08)*, pp. 100–105, Kintex, April 2008.
3. G. Warwick, “US Army looking at three configuration concepts for large cargo rotorcraft,” September 2013, <http://www.flighthglobal.com/news/articles/us-army-looking-at-three-configuration-concepts-for-large-cargo-rotorcraft-217956/>.
4. R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: modeling, estimation, and control of quadrotor,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.
5. R. Mahony and V. Kumar, “Aerial robotics and the quadrotor,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, p. 19, 2012.
6. I. Gaponov and A. Razinkova, “Quadcopter design and implementation as a multidisciplinary engineering course,” in *Proceedings of the 1st IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE ‘12)*, pp. B16–B19, IEEE, Hong Kong, China, August 2012.
7. N. Guenard, T. Hamel, and R. Mahony, “A practical visual servo control for an unmanned aerial vehicle,” *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, 2008.
8. S. Bouabdallah and R. Siegwart, “Towards intelligent miniature flying robots,” in *Tractsaction in Advanced Robotics*, vol. 25, pp. 429–440, Springer, Berlin, Germany, 2006.
9. G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, “The stanford testbed of autonomous rotorcraft for multi agent control (STARMAC),” in *Proceedings of the 23rd Digital Avionics Systems Conference (DASC ‘04)*, vol. 2, pp. 12.E.4–121.10, Salt Lake City, Utah, USA, October 2004.
10. L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “PIXHAWK: a system for autonomous flight using onboard computer vision,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA ‘11)*, pp. 2992–2997, IEEE, Shanghai, China, May 2011.

11. S. K. Phang, C. Cai, B. M. Chen, and T. H. Lee, “Design and mathematical modeling of a 4-standard-propeller (4SP) quadrotor,” in *Proceeding of the 10th World Congress on Intelligent Control and Automation (WCICA'12)*, pp. 3270–3275, Beijing, China, July 2012.
12. M. S. Hossain, A. M. Kabir, and P. Mazumder, “Design and development of an Y4 Copter control system,” in *Proceedings of the 14th International Conference on Computer Modeling and Simulation (UKSim'12)*, pp. 251–256, Cambridge, UK, March 2012.
13. F. Senkul and E. Altug, “Modeling and control of a novel tilt: roll rotor quadrotor UAV,” in *Proceeding of the International Conference on Unmanned Aircraft Systems (ICUAS'13)*, pp. 1071–1076, Atlanta, Ga, USA, May 2013.
14. M. Ryll, H. H. Bulthoff, and P. R. Giordano, “First flight tests for a quadrotor UAV with tilting propellers,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*, pp. 295–302, IEEE, Karlsruhe, Germany, May 2013.
15. S. Driessens and P. E. I. Pounds, “Towards a more efficient quadrotor configuration,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '13)*, pp. 1386–1392, Tokyo, Japan, 2013.
16. H. Lim, J. Park, D. Lee, and H. J. Kim, “Build your own quadrotor: Open-source projects on unmanned aerial vehicles,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 33–45, 2012.
17. D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 2520–2525, IEEE, Shanghai, China, May 2011.
18. Q. Zhan, J. Q. Wang, and X. Xi, “Control system design and experiments of a quadrotor,” in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '12)*, pp. 1152–1157, IEEE, Guangzhou, China, December 2012.
19. L. Jun and Y. T. Li, “Dynamic analysis and PID control for a quadrotor,” in *Proceedings of the International Conference on Mechatronics and Automation*, pp. 573–578, IEEE, Beijing, China, August 2011.
20. A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid, “Flight PID controller design for a UAV quadrotor,” *Scientific Research and Essays*, vol. 5, no. 23, pp. 3660–3667, 2010.

21. A. B. Milhim, Y. Zhang, and C. A. Rabbath, “Quadrotor UAV-high fidelity modeling and nonlinear PID control,” in *Proceedings of AIAA Modeling and Simulation Technologies Conference*, pp. 2010–8362, Toronto, Canada, 2010.
22. R. Zawiski and M. Blachuta, “Model development and optimal control of quadrotor aerial robot,” in *Proceedings of the 17th International Conference on Methods and Models in Automation and Robotics (MMAR '12)*, pp. 475–480, Miedzyzdroje, Poland, August 2012.
23. E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, “LQR control for a quadrotor using unit quaternions: modeling and simulation,” in *Proceedings of the 23rd International Conference on Electronics, Communications and Computing (CONIELECOMP '13)*, pp. 172–178, IEEE, Cholula, Mexico, March 2013.
24. M. Rich, N. Elia, and P. Jones, “Design and implementation of an  $H\infty$  controller for a quadrotor helicopter,” in *Proceeding of the 21st Mediterranean Conference on Control and Automation (MED'13)*, pp. 1189–1198, Chania, Greece, June 2013.
25. P. Mukherjee and S. L. Waslander, “Direct adaptive feedback linearization for quadrotor control,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pp. 2012–4917, Minneapolis, Minn, USA, August 2012.
26. A. Khalifa, M. Fanni, A. Ramadan, and A. Abo-Ismail, “Modeling and control of a new quadrotor manipulation system,” in *Proceedings of the 1st International Conference on Innovative Engineering Systems (ICIES '12)*, pp. 109–114, Alexandria, Egypt, December 2012.
27. L. Pollini and A. Metrangolo, “Simulation and robust backstepping control of a Quadrotor aircraft,” in *Proceedings to AIAA Modeling and Simulation Technologies Conference and Exhibit*, pp. 2008–6363, Honolulu, Hawaii, USA, 2008.
28. A. A. Mian and D. B. Wang, “Modeling and backstepping-based nonlinear control strategy for a 6 DOF quadrotor helicopter,” *Chinese Journal of Aeronautics*, vol. 21, no. 3, pp. 261–268, 2008.
29. F. Casolo, *Motion Control*, InTech, Croatia, Rijeka, 2010.
30. M. Santos, V. López, and F. Morata, “Intelligent fuzzy controller of a quadrotor,” in *Proceedings of the IEEE International Conference on Intelligent Systems and Knowledge Engineering (ISKE '10)*, pp. 141–

- 146, IEEE, Hangzhou, China, November 2010.
31. C. M. Korpela, T. W. Danko, and P. Y. Oh, “MM-UAV: mobile manipulating unmanned aerial vehicle,” *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1–4, pp. 93–101, 2012.
  32. D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers,” *Springer Tractsaction in Advanced Robotics*, vol. 79, pp. 361–373, 2014.
  33. H. M. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09)*, pp. 3277–3282, IEEE, Kobe, Japan, May 2009.
  34. N. V. Hoffer, C. Coopmans, A. M. Jensen, and Y. Q. Chen, “A survey and categorization of small low-cost unmanned aerial vehicle system identification,” *Journal of Intelligent and Robotic Systems*, vol. 74, no. 1-2, pp. 129–145, 2014.
  35. S. Gupte, P. I. T. Mohandas, and J. M. Conrad, “A survey of quadrotor unmanned aerial vehicles,” in *Proceeding of the IEEE Southeastcon*, pp. 1–6, Orlando, Fla, USA, March 2012.
  36. A. R. Partovi, A. Z. Y. Kevin, H. Lin, B. M. Chen, and G. Cai, “Development of a cross style quadrotor,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, St. Paul, Minn, USA, August 2012.
  37. E. Altu ˘g, J. P. Ostrowski, and R. Mahony, “Control of a quadrotor helicopter using visual feedback,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 72–77, IEEE, Washington, DC, USA, May 2002.
  38. D. S. Miller, *Open loop system identification of a micro quadrotor helicopter from closed loop data [M.S. dissertation]*, University of Maryland, College Park, Md, USA, 2011.
  39. J. Dvorak, *Micro quadrotor-design, modeling, identification and control [M.S. thesis]*, Czech Technical University, Prague, Czech Republic, 2011.
  40. P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a large quadrotor robot,” *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.
  41. J. H. Kim, M.-S. Kang, and S. Park, “Accurate modeling and robust

- hovering control for a quad-rotor VTOL aircraft,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1–4, pp. 9–26, 2010.
- 42. M. Elsamanty, A. Khalifa, M. Fanni, A. Ramadan, and A. Abo-Ismail, “Methodology for identifying quadrotor parameters, attitude estimation and control,” in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM ‘13)*, pp. 1343–1348, Wollongong, Australia, July 2013.
  - 43. J. Kim, M.-S. Kang, and S. Park, “Accurate modeling and robust hovering control for a quad-rotor VTOL aircraft,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1–4, pp. 9–26, 2010.
  - 44. T. Bresciani, *Modelling, identification and control of a quadrotor Helicopter [M.S. thesis]*, Lund University, Lund, Sweden, 2008.
  - 45. H. J. Wang, “The essence of dynamic mechanics for lagrange equation,” *Journal of Hebei University of Science and Technology*, vol. 24, no. 2, pp. 56–60, 2003 (Chinese).
  - 46. P. Castillo, R. Lozano, and A. Dzul, “Stabilization of a mini-rotorcraft having four rotors,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ‘04)*, pp. 2693–2698, Sendai, Japan, October 2004.
  - 47. R. W. Beard, *Quadrotor Dynamics and Control*, ‘Brigham Young University, 2008.
  - 48. P. McKerrow, “Modelling the draganflyer four-rotor helicopter,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3596–3601, New Orleans, La, USA, May 2004.
  - 49. M. D. L. C. de Oliveira, *Modeling, Identification and Control of a Quadrotor Aircraft [M.S. thesis]*, Czech Technical University, Prague, Czech Republic, 2011.
  - 50. S. Bouabdallah, P. Murrieri, and R. Siegwart, “Design and control of an indoor micro quadrotor,” in *Proceeding of the 2004 IEEE International Conference on Robotics and Automation (ICRA ‘04)*, vol. 5, pp. 4393–4398, May 2004.
  - 51. A. Benallegue, A. Mokhtari, and L. Fridman, “Feedback linearization and high order sliding mode observer for a quadrotor UAV,” in *Proceedings of the International Workshop on Variable Structure Systems (VSS ‘06)*, pp. 365–372, Alghero, Italy, June 2006.
  - 52. S. Bouabdallah and R. Siegwart, “Full control of a quadrotor,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent*

- Robots and Systems (IROS '07)*, pp. 153–158, San Diego, Calif, USA, November 2007.
- 53. C. A. Herda, *Implementation of a quadrotor unmanned aerial vehicle [M.S. dissertation]*, California State University, Los Angeles, Calif, USA, 2012.
  - 54. J. M. B. Domingues, *Quadrotor prototype [M. S. dissertation]*, Instituto Superior Tecnico, Göttingen, Germany, 2009.
  - 55. A. Tayebi and S. McGilvray, “Attitude stabilization of a VTOL quadrotor aircraft,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 562–571, 2006.
  - 56. N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, “Rigid-body attitude control,” *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 30–51, 2011. | MathSciNet
  - 57. V. Mistler, A. Benallegue, and N. K. M’Sirdi, “Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback,” in *Proceedings of the 10th IEEE International Workshop on Robot and Human Communication*, pp. 586–593, Paris, France, September 2001.
  - 58. J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, “Real-time indoor autonomous vehicle test environment,” *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008. | MathSciNet
  - 59. M. J. Stepaniak, *A quadrotor sensor platform [Ph.D. dissertation]*, Ohio University, Athens, Ohio, USA, 2008.
  - 60. A. Nanjangud, “Simultaneous low-order control of a nonlinear quadrotor model at four equilibria,” in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2914–2919, Florence, Italy, December 2013.
  - 61. R. K. Agarwal, *Recent Advances in Aircraft Technology*, InTech, Rijeka, Croatia, 2012.
  - 62. G. M. Hoffmann, H. M. Huang, S. L. Waslander, and C. J. Tomlin, “Quadrotor helicopter flight dynamics and control- theory and experiment,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6461, Hilton Head, SC, USA, 2007.
  - 63. P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. Roberts, “Towards dynamically favourable quad-rotor aerial robots,” in *Proceedings of the Australasian Conference on Robotics and Automation*, Canberra, Australia, December 2004.
  - 64. G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, “Quadrotor

- helicopter trajectory tracking control,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, USA, August 2008.
- 65. G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Precision flight control for a multi-vehicle quadrotor helicopter testbed,” *Control Engineering Practice*, vol. 19, no. 9, pp. 1023–1036, 2011.
  - 66. P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a quadRotor robot,” in *Proceedings to International Conference on Robotics and Automation*, Auckland, New Zealand, 2006.
  - 67. R. W. Prouty, *Helicopter Performance, Stability and Control*, Krieger, Melbourne, Fla, USA, 1995.
  - 68. J. Seddon, *Basic Helicopter Aerodynamics*, BSP, London, UK, 1990.
  - 69. J. G. Leishman, *Principles of Helicopter Aerodynamics*, Cambridge University Press, London, UK, 2nd edition, 2006.
  - 70. R. Zawiski and M. Błachuta, “Dynamics and optimal control of quadrotor platform,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA 2012-4915, Minneapolis, Minn, USA, August 2012.
  - 71. S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying [M.S. thesis]*, Université Abou Bekr Belkaïd Tlemcen, Tlemcen, Algeria, 2007.
  - 72. C. Powers, D. Mellinger, and A. Kushleyev, “Influence of aerodynamics and proximity effects in quadrotor flight,” in *Proceedings of the International Symposium on Experimental Robotics*, pp. 17–21, Quebec, Canada, 2012.
  - 73. N. Guenard, T. Hamel, and L. Eck, “Control laws for the tele operation of an unmanned aerial vehicle known as an X4-flyer,” in *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, pp. 3249–3254, Beijing, China, October 2006.
  - 74. W. Johnson, *Helicopter Theory*, Dover, New York, NY, USA, 1980.
  - 75. L. Ljung, “Perspectives on system identification,” *Journal of Intelligent and Robotic Systems*, no. 74, pp. 129–145, 2014.
  - 76. M. Gevers, “A personal view of the development of system identification,” *IEEE Control Systems Magazine*, vol. 26, no. 6, pp. 93–105, 2006.
  - 77. M. Bergamasco, “Identification of linear models for the dynamics

- of a hovering quadrotor,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1696–1707, 2014.
- 78. L. K. Burkamshaw, *Towards a low-cost quadrotor research platform [M.S. thesis]*, Naval Postgraduate School, Monterey, Calif, USA, 2010.
  - 79. D. Sonntag, *A study of quadrotor modeling [M. S. dissertation]*, Linkopings Universitet, Linköping, Sweden, 2011.
  - 80. L. Derafa, T. Madani, and A. Benallegue, “Dynamic modelling and experimental identification of four rotors helicopter parameters,” in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT ‘06)*, pp. 1834–1839, Mumbai, India, December 2006.
  - 81. P. Pounds, R. Mahony, and P. Corke, “System identification and control of an aerobot drive system,” in *Proceedings of the Information, Decision and Control Conference (IDC ‘07)*, pp. 154–159, Adelaide, Australia, February 2007.
  - 82. S. Bouabdallah, P. Murrieri, and R. Siegwart, “Towards autonomous indoor micro VTOL,” *Autonomous Robots*, vol. 18, no. 2, pp. 171–183, 2005.
  - 83. Y. Naidoo, R. Stopforth, and G. Bright, “Quad-rotor unmanned aerial vehicle helicopter modeling and control,” *International Journal of Advanced Robotic Systems*, vol. 8, no. 4, pp. 139–149, 2011.
  - 84. L. Derafa, T. Madani, and A. Benallegue, “Dynamic modelling and experimental identification of four rotors helicopter parameters,” in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT ‘06)*, pp. 1834–1839, December 2006.
  - 85. O. Falkenberg, J. Witt, U. Pilz, U. Weltin, and H. Werner, “Model identification and  $H^\infty$  control for MAV’s,” in *Proceedings of the International Conference on Intelligent Robotics and Applications*, pp. 460–471, Montreal, Canada, 2012.
  - 86. G. Gremillion, “System identification of a quadrotor micro air vehicle,” in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference*, p. 7644, Toronto, Canada, 2010.
  - 87. W. Wei, “Frequency-domain system identification and simulation of a quadrotor controller,” in *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, pp. 1834–1839, National Harbor, Maryland, Md, USA, 2014.
  - 88. N. Abas, A. Legowo, and R. Akmeliauwati, “Parameter identification

- of an autonomous quadrotor,” in *Proceeding of the 4th International Conference on Mechatronics (ICOM'11)*, pp. 1–8, Kuala Lumpur, Malaysia, May 2011.
- 89. L. L. Liu, *Research on the modeling and control to a quadrotor helicopter simulator [M.S. thesis]*, Central South University, Changsha, China, 2009.
  - 90. Q. L. Luo, *Application of RBF-ARX model-based predictive control on quad-rotor helicopter simulator [M. S. dissertation]*, Central South University, Changsha, China, 2012, (Chinese).
  - 91. S. Yin, X. W. Li, H. J. Gao, and O. Kaynak, “Data-based techniques focused on modern industry: an overview,” *IEEE Transactions on Industrial Electronics*, no. 99, pp. 1–11, 2014.
  - 92. S. Yin, S. X. Ding, X. C. Xie, and H. Luo, “A review on basic data-driven approaches for industrial process monitoring,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, 2014.
  - 93. S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, “A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process,” *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.
  - 94. S. Yin, G. Wang, and H. R. Karimi, “Data-driven design of robust fault detection system for wind turbines,” *Journal of Mechatronics*, vol. 24, no. 4, pp. 298–306, 2014.
  - 95. S. Yin, S. X. Ding, A. H. A. Sari, and H. Hao, “Data-driven monitoring for stochastic systems and its application on batch process,” *International Journal of Systems Science*, vol. 44, no. 7, pp. 1366–1376, 2013. | MathSciNet
  - 96. S. Yin, H. Luo, and S. X. Ding, “Real-time implementation of fault-tolerant control systems with performance optimization,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 5, pp. 2402–2411, 2014.
  - 97. W. C. Sun, H. J. Gao, and O. Kaynak, “Adaptive backstepping control for active suspension systems with hard constraints,” *IEEE Transactions on Mechatronics*, vol. 18, no. 3, pp. 1072–1079, 2013.
  - 98. W. C. Sun, Z. L. Zhao, and H. J. Gao, “Saturated adaptive robust control for active suspension systems,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 9, pp. 3889–3896, 2013.

# **CHAPTER**

# **13**

## **Visual Flight Control of a Quadrotor Using Bioinspired Motion Detector**

---

**Lei Zhang,<sup>1</sup> Tianguang Zhang,<sup>1</sup> Haiyan Wu,<sup>1</sup> Alexander Borst,<sup>2</sup> and Kolja Kühnlenz<sup>1,3</sup>**

<sup>1</sup>Institute of Automatic Control Engineering (LSR), Technische Universität München, 80290 München, Germany

<sup>2</sup>Department of Systems and Computational Neurobiology, Max Planck Institute of Neurobiology, Am Klopferspitz 18, D-82152 Martinsried, Germany

<sup>3</sup>Institute for Advanced Study (IAS), Technische Universität München, 80290 München, Germany

---

**Citation:** Lei Zhang, Tianguang Zhang, Haiyan Wu, Alexander Borst, and Kolja Kühnlenz: "Visual Flight Control of a Quadrotor Using Bioinspired Motion Detector", International Journal on Navigation and Observation, Volume 2012 |Article ID 627079, 9 pages, <https://doi.org/10.1155/2012/627079>.

**Copyright:** © 2012 Lei Zhang et al. This is an open access article distributed under the Creative Commons Attribution License

## ABSTRACT

Motion detection in the fly is extremely fast with low computational requirements. Inspired from the fly's vision system, we focus on a real-time flight control on a miniquadrotor with fast visual feedback. In this work, an elaborated elementary motion detector (EMD) is utilized to detect local optical flow. Combined with novel receptive field templates, the yaw rate of the quadrotor is estimated through a lookup table established with this bioinspired visual sensor. A closed-loop control system with the feedback of yaw rate estimated by EMD is designed. With the motion of the other degrees of freedom stabilized by a camera tracking system, the yaw-rate of the quadrotor during hovering is controlled based on EMD feedback under real-world scenario. The control performance of the proposed approach is compared with that of conventional approach. The experimental results demonstrate the effectiveness of utilizing EMD for quadrotor control.

## INTRODUCTION

Flying insects have tiny brains and mostly possess compound eyes which can get panoramic scene to provide an excellent flying performance. Comparing with state-of-the-art artificial visual sensors, the optics of compound eye provide very low spatial resolution. Nevertheless, the behavior of flying insects is mainly dominated by visual control. They use visual feedback to stabilize flight [1], control flight speed, [2] and measure self-motion [3]. On the other hand, highly accurate real-time stabilization and navigation of unmanned aerial vehicles (UAVs) or microaerial vehicles (MAVs) is becoming a major research interest, as these flying systems have significant value in surveillance, security, search, and rescue missions. Thus, the implementation of a bio-plausible computation for visual systems could be an accessible method to replace the traditional image processing algorithms in controlling flying robots such as a quadrotor.

Most of early applications using insect-inspired motion detector focus on motion detection tasks rather than velocity estimation. In robotics and automation applications, EMDs are mainly used for a qualitative interpretation of video image sequence, to provide general motion information such as orientation and infront obstacles. In [4], a microflyer with an onboard lightweight camera is developed, which is able to fly indoor while avoiding obstacles by detecting certain changes in optic flow. The recent approach for the navigation in a corridor environment on an autonomous quadrotor by using optical flow integration is shown in [5].

Another example is a tethered optic flow-based helicopter that mimics insect behaviors such as taking off, cruise, and landing [6, 7], and in [8] the EMDs visual sensors were tested and characterized in field experiments under various lighting conditions. Numerous authors have pointed out that the Reichardt model, while sensitive to motion, does not measure velocity [9–11]. However, some efforts have been made, examining the possibility of velocity estimation tasks by introducing elaborated models [12, 13]. In [14], yaw rate estimates on a coaxial helicopter testbed are obtained using a matched filter approach which yet incorporates a virtual 3D environment in the control loop. Although a lot of work has been done on a simulation level or involving simulation tools, further robotic applications with EMDs considering closed-loop velocity control are still to be investigated under real-world scenarios.

In this paper, a quadrotor system with bioinspired visual sensor is described. The novel image processing methods and the control laws are implemented in real-time experiments. The yaw rate control is totally based on the visual feedback of the on-board camera. The reference velocity value is provided by the on-board inertial measurement unit (IMU). For a 6-DOF (degrees of freedom) flying robot control, a tracking system of multicamera configuration is also utilized to achieve the altitude and attitude stabilization near hover. Due to the noise in real-world, the velocity estimation tasks would be more challenging. In this work, the approach of building an empirical lookup table from open-loop test results is introduced for this task. The Reichardt motion detector is modified which describes, at an algorithmic level, the process of local motion detection in flying behaviors. Certain patterns of receptive fields, which respond to particular optic flow, are utilized to estimate the global ego-motion through the environment. Another main issue which needs to be tackled carefully in this work is that the flying robot should be well stabilized during hovering. By multicamera tracking, the absolute position as well as the pose is determined from the positions of four on-board markers.

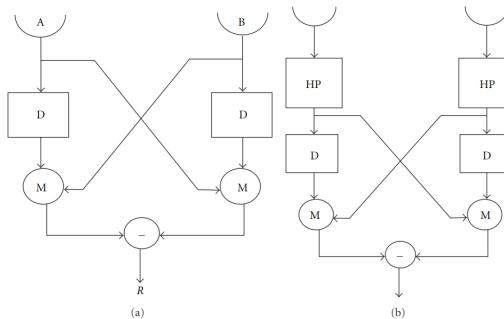
The remainder of this paper is organized as follows: in Section 2, we firstly introduce the bioinspired visual image processing methods used in this work. In Section 3, the 3D pose estimation using visual tracking system is described. The control strategy of the system as well as the software structure of algorithms is presented in Section 4. Then in Section 5, an overview of the whole experimental platform is illustrated. The control performance is also evaluated based on the experimental results in this section. Conclusions are given in Section 6, with directions on future works.

## BIOINSPIRED IMAGE PROCESSING

In this section, we introduce the essential part of this work: using biological models for yaw rate estimation of a quadrotor. The EMDs are utilized for this task. The whole methodology is introduced in detail. To achieve the yaw rate control, the system also requires accurate visual tracking for pose stabilization (Section 3) and efficient controllers (Section 4).

In an insect's perspective, motion information has to be computed from the changing retinal images by the nervous system [15]. For engineering applications, some properties of the biological visual system are converted into computational algorithms.

The elaborated EMD model used in this work is a modified model of the famous Reichardt motion detector [16]. The original Reichardt motion detector (Figure 1(a)) has only low-pass filters and two correlations. In this work, a temporal high-pass filter is added before the low-pass filter to obtain a simple response to step edges [17] (Figure 1(b)). The high-pass filters and low-pass filters in this model are all designed to be of first order.



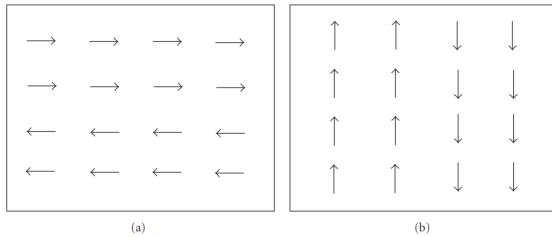
**Figure 1:** The simple Reichardt motion detector (a) and the elaborated EMD model (b). A and B are two visual signal inputs. The moving luminance signal is observed by this pair of visual sensors (such as the ommatidia of a fruit fly). The detector compares visual sensor inputs and generates a direction sensitive response R corresponding to the visual motion. D: delay block. Here it refers to low-pass filter; HP: High-pass filter; M: multiplication.

In [12], a mathematical analysis of the original Reichardt motion detector is given regarding the response to different images (sinusoidal gratings as well as natural images). Without loss of generality, we firstly consider the response of this modified model to a moving natural image (which possesses energy at all spatial frequencies). Similar to the response of the simplified model [12], for this modified model, the output is

$$R = \int_0^{\infty} \frac{2\pi\tau_L f_s v \sin(2\pi f_s \Delta\phi)}{1 + (\tau_L/\tau_H)^2 + (1/2\pi f_s v \tau_H)^2 + (2\pi f_s v \tau_L)^2} P(f_s) df_s, \quad (1)$$

where  $\Delta\phi$  is the angular displacement between the two vision sensors,  $f_s$  is the spatial frequency of the image input to the detector,  $v$  stands for the velocity of the moving image,  $\tau_L$  and  $\tau_H$  are the time constants of the low- and high-pass filters, respectively, and  $P(f_s)$  represents the power spectral density. So according to (1), the local motion information is calculated.

To obtain a global ego-motion estimation, certain receptive fields of the motion-sensitive widefield neurons in the fly brain are applied. Considering the specified experimental scenario in this work, two novel templates of receptive fields for rotation detection are utilized (Figure 2), which are proposed in [17].



**Figure 2:** Rotation templates: horizontal (a) and vertical (b). The response of horizontal rotation is calculated by subtracting the upper part and lower part of the template of receptive fields, while the vertical rotation is detected by the difference between left part and right part of the global optical flow.

The algorithms for calculating the rotation global response can be described as (image size: length  $\times$  width; RH: response of local horizontal motion;  $R_V$ : response of local vertical motion):

$$\begin{aligned} \text{Horizontal rotation} &= \sum_{i=1}^{\text{width}/2} \sum_{j=1}^{\text{length}} R_H - \sum_{i=\text{width}/2+1}^{\text{width}} \sum_{j=1}^{\text{length}} R_H, \\ \text{Vertical rotation} &= \sum_{i=1}^{\text{width}} \sum_{j=1}^{\text{length}/2} R_V - \sum_{i=1}^{\text{width}} \sum_{j=\text{length}/2+1}^{\text{length}} R_V. \end{aligned} \quad (2)$$

Now we examine the feasibility of using this model for velocity estimation tasks. In [12], two criteria are quantified for an accurate velocity estimation system: (1) image motion at a fixed velocity should always have

approximately response; (2) at a given velocity, the response to motion should be unambiguous over certain range. In simulation, we find that by introducing this modified model, the response to a specific velocity of image motion can meet the two basic requirements at low velocities (above which the response output is ambiguous). Thus, for velocity estimation tasks, the motion velocity should be limited in a certain range due to essential property (bell-shaped response) of the Reichardt model. In order to reduce the brightness sensitivity, logarithmic transformation could be also applied (as the modified model in [17]). However, by doing this the discrimination of response is also highly reduced. That means, the response at a given velocity cannot differ significantly from the response at other velocities, which is not appropriate for quantifying velocity. Moreover, regarding the brightness sensitivity problem, a stable lighting condition is demanded in real-time experiments.

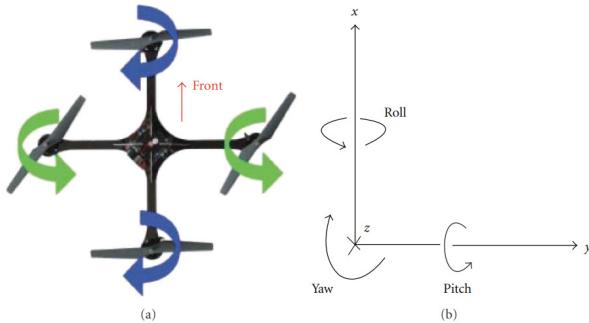
We firstly examine the open-loop characteristics of the system only for yaw rate estimation. The quadrotor is tethered in the air, with an on-board camera looking directly to the ground texture (the complete system is further introduced in Section 5). This scenario in the indoor environment involves a black-white chessboard ground texture. It is considered to be the most suitable scenario for detecting rotation motion of a flying robot. Compared to other forms of textures, the high image contrast can also help to improve the discrimination for quantifying velocity (due to the characteristics of the biological model itself). The quadrotor is rotated on horizontal level without control, and we can get the relationship between the response output and the rotation velocity (yaw rate). A lookup table is then built. Due to the system noise and discrimination limitations of the experiments, the curve has some nonmonotonic regions. The polynomial minimum quadric method is used to fit the curve, (where  $d_i$  is a residual which is defined as the difference between the predicted value  $y_i$  and the actual value  $f(x_i)$ ):

$$\sum_{i=1}^n d_i^2 = \sum_{i=1}^n [y_i - f(x_i)]^2 = \text{minimum.} \quad (3)$$

## MULTICAMERA 3D POSE ESTIMATION

Quadrotor is an underactuated vehicle. The 6 DOFs of the quadrotor are controlled by four motor inputs (pitch, roll, thrust, and yaw) by varying the lift forces and the torque balance through changing the rotating speed of the rotors (see Figure 3(a)). Yaw control is realized by tuning the differential

speed between the two counterrotating rotor pairs. Increasing the rotating speed of all the four motors at the same amount will cause an upward movement of the quadrotor. When tuning the differential speed between the two motors of either single rotor pair, the quadrotor will fly sideways.



**Figure 3:** Quadrotor dynamics. (a) the overhead view of a quadrotor; (b) six degrees of freedom of the flying robot.

The work in this section is based on the former related work in [18]. In this work, we set up an indoor GPS system by using multicamera tracking instead of the former two-camera tracking. By tracking the four markers installed on the axis of the quadrotor, the 3D position as well as the pose of the flying robot can be estimated. The experimental setup of 3D tracking is further introduced in Section 5. The frame of quadrotor dynamics is the same as the in Figure 3(b). We have the following definition:

Marker i position vector:  $S_i = (x_i, y_i, z_i)^T$  ( $i = 1, 2, 3, 4$ ).

Central point vector between two nonadjacent markers:  $M_j = (x_{Mj}, y_{Mj}, z_{Mj})^T$  ( $j = 1, 2$ );

Estimated central point vector of the quadrotor:  $M_q = (x_M, y_M, z_M)^T$ ;

Orientation of marker i:  $V_i = (x_{Vi}, y_{Vi}, z_{Vi})^T$  ( $i = 1, 2, 3, 4$ );

The counting of the markers is clockwise, while the first marker is on the main axis. For 3D pose control, the central point should be used as the reference position of the quadrotor. The central points of the distance between marker 1 and marker 3 as well as between marker 2 and marker 4 are  $M_1 = (1/2)(S_1 + S_3)$  and  $M_2 = (1/2)(S_2 + S_4)$ . In consequence of the marker's noise through the tracking system, the two central points in the two equations above are not identical. Thus, the central point of the quadrotor is  $M_q = (1/2)(M_1 + M_2)$ . The vectors between the central point and marker 1 for pitch as well as between the central point and marker 2 for roll are  $V_i =$

$S_i - Mq$  ( $i = 1, 2$ ). The values of pitch  $\theta$ , roll  $\phi$ , and yaw  $\psi$  angles can be then calculated, and thus the 3D pose of the quadrotor can be estimated:

$$\theta = \text{sgn}(z_{v_1}) \arccos \left( \frac{\sqrt{x_{v_1}^2 + y_{v_1}^2}}{\sqrt{x_{v_1}^2 + y_{v_1}^2 + z_{v_1}^2}} \right), \quad (4)$$

$$\phi = \text{sgn}(z_{v_2}) \arccos \left( \frac{\sqrt{x_{v_2}^2 + y_{v_2}^2}}{\sqrt{x_{v_2}^2 + y_{v_2}^2 + z_{v_2}^2}} \right), \quad (5)$$

$$\psi = -\text{sgn}(y_{v_1}) \arccos \left( \frac{x_{v_1}}{\sqrt{x_{v_1}^2 + y_{v_1}^2}} \right). \quad (6)$$

## CONTROLLER

At first the quadrotor should be regulated to hover in the air on horizontal plane with little shaking. That means, the stable state commands ( $u_p^s$  for pitch,  $u_r^s$  for roll,  $u_y^s$  for  $y_w^a$ , and  $u_t^s$  for thrust) should be adjusted firstly. Basing on these parameters, the control commands can be calculated next. For each controller, we have an output value ( $u_p^q$  for pitch,  $u_r^q$  for roll,  $u_y^q$  for yaw, and  $u_t^q$  for thrust) between  $-1$  and  $1$ , which is then added with the corresponding stable state command. Since we only consider the rotation movement in this experiment, which means, the quadrotor is not always heading with the main axis towards X direction, the pitch and roll commands should be adjusted in the  $\psi$  direction with a rotation matrix:

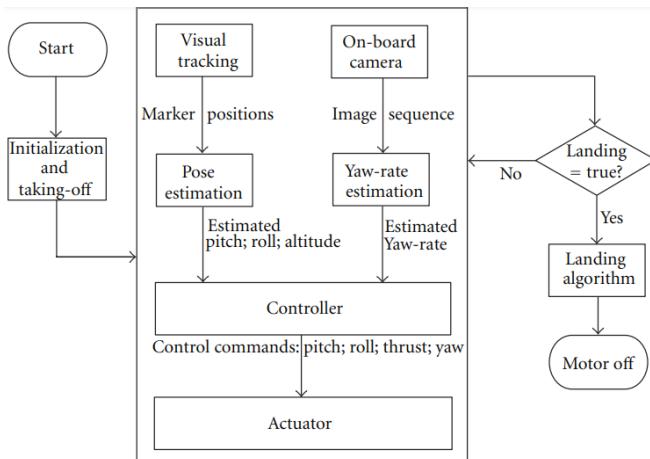
$$\begin{pmatrix} u_p \\ u_r \\ u_t \\ u_y \end{pmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 & 0 \\ \sin \psi & \cos \psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} u_p^q \\ u_r^q \\ u_t^q \\ u_y^q \end{pmatrix} + \begin{pmatrix} u_p^s \\ u_r^s \\ u_t^s \\ u_y^s \end{pmatrix}. \quad (7)$$

We choose proportional-integral (PI) controller for the yaw rate control. The pitch, roll, and thrust commands are controlled by proportional-integral-derivative (PID) controllers:

$$\begin{aligned}
u_y^q &= k_y^P(\dot{\psi}_0 - \dot{\psi}) + k_y^I \int (\dot{\psi}_0 - \dot{\psi}) dt, \\
u_p^q &= k_p^P(\theta_0 - \theta) + k_p^I \int (\theta_0 - \theta) dt + k_p^D \frac{d}{dt}(\theta_0 - \theta), \\
u_r^q &= k_r^P(\phi_0 - \phi) + k_r^I \int (\phi_0 - \phi) dt + k_r^D \frac{d}{dt}(\phi_0 - \phi), \\
u_t^q &= k_t^P(z_0 - z) + k_t^I \int (z_0 - z) dt + k_t^D \frac{d}{dt}(z_0 - z).
\end{aligned} \tag{8}$$

In this experiment, the reference values  $\theta_0$  and  $\phi_0$  are set to zero. The desired altitude  $z_0$  is 0.35 m near hover. The measured values  $\theta$  and  $\phi$  are calculated from the received data of the visual multicamera tracking system using (4) and (5), whereas  $\psi$  is searched out from a certain empirical lookup table using the response value, which is calculated by insect-inspired motion detectors. The yaw velocity can also be obtained from (6) by time derivative, which is, for the heading stabilization, used as a reference (ground truth). The closed-loop results will be shown and further discussed in Section 5.

The main loop in the whole software architecture (Figure 4) consists of two simultaneous processes: yaw rate control using on-board camera visual feedback and X, Y, and Z position/poses control using visual tracking system. A graphical user interface (GUI) is developed basing on Qt crossplatform application, which provides data visualization (e.g., 3D trajectory of the quadrotor, battery voltage and sensory data information), commands input and real-time online configuration of control parameters.

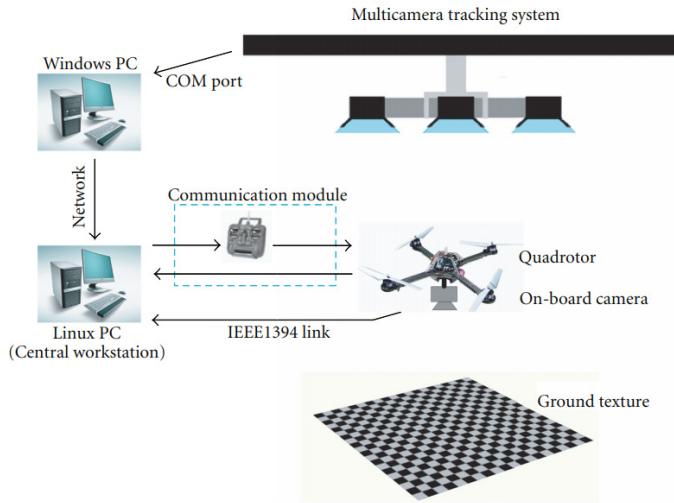


**Figure 4:** Software architecture.

## EXPERIMENTS AND RESULTS

### Experimental Setup

The whole experimental platform is shown in Figure 5. It mainly consists of a quadrotor testbed, off-board workstation, and video camera tracking system.



**Figure 5:** Experimental setup.

#### 1) Quadrotor

The miniquadrotor used in this work is a “Hummingbird” with an “AutoPilot” central control board from ascending technologies. It offers a 1 kHz control frequency and motor update rate, which guarantees fast response to changes in the environment. The size of the whole quadrotor testbed is 36.5 cm in diameter, and the four rotors (each with a propeller size of 19.8 cm) are directly driven by four high-torque DC brushless motors respectively. Powered by a state-of-the-art 3-cell 2100mAh lithium-polymer battery, the vehicle is able to hover up to 15 minutes (with about 120 g of payload in this work).

#### 2) On-Board Camera

Considering the limited payloads of the quadrotor, the PointGrey Firefly MV CMOS camera which has a light weight (14 g) and a tiny size (25 mm × 40 mm) is selected as the on-board camera. We choose a standard resolution of 640 × 480 (pixels), and the frame rate is 60 Hz. The camera is equipped

with a 6 mm microlens, providing a viewing angle of 56 deg and 38 deg in the length and width directions, respectively. It uses a 5-pin USB 2.0 digital interface with a 480 Mb/s transfer rate and 8-bit raw Bayer data format (connected through IEEE 1394 to workstation). The camera is mounted under the base board of the quadrotor, looking directly down to the ground texture.

### *3) Workstation and Communication Module*

An off-board Linux PC (AMD Athlon 5200+; 2 GB RAM) is used for image data processing, 3D pose estimation and control law execution in this case. The quadrotor is equipped with XBeePro wireless communication module from MaxStream/Digi, which enables the data transmission from the on-board inertial measurement unit (IMU) and the control command reception (with R/C transmitter enabled) from workstation at a rate of 100 Hz.

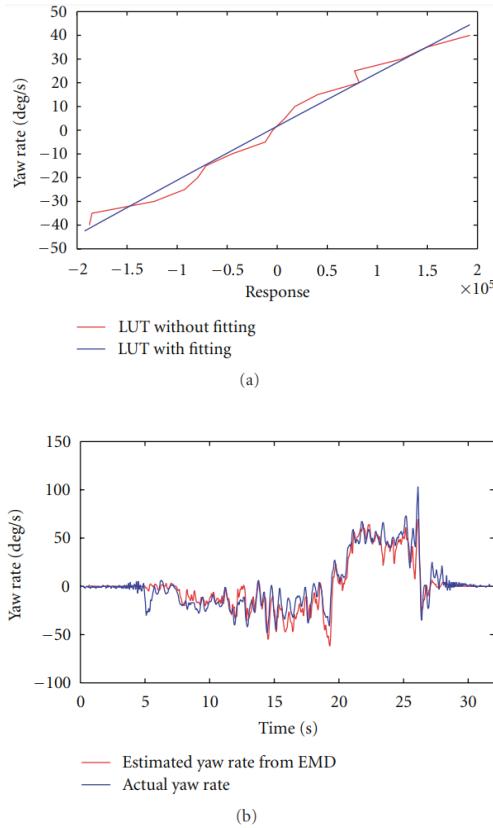
### *4) Visual Tracking System and Marker Placement*

The tracking system VisualeyezII VZ4000 from Phoenix Technologies Incorporated is used to get the absolute position of the quadrotor. It has three cameras inside which can capture the certain markers installed on the four axes of the quadrotor in an accuracy of millimeter level. In this work, the tracking system is installed on the ceiling of the lab (Figure 5). The software VZSoft is installed in another Windows PC (AMD Athlon XP 3000+; 2.1 GHz). It gets the data from the tracking system through a COM interface. The data will be then sent to the workstation with the interface Babelfish which is developed by the Institute of Autonomic Control Engineering, Technical University Munich, using Internet Communications Engine (ICE).

## **Velocity Estimation**

To validate the designed templates of receptive fields for rotation detection, the bioinspired image processing algorithm is implemented with C++ language using Open CV.

Under low velocities and within certain altitude range, the response can be regarded as monotonic and near linear from the test results. In this work, the yaw rate is under 100 deg/s and the altitude value is set to 0.35 m. The lookup table is shown in Figure 6(a), with a polynomial curve fitting. From the comparison in Figure 6(b), this approach provides a fairly accurate yaw rate estimation (the mean error is 1.85 deg/s and the standard deviation of error is 10.22 deg/s). This lookup table could be then used in the closed-loop control under the same light condition.

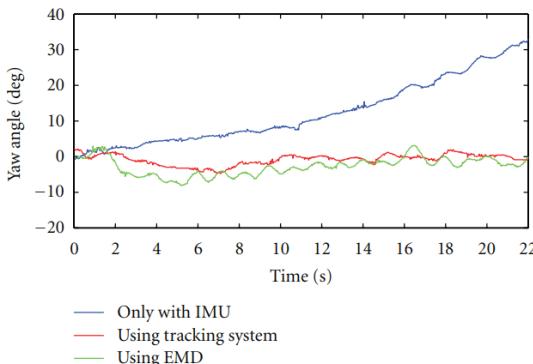


**Figure 6:** (a) lookup Table (LUT) in this work. This empirical LUT shows the relationship between yaw rate (in low speed) and visual response under the experimental environment; (b) open-loop characteristics for rotation motion near hover (without control).

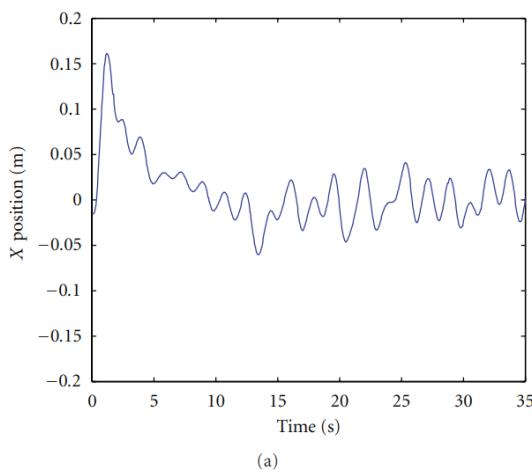
## Heading Stabilization

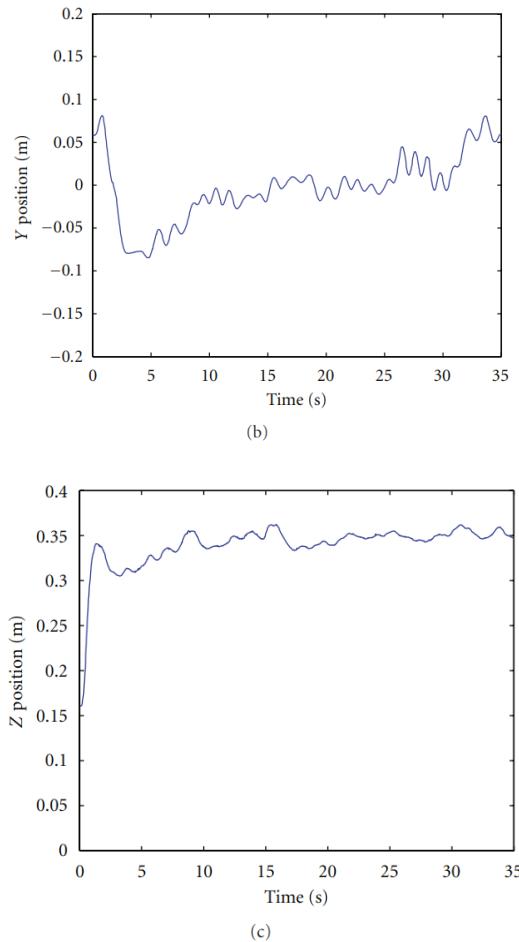
In this experiment, we compare the heading control performance using EMD with those using IMU or tracking system respectively. At first the stable commands ( $u_p^s$ ,  $u_r^s$ ,  $u_y^s$ , and  $u_t^s$ ) should be determined experimentally, so that without any controllers off board, the quadrotor can be hovering in the air nearly on a horizontal level and rotating as little as possible, with all the payloads mounted (in this experiment, with on-board bread board for tracking system using TCM8 mode, and with cable power supply instead of battery). The X, Y, and Z positions should be further controlled using the feedback from the tracking system, while the yaw position has no controllers

except the on-board IMU at the first attempt. The IMU controller is already integrated on the base board so that no other off-board controller is needed for IMU controlling. A major disadvantage of using IMUs for navigation is that they typically suffer from accumulated error (see Figure 7 blue curve). In this case, in about 20 seconds the yaw position will deviate by 30 degrees if only IMU is used for the heading stabilization. The second reference is the yaw position with the control using tracking system, but without using EMD (the red curve in Figure 7). In Figure 8, the 3D position when using EMD for heading stabilization is shown. By using tracking system and EMDs (the green curve in Figure 7) a satisfying performance could be both achieved. Despite some deviation (for tracking system maximal  $\pm 5$  degrees and for EMD maximal  $\pm 7$  degrees), the quadrotor can hover very well with straight heading direction.



**Figure 7:** Heading stabilization.



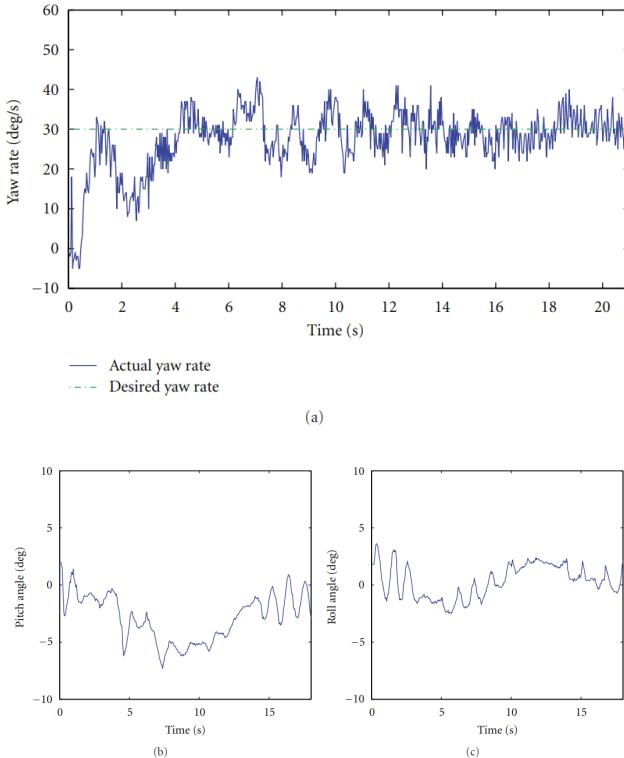


**Figure 8:** 3D position measurements when the quadrotor is in stabilized hover ((a), (b), and (c) for X, Y, and Z positions, resp.). The 3D position is stabilized by multicamera tracking system exclusively. The X and Y positions are set to zero, which means the on-board camera is heading directly to the central point of the ground texture as well as the origin of world reference coordinate. The altitude (Z position) is set to 0.35 m.

## Yaw Rate Control

The next step is to achieve the velocity control using EMDs. The yaw rate should be set in a low-speed area considering a monotonic relationship between response and velocity. In this case, the desired velocity is 30 degrees/s. The results are shown in Figure 9(a). The settling time is about

4 seconds and the maximal error is about  $\pm 10$  degrees/s. The inflight performance of 3D pose is shown in Figures 9(b) and 9(c). Since the IMU provides only the angular velocity values, the angle positions are integrated by the base control board on the quadrotor in order to get the angle positions, which are sent to the central workstation.



**Figure 9:** Attitude measurements of the flying robot near hover with yaw-rotation (a): yaw rate control. After takeoff, the quadrotor is switched to a desired yaw rate of 30 deg/s; (b) and (c): pitch and roll angles of the quadrotor.

For velocity estimation, although the EMD is not a pure velocity detector, a closed-loop control of yaw rate is achieved with restrictions of the structured environment and the limitation of velocity in low-speed area. Including image translation delay through IEEE1394/USB cable, image processing by CPU costs 10–20 ms (the program takes nearly 10 ms to wait for images from the camera, while the actual computing time is only several milliseconds). So the EMD computing is extremely fast, which provides an evidence of the efficiency when using biological models.

## CONCLUSIONS AND FUTURE WORKS

In this work, the closed-loop control of a flying robot is achieved by using bioinspired image processing method, which proves to be an effective approach with low computational cost. For real-time implementation, the experimental results of heading stabilization show that, by using EMD response as a feedback, the accumulating drift from the on-board IMU is compensated. Another trial regarding the EMDs as a velocity sensor has realized a low-speed control of the yaw rate on the quadrotor in real-world scenario. In the future, all the 6 DOFs should be controlled by using bioinspired image processing exclusively, without relying on any off-board visual sensors or GPS. The absolute position should be determined by certain advanced algorithms. For the future works, some efforts should be put in the development of novel approaches for highly robust flying performance.

## ACKNOWLEDGMENTS

This work is supported in part by the DFG excellence initiative research cluster Cognition for Technical Systems-CoTeSys, see also [www.cotesys.org](http://www.cotesys.org), the Bernstein Center for Computational Neuroscience Munich, see also <http://www.bccn-munich.de/>, and the Institute for Advanced Study (IAS), Technische Universität München, see also <http://www.tum-ias.de/>.

## REFERENCES

1. M. Egelhaaf and A. Borst, "Motion computation and visual orientation in flies," *Comparative Biochemistry and Physiology A*, vol. 104, no. 4, pp. 659–673, 1993.
2. M. V. Srinivasan and S. W. Zhang, "Visual navigation in flying insects," *International Review of Neurobiology*, vol. 44, pp. 67–92, 2000.
3. H. G. Krapp and R. Hengstenberg, "Estimation of self-motion by optic flow processing in single visual interneurons," *Nature*, vol. 384, no. 6608, pp. 463–466, 1996.
4. A. Beyeler, J.-C. Zufferey, and D. Floreano, "3D vision-based navigation for indoor microflyers," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, pp. 1336–1341, April 2007.
5. J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous Robots*, vol. 27, no. 3, pp. 199–200, 2009.
6. F. Ruffier and N. Franceschini, "Optic flow regulation: the key to aircraft automatic guidance," *Robotics and Autonomous Systems*, vol. 50, no. 4, pp. 177–194, 2005.
7. N. Franceschini, F. Ruffier, and J. Serres, "A bio-inspired flying robot sheds light on insect piloting abilities," *Current Biology*, vol. 17, no. 4, pp. 329–335, 2007.
8. F. Expert, S. Viollet, and F. Ruffier, "Outdoor field performances of insect-based visual motion sensors," *Journal of Field Robotics*, vol. 28, no. 4, pp. 529–541, 2011.
9. E. Buchner, "Behavioural analysis of spatial vision in insects," in *Photoreception and Vision in Invertebrates*, M. A. Ali, Ed., pp. 561–621, Plenum Press, New York, NY, USA, 1984.
10. M. Egelhaaf and W. Reichardt, "Dynamic response properties of movement detectors: theoretical analysis and electrophysiological investigation in the visual system of the fly," *Biological Cybernetics*, vol. 56, no. 2-3, pp. 69–87, 1987.
11. M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett, "Honeybee navigation en route to the goal: visual flight control and odometry," *Journal of Experimental Biology*, vol. 199, no. 1, pp. 237–244, 1996.
12. R. O. Dror, D. C. O'Carroll, and S. B. Laughlin, "Accuracy of velocity

- estimation by Reichardt correlators,” *Journal of the Optical Society of America A*, vol. 18, no. 2, pp. 241–252, 2001.
- 13. S. Rajesh, D. O’Carroll, and D. Abbott, “Elaborated Reichardt correlator for velocity estimation tasks,” in *Proceedings of the Biomedical Applications of Micro- and Nanoengineering*, pp. 241–253, December 2002.
  - 14. S. Han, A. Straw, M. Dickinson, and R. Murray, “A real-time helicopter testbed for insect-inspired visual flight control,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.
  - 15. A. Borst, C. Reisenman, and J. Haag, “Adaptation of response transients in fly motion vision. II: model studies,” *Vision Research*, vol. 43, no. 11, pp. 1311–1324, 2003.
  - 16. B. Hassenstein and W. Reichardt, “Systemtheoretische Analyse der Zeit-Reihenfolgen, und Vorzeichenauswertung bei der Bewegungsperzeption des Ruesselkaefers Chlorophanus,” *Naturforsch*, vol. 11, pp. 513–524, 1956.
  - 17. T. Zhang, H. Wu, A. Borst, K. Kuehnlenz, and M. Buss, “An FPGA implementation of insect-inspired motion detector for high-speed vision systems,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, Calif, USA, May 2008.
  - 18. M. Achtelik, T. Zhang, K. Kuhnlenz, and M. Buss, “Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors,” in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Changchun, China, 2009.

# **SECTION 4**

# **APPLICATIONS OF UAV-S**

# **AND DRONES**



# **CHAPTER**

# **14**

## **Development of Rescue Material Transport UAV (Unmanned Aerial Vehicle)**

---

**Daeil Jo, Yongjin Kwon**

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon, South Korea

### **ABSTRACT**

Recently, the market for drones is growing rapidly. Commercial UAVs (Unmanned Aerial Vehicles, or drones) are increasingly being used for various purposes, such as geographic survey, rescue missions, inspection of industrial facilities, traffic monitoring and delivery of cargos and goods. In particular, the drones have great potential for life-saving operations. A

---

**Citation:** Jo, D. and Kwon, Y. (2017) "Development of Rescue Material Transport UAV (Unmanned Aerial Vehicle)". *World Journal of Engineering and Technology*, **5**, 720-729. doi: 10.4236/wjet.2017.54060.

**Copyright:** © 2017 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

missing person, for example, can be rapidly and effectively searched using a drone in comparison with the conventional search operations. However, there is no commercially available rescue UAV until now. The motivation for this study is to design an unmanned aerial vehicle capable of vertical takeoff and landing, while containing a high power propellant apparatus in order to lift a heavy cargo that contains rescue materials (such as water, food, and medicine). We used the EDF (Electric Ducted Fan) technology as opposed to the conventional motor and prop combination. The EDF can produce the power about three times higher than the motor-prop combination. This became suitable for transportation of rescue goods, and can be widely used in rescue operations in natural environments. Based on these results, the UAV for rescue material transport capable of heavy vertical takeoff and landing is developed, including airframe, flight control computer and GCS (ground control station).

**Keywords:** High-Powered Propellant, Vertical Take-Off and Landing, Rescue Drone, GCS (Ground Control Station), FCC (Flight Control Computer)

## INTRODUCTION

In recent years, the use of drones or UAVs (unmanned aerial vehicles) is increasingly getting popular around the world. Drones are utilized in aerial photography for both personal hobbies and commercial uses. In industrial sectors, drones are deployed in facility inspection, power line inspections, and monitoring of fire and flood purposes. In almost every imaginable part, the drones are being used [1] [2]. The rescue operation is in no exception. Due to its fast flying characteristics, which can cover a wide area in less time, drones are now being used by law enforcement agencies for search missing persons. The current rescue drones are typically equipped with thermal image cameras; hence it can detect the missing person from the surroundings, even though the missing person is not visually identifiable from the surrounding environment. However, there is no rescue drones that can carry the emergency materials, such as water, food, blanket, and medicine, and being capable of conducting rescue operations [3] [4] [5]. In order to do this, the drones must have a high power propellant technology that is different from the conventional motor and prop combination. Additionally, the rescue drone should be able to vertically takeoff and land, while carrying the rescue materials. Therefore, the motivation of this study is to develop a rescue drone that is capable of conducting a search and rescue operation with

onboard carbo capacity. Through this development, we design and produce the actual airframe and cargo handling device, set up the firmware SW of flight control computer, and run a flight stability test to validate the safe flying characteristics. We will actually build a structural drone that is still in the concept phase and prepare a technology base so that a large number of IPs can be created in the future.

Currently, various hobby and surveillance drones are being developed and commercially available in Korea, but no high power drones are developed yet, with the high load carrying capacity. Most hobby drones cannot carry enough load, and even the large size drone can only carry less than 1 or 2 Kilograms. The biggest agricultural drones can carry about 10 to 15 Kilograms of payload, but they are too large for police rescue operations [6] [7]. In this context, the drone should be small enough to fit in the police car and easy to handle. At the same time, the drone should be generating enough lifting power, so that it can carry the rescue materials for the stranded or missing person. This will complement the shortcomings of the conventional UAV that is lacking the power capacity. As the demand for UAV increases in the future, it will lay the foundations for developing various types of UAVs (courier, transportation, delivery, lifeguard, etc.) that can meet the customer demands.

## **DEVELOPMENT OF SOFTWARE AND HARDWARE**

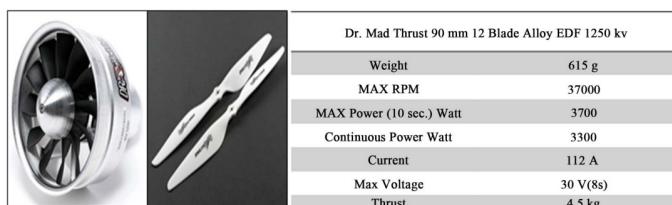
### **Motors and Propellers**

The drones that we want to make in this study basically have to deliver relief goods to people located in the middle of the building in case of fire and emergency situation in high-rise building. At this point, the most important point is that the drones must be close to the outer wall of the building to deliver relief supplies. In case of EDF (Electric Ducted Fan), it is possible to approach close to the building because the rotating prop is contained inside the turbine housing. However, when the conventional propeller motor is used, the drone is difficult to approach the building because the exposed propeller may hit the structure and the drone will fall from the sky. Ultimately, this difference is the biggest reason to use the EDF motors in developing the rescue drone [8] [9] [10]. There are also considerations when comparing EDF and propeller motors. It is whether or not you can protect the safety of your finger when you fly it. Due to the nature of the relief donor drone, the person must directly reach out and remove the relief

item from the box in the lower part of the drone. At this time, according to different rotation method of EDF motor and propeller motor, the safety protection of the finger is different. In other words, EDF drone do not have a risk of injuring your fingers unless you intentionally put your fingers into the turbine housing. However, in propeller motors, the propeller itself is located outside the drones, so there is a risk that the fingers can be cut any time. This means that if a propeller motor is used, the drones used for the rescue may cause secondary damage. Because of this, we have chosen a high thrust EDF, instead of conventional moto and prop combination. Electronic Ducted-Fan type airplane itself consists of engine, propeller, fixed wing and flap. Theoretically, moving the fluid flow from the rotating body to the duct increases the thrust efficiency by 41%. However, the weight of ducts to be increased in comparison with the propeller should also be considered. The aim of this study is to lift a total weight of about 10 kg, which has four EDFs, each of which can maintain the duration of 50% of the maximum thrust at a weight of up to 18 kg, as shown in Figure 1.

## Battery Selection

The battery is a major factor in controlling the drones' flight time and output. The larger the capacity of the battery, the longer the flight time, and the higher the number of cells in a series connected battery. The selected EDF requires a higher voltage than the propeller and motor combination, so we select a battery with 6 - 8 cells. The battery has a high stability and uses a lithium ion polymer material. This is because the electrolyte of the lithium ion battery is a liquid and basically has a problem in stability. We also considered trade-offs in battery capacity and weight. Electric vehicles' biggest dilemma is the proportional relationship between the capacity and weight. When a heavy battery is used in a



| Dr. Mad Thrust 90 mm 12 Blade Alloy EDF 1250 kv |          |
|---|----------|
| Weight  | 615 g    |
| MAX RPM   | 37000    |
| MAX Power (10 sec.) Watt                        | 3700     |
| Continuous Power Watt                           | 3300     |
| Current   | 112 A    |
| Max Voltage                                     | 30 V(8s) |
| Thrust  | 4.5 kg   |

**Figure 1.** Selected EDF: Highly safe, theoretically efficient EDF around the wing with Duct. Maximum thrust per EDF  $4.5 \text{ kg} \times 4 = 18 \text{ kg}$  Expectations, excluding frame and self weight-Estimated thrust at least  $8 \text{ kg}/900 \text{ g} \times 4 = 3.6 \text{ kg}$  (left), EDF Specification (right).

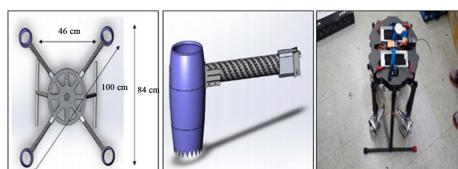
EDF with a duct, the weight of an automatic cardioverter defibrillator (AED), a fire mask, a lantern, and other rescue materials need to be reduced. Also, the weight of the drone itself can be increased significantly. The selected battery is PT-B10000-NSR35 model and has specifications of 22.2 V, 6S1P, 35C+, as shown in Figure 2. To meet the selected EDF output and minimum duration, the total capacitance that can be supplied by connecting two batteries in parallel was increased.

## Frame Design

Four EDFs are designed in four directions to balance the drones. Fixtures for installing ESC (electric speed controller), battery holder, GPS transceiver and other fixtures are installed on the center top plate. In addition, a fixture for installing a structural goods transport box is installed in the lower part, and two supports for stably loading and unloading the drones are installed. The material of the frame was made of carbon material to minimize the weight of the drone, and some aluminum parts were also used, as shown in Figure 3. A cylindrical EDF fixture using a 3D printer was designed as an EDF dedicated frame for fixing each EDF. In the end, thanks to the frame made of carbon material, it was able to create a frame that can withstand a high-output EDF, while lightening the weight as much as possible.



**Figure 2.** Selected batteries: Selecting a battery that can supply the appropriate voltage for the EDF output ( $400 \text{ g} \times 2 = 800 \text{ g}$ ).

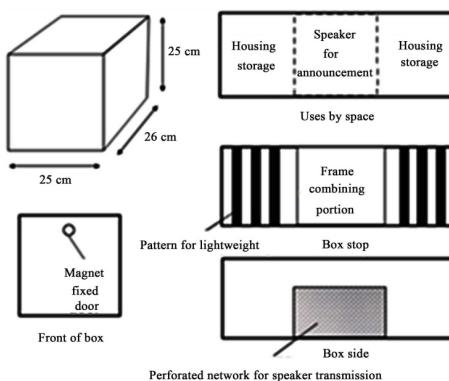


**Figure 3.** Frame drawing and specification: Carbon frame selection/2.2 kg (left), EDF fixture design drawing (center), Drone assembly and fabrication process (right).

## Design and Manufacture of Rescue Material Box

In order to minimize the imbalance in the flight of the drones, a rectangular box-shaped model penetrating the bottom of the drones was adopted. The vertical length is the minimum length, which is larger than the length of the frame including the EDF to avoid interference with the EDF. The height is defined as the maximum length that can be installed between the support and center of the drones frame. In order to deliver the rescue supplies, the drones must be approached through the windows and the being rescued person must reach the arm and bring the rescue supplies. Since the drones are not fully accessible to the building, the length of the arm of the person should be considered with the minimum length. In this case, it is necessary to design the arm length so that it is not difficult for both a person with a shorter arm and a person with a longer arm to approach, as shown in Figure 4. The design should be based on the average (498 mm) of the arm length first decile (lowest quartile) for women between 10 and 80 years of age. Height, width, and length are based on the minimum and maximum length.

The material called “Foam-max” was considered because of the weight and the strength of the box itself. Foam-max is foamed and compressed by using PVC as raw material and it is very easy to process while maintaining the advantages of plastic and wood. It was to create boxes and to make room for speakers in the middle. Speakers installed in the drones are able to communicate with the stranded person who are in need. It is also possible to reserve extra batteries in case of remaining space. In addition, a partition was installed to prevent internal objects from moving during the flight, thus preventing the center of gravity of the drones from being disturbed [11] [12].



**Figure 4.** Prototype of rescue material box.

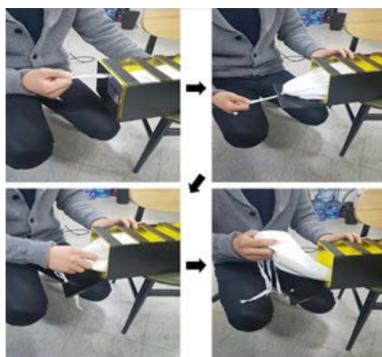
## Type of Rescue Material and Delivery Method

In the case of storage space for rescue materials, it is possible to store about 1 kilogram of materials. It is possible to store items according to the purpose and to greatly increase the survival rate in case of fire, such as oxygen cans, masks, firefighting clothes, fire extinguisher, water, water towel, and so on. Structural rope, radio, and emergency medicine can also be stored. In addition, if a cardiac arrest patient is present, the attached box can be removed and the cardiac defibrillator, portable AED, attached to the lower part of the drone can be delivered. When a high-rise building is delivered to a rescue target in the middle or upper floors, it approaches the building window and draws the rope connected to the storage bag connected to the storage box to deliver the rescue goods. As illustrated in Figure 5, the operation procedure is as follows:

- 1) Pull out the line outside the box.
- 2) At the same time the door is opened, the rescue materials come out.
- 3) Hold the rescue supplies.
- 4) Pull the rescue supplies.

## RESEARCH AND DEVELOPMENT RESULTS

The details of the developed drone is given in Figure 6. The drone was tested and evaluated for its flight characteristics. Figure 7 shows the test flight of takeoff and landing. The test was proceeded smoothly and the flight was very stable. In this study, EDF was selected because of the safety problem of the propeller, but it was not able to produce 4.5 kg of thrust as indicated in the product specification. 4.5 kg When 4 EDFs were used, the specification gave 18 kg of thrust.



**Figure 5.** Rescue material transport system.



| EDF                        |             |
|----------------------------|-------------|
| Duration (Per change)      | 3 minutes   |
| Noise                      | 74 dB       |
| Weight                     | 3140 ollars |
| Cost                       | 8.3 kg      |
| Speed                      | 1.4 m/s     |
| Actual loading weight      | 1.2 kg      |
| Theoretical maximum thrust | 18 kg       |
| Average thrust             | 9.5 kg      |

**Figure 6.** Developed drone with structural box and relief items and its specification.



**Figure 7.** Drones test flight: takeoff and landing.

Except for the weight of the body, 10 kg of thrust was expected, but the actual thrust was only 10.5 kg. And a run-time of 1/10th compared to propeller aircraft capable of flying over 30 minutes.

Thrust determination equation is as follows [9] [10] [11] [12].

$$\text{Power} = \text{Prop Const} * \text{rpm}^{\text{Power Factor}}$$

The equation to determine the propeller was as follows.

$$T = \frac{\pi}{4} D^2 \rho v \Delta v$$

T = thrust [N]

D = propeller diameter [m]

v = velocity of air at the propeller [m/s]

$\Delta v$  = velocity of air accelerated by propeller [m/s]

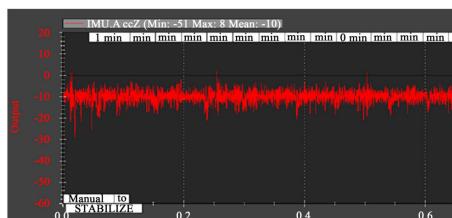
$\rho$  = density of air [1.225 kg/m<sup>3</sup>]

The thrust ratio was calculated by excluding the air velocity (v) related factor through the thrust formula. And when using a propeller, it was calculated that four times as much thrust was possible. This is because the diameter of the propeller has the greatest influence on thrust. To develop an EDF that produces

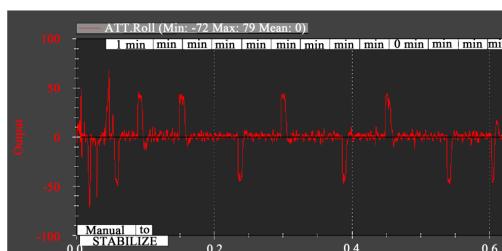
the same thrust as a 4× long propeller, it required 16 times the revolutions per second, which drastically reduced battery life. In addition, it required a high voltage of 30 V compared to a propeller requiring 22.2 V. In the case of a commercially available battery, the maximum voltage was 6s ( $3.7\text{ V} \times 4 = 22.2\text{ V}$ ). Finally, the weight of the EDF itself was 900 g, totaling 3.6 kg. It was about 9 times the weight of the propeller and I think it was the biggest weight gain factor in the airframe design that we are trying to reduce 1g. In order to compare the exact thrust between the propeller and the EDF, I think it would be necessary to consider vortex elimination and air compression rise through the duct. In order to actually introduce EDF, which has the advantage of safety, we need three things: 1, lighter weight, 2. efficiency, and 3. propeller size [13] [14] [15] [16]. Figures 8-12 shows the flight characteristics data, all of which shows a very stable flight characteristics.

|      | Wingspan         | Rotational Speed/s   | T/pv         |        |
|------|------------------|----------------------|--------------|--------|
| Prop | 45.72 cm         | 133.20               | 26.77        |        |
| EDF  | 11 cm            | 616.67               | 6.64         |        |
|      | Required Voltage | Used Battery Voltage | Thrust Ratio |        |
| Prop | 22.2v (6s)       | 22.2 (6s)            | 4            | 220(g) |
| EDF  | 30v (8s)         | 22.2 (6s)            | 1            | 900(g) |

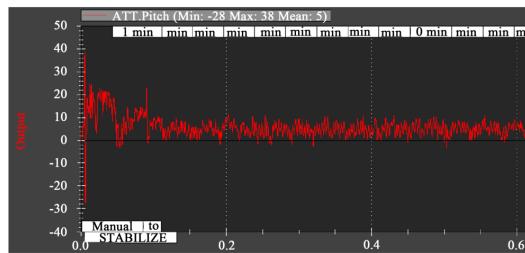
**Figure 8.** EDF and propeller comparison.



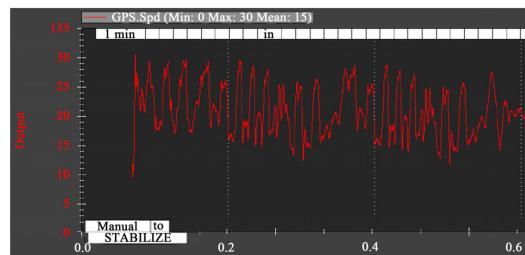
**Figure 9.** Z-axis acceleration data.



**Figure 10.** Bank angle data.



**Figure 11.** Pitch angle data.



**Figure 12.** Speed data.

## CONCLUSION

UAVs capable of high-power, vertical takeoff and landing can complement the disadvantages of conventional UAV's lack of power and maximize utilization throughout the industry. Through this technology development, we localize the core technology of UAV that can control high power and vertical flight stably. As the demand for UAV increases in the future, it will lay the foundations for developing various types of UAVs (courier, transportation, delivery, lifeguard, etc.) that meet customer demand. Expected effects of meeting domestic demand, import substitution and export will be created. In the future, we will complement the problems analyzed in this study and design a new type of drone. In the end, we will develop more than 25 kg payload so that it can be utilized for life-saving and transportation of rescue materials.

## ACKNOWLEDGEMENTS

This work was supported by the Ajou University research fund.

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

## REFERENCES

1. Yu, S. and Kwon, Y. (2017) Development of VTOL Drone for Stable Transit Flight. *Journal of Computer and Communications*, 5, 36-43. <https://doi.org/10.4236/jcc.2017.57004>
2. Yu, S. and Kwon, Y. (2017) Development of Multi-Purpose, Variable, Light Aircraft Simulator. *Journal of Computer and Communications*, 5, 44-52. <https://doi.org/10.4236/jcc.2017.57005>
3. Jo, D. and Kwon, Y. (2017) Analysis of VTOL UAV Propellant Technology. *Journal of Computer and Communications*, 5, 76-82. <https://doi.org/10.4236/jcc.2017.57008>
4. De la Torre, G.G., Ramallo, M.A. and Cervantes, E.G. (2016) Workload Perception in Drone Flight Training Simulators. *Computers in Human Behavior*, 64, 449-454.
5. Liu, Z. and Li, H. (2006) Research on Visual Objective Test Method of High-Level Flight Simulator. *Xitong Fangzhen Xuebao (Journal of System Simulation)*.
6. Luan, L.-N. (2013) Augmenting Low-Fidelity Flight Simulation Training Devices via Amplified Head Rotations. Loughborough University.
7. Lee, S. and Lim, K. (2008) PCB Design Guide Book. Sehwa Publishing Co., South Korea.
8. Kang, M. and Shin, K. (2011) Electronic Circuit. Hanbit Media, South Korea.
9. Kim, D., Kim, J. and Yoon, S. (2016) Development and Validation of Manned and Unmanned Aircraft Simulation Engine for Integrated Operation in NAS. *Journal of the Korean Society for Aeronautical & Space Sciences*, 44, 423-430.
10. Sponholz, C.-B. (2015) Conception and Development of a Universal, Cost-Efficient and Unmanned Rescue Aerial Vehicle. Techische Hochschule Wildau Technical University of Applied Sciences.
11. Guerrero, M.E., Mercado, D.A. and Lozano, R. (2015) IDA-PBC Methodology for a Quadrotor UAV Transporting a Cable-Suspended Payload. 2015 International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, 9-12 June 2015, 470-476. <https://doi.org/10.1109/ICUAS.2015.7152325>
12. Gomez, C. and Purdie, H. (2016) UAV-Based Photogrammetry and Geocomputing for Hazards and Disaster Risk Monitoring—A Review.

- Geoenvironmental Disasters, 3, 1. <https://doi.org/10.1186/s40677-016-0060-y>
- 13. Lee, E., Kim, S. and Kwon, Y. (2016) Analysis of Interface and Screen for Ground Control System. Journal of Computer and Communications, 4, 61-66. <https://doi.org/10.4236/jcc.2016.45009>
  - 14. Kwon, Y., Heo, J., Jeong, S., Yu, S. and Kim, S. (2016) Analysis of Design Directions for Ground Control Station (GCS). Journal of Computer and Communications, 4, 1-7. <https://doi.org/10.4236/jcc.2016.415001>
  - 15. Yu, S., Heo, J., Jeong, S. and Kwon, Y. (2016) Technical Analysis of VTOL UAV. Journal of Computer and Communications, 4, 92-97. <https://doi.org/10.4236/jcc.2016.415008>
  - 16. Hong, J., Baek, S., Jung, H., Kim, S. and Kwon, Y. (2015) Usability Analysis of Touch Screen for Ground Operators. Journal of Computer and Communications, 3, 133-139. <https://doi.org/10.4236/jcc.2015.311021>

# **CHAPTER**

# **15**

## **Railway Transport Infrastructure Monitoring by UAVs and Satellites**

---

**Sergey I. Ivashov, Alexander B. Tataraidze, Vladimir V. Razevig,  
Eugenia S. Smirnova**

Remote Sensing Laboratory, Bauman Moscow State Technical University,  
Moscow, Russia

### **ABSTRACT**

Improving the rail transport security requires development and implementation of neoteric monitoring and control facilities in conditions of increasing speed and intensity of the train movement and high level of terrorist threat. Use of Earth remote sensing (ERS), permitting to obtain information from large

---

**Citation:** Ivashov, S., Tataraidze, A., Razevig, V. and Smirnova, E. (2019), "Railway Transport Infrastructure Monitoring by UAVs and Satellites". *Journal of Transportation Technologies*, 9, 342-353. doi: 10.4236/jtts.2019.93022.

**Copyright:** © 2019 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

areas with a sufficiently high resolution, can provide significant assistance in solving the mentioned problems. This paper discusses the possibility of using various means of remote sensing such as satellites and unmanned aerial vehicles (UAV), also known as drones, for receiving information in different ranges of the electromagnetic spectrum. The paper states that joint using of both these means gives new possibilities in improving railroad security.

**Keywords:** Transport Infrastructure Monitoring, Remote Sensing, Satellite, Unmanned Aerial Vehicle (UAV), Aerial Photography, Radar Sensing, 3D Image Processing

## INTRODUCTION

The railway transportation safety demands constant attention from the staff and management of company that operates the facility. This applies to the locomotive drivers' condition monitoring, as well as to all services related to ensuring the smooth operation of the railway: manifestation that is called the human factor. These problems can be solved by a set of organizational and technical measures that improve discipline and working conditions of employees.

Unfortunately, there are still some technical factors, such as violation of the railway line integrity (rail rupture, destruction of railway arrow, etc.), which also require permanent monitoring. The another thing to mention is a problem of countering terrorist acts with the use of improvised explosive devices, which can lead to even more dire consequences. The use of remote sensing means such as piloted helicopters and drones could render substantial assistance in determining the scale of a railway catastrophe and ways of its elimination, including assistance to possible victims [1] [2] [3].

There is one more task connected to the topic of improving the safety of rail transportation. These are incidents related to natural phenomena including mudflows and avalanches, heavy snowfalls in the plains, and some others. Remote monitoring such phenomena could effectively help in its prediction and consequences management. To monitor and control the state of snow cover on the mountain slopes in avalanche prone areas, the UAV could be successfully used for reconstructing three-dimensional images of the terrain in the summer (without snow cover) and in the winter after heavy snowfalls. After comparison with such images, it will be possible to determine the height of the snow cover, and UAVs equipped with passive radar (radiometer) operated in centimeter or decimeter wavelengths will provide an opportunity to measure snowpack humidity [4]. These data could

be added by measurements obtained from a ground-based radar network [5]. The combination of these factors (stockpile and humidity of snow) will allow creating a fairly accurate forecast of avalanches probability.

The railway is a transporter of dangerous consignments such as explosives, perilous for environment and poisonous chemical compounds, combustible and flammable materials. In case of an accident, it is necessary to promptly assess the scale of the disaster and make decisions on the prompt elimination of its consequences. The described examples widely show possible applications of UAVs and other remote sensing tools, including earth observation satellites, which can be used to ensure the safety of rail transportation. The main goal of current paper is a comparison of the satellite and drone effectiveness in recording images of railway infrastructure in visual and microwave frequency range, and gives recommendation in their joint using. This matter was not analyzed before and main emphasis was mostly done on drones' applications in infrastructure inspection [6].

## **ADVANTAGES AND DISADVANTAGES OF REMOTELY PILOTED AIRCRAFTS**

Recent years have been characterized by an increasing use of UAVs with the expansion of tasks ranges, which are solved with their help. While in the past UAVs were used exceptionally for military purposes, now their civilian use is becoming extremely important. Military original interest to drones, which arose in the late 60s, was stipulated by a desire to avoid the risk for the crew lives during reconnaissance flights. The disadvantage of these drones was the need of returning to "home" territory after the flight mission for the treatment of the recorded photographic films and decryption of the material. That reduced the relevance of obtaining information.

Hereinafter, the most successful development of UAVs was implemented in Israel (tactical and operational UAVs) and the United States (tactical and strategic UAVs). Progress in the designing a new generation of reconnaissance UAVs is primarily associated with the emergence of digital photography, including cameras functioning in different wavelength ranges, navigation and communication through satellites. This significantly improved the performance of on-board equipment, especially the weight and size, and allowed improving the parameters of the UAV such as endurance, range and height of flight. It should also be noted that the cost of the UAV has sharply decreased, and access to satellite information has been improved in recent years.

Modern drones for civil purposes are characterized by low operation cost and no need for stationary airfields with paved runways. As a rule, they are launched from simplest ground airfields, mobile catapults or directly from the operator's hand. Remarkable progress has been achieved in the development of multi-rotor drone helicopters with electric power. UAVs of this type are characterized by simplicity of design and affordability. This is mainly due to the lack of such an essential for conventional helicopters a structural element as rotor hinge system, and the use asynchronous control of electric motors traction, that allows running the movement of the device in flight. Figure 1 shows a six-rotor UAV helicopter, which was used to obtain images of the Experimental Railway Ring (ERR) in Shcherbinka, Moscow region, in current project. Advantage of this type of UAV is also the fact that now their use in the exclusion zone of the railways at altitudes of less than 100 m does not require special permission for flights.

The disadvantage of multi-rotor electrical UAVs is a relatively short flight time usually less than an hour, and consequently restrictions on range and altitude of the flight. However, this is not essential when the operator controlling UAV flight is directly placed on the action place or near it. In this case, information about objects and events happening on earth surface can be obtained in real time and is compared with operator's own vision, what is essential, for example for making decisions in the elimination of railway accident consequences. An increase of the drone flight time can be achieved by replacing the batteries during intermediate landings, although the range limit remains, and it is still sometimes a significant factor.



**Figure 1.** Helicopter-type UAV: (a) The UAV during the flight with a camera mounted in the gimbal; (b) The operator with the remote control of the UAV on the background.

## ANALYSIS OF SATELLITE RESOURCES FOR OBTAINING INFORMATION

In the 1960s and 1970s the main users of photographic and radar satellite information were military, and this information itself was classified, but in recent decades a significant segment of consumers is the highly profitable civilian sector. In the first instance to the tasks to be solved by Earth observation satellites the following items should be included:

- Research of natural resources;
- Prediction, analysis and control of emergencies and their consequences;
- Geological exploration;
- Weather forecast;
- Control of the environment and its sources of pollution;
- Agriculture and prediction of harvest;
- Forestry and forest protection from fire and pests;
- Construction and other industries.

Remote sensing by satellites registers the radiation of the Earth's surface in different ranges of the electromagnetic spectrum. For above mentioned tasks multispectral photography and side-looking radars are mostly used. In some cases the lasers are used especially in surveying for the measurement of Earth's surface topography. Remote sensing data obtained from a spacecraft equipped with visible-range photographic cameras is characterized by a high degree of dependence on the atmosphere transparency and the times of day. Therefore, in order to remove these restrictions, remote sensing satellites equipped with side-looking radars have been widely developed. Radar image has worse resolution in comparison with optical one, but it has an ability to obtain images irrespectively of the weather and time of day, which is crucial for example for the navigation of ships on the Northern Sea Route during the polar night. Currently remote sensing satellites together with communication and navigation satellites constitute a significant segment of commercial space industry services. Often remote sensing satellites are created by organizations associated with the military industry of their own countries, which decreases the cost of their development due to the continuity of design solutions with military reconnaissance satellites.

Remote sensing satellites are usually launched into the so-called sun-synchronous orbit (sometimes called heliosynchronous), which has such

parameters that the satellite on this orbit, passes over any point of the earth's surface at about the same local solar time. Consequently, the angle of sun over the earth's surface is approximately the same at all paths of the satellite above the same earth's place. Permanent lighting conditions are very convenient for use in satellites receiving optical images of the earth's surface including remote sensing and meteorological satellites. The parameters of the sun-synchronous orbits fall within the range: the height above the earth's surface is 600 - 800 km, the period of rotation is 96 - 100 min, and the inclination of the orbit is about 98°.

The main advantage of the remote sensing optical satellites is the ability to obtain high-quality multispectral images, which resolution can reach 0.3 m or lesser. Nevertheless, visual spectrum satellites are not able to function in night or cloudy weather. While the first shortcoming can still be potentially overcome, for example, by using infrared optics, the cloudiness, which in mid-latitudes in winter can hide the earth for weeks or even months, makes impossible using even infrared satellites. All-weather satellites are radar ones, even though they have the worst resolution. At the same time, radar provides additional opportunities, for example, obtaining images of the recorded signal in different polarizations or finding underground objects [7].

## THE DATABASES OF SATELLITE INFORMATION

Currently, the results of satellite photography are widely distributed due to the accessibility and simplicity of work with them. For example, several sites in the Internet such as Google Maps (<https://www.google.com/maps>), Yahoo! Maps (<https://maps.yahoo.com/>) and others provide free access to photos recorded by satellites.

Satellite images and topographic maps based on these images are widely used even for everyday household tasks as navigating private cars or calling a taxi. Some websites provide only the results of satellite imagery allowing working with databases of photographs: NASA World Wind, TerraServer-US, Space image, Landsat Look Viewer (USGS).

Although free information services provide images with a sufficiently high resolution, they still do not give an opportunity to obtain actual to the moment information because data in them are rarely updated.

Photos for that websites were usually recorded in the warm season without the snow cover. Such images can be used for some types of analytical and theoretical works, but their drawbacks do not allow obtaining

timely and relevant information about the event or object of interest. For getting actual information, you should contact the specialized organizations and place the order for recording satellite information. This is a chargeable service but it allows the customer getting information according to his own needs: time, place and conditions of photography, resolution of images and other parameters.

The information may be provided as up-to-date one, i.e. recently received but also can be extracted from the database accumulated from previous years. The latter is important for information used in environmental studies, where it is significant to compare changes in the landscape over long period.

## **COMPARISON OF INFORMATION RECEIVED FROM DIFFERENT SOURCES**

To compare satellite and airborne images of the railway transport infrastructure, the Experimental Railway Ring in Shcherbinka was selected.

The choice of this object was related to the following circumstances. Firstly, ERR is well studied and located close to Moscow. Secondly, there were no difficulties with getting the permission for UAVs flights over the ERR territory because of its experimental status. Figure 2 shows the Experimental Ring image taken from the Google Maps database.

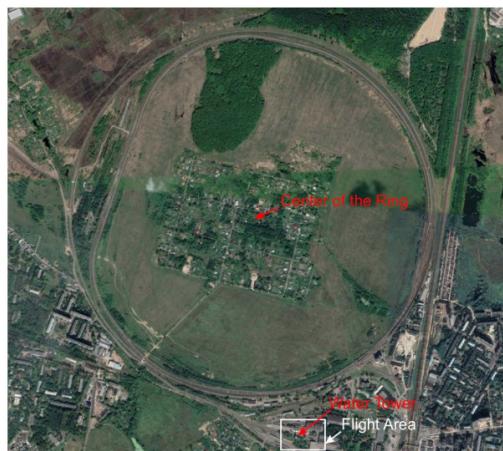
The center of the ERR ring is tagged with an arrow, its coordinates are 55.521373, 37.550190 (Northern latitude and Eastern longitude in degrees accordingly). Inside the Ring, a lot of summer country houses are located. The Experimental Railway Ring itself includes two double-tracks rings: outer and internal.

The outer ring has the shape of a regular circle, and the inner ring contains a straight section.

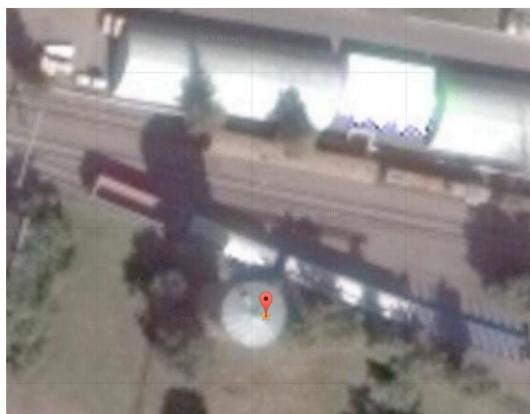
A rail depot is visible in the lower right corner of this picture. A rectangle marks flight area of UAV which photos are used in this article. The another arrow in the rectangle marks the location of a water tower, which served as a landmark for UAV's flights.

The enlarged image of the water tower obtained from the satellite is shown in Figure 3.

This is maximal scale that the Google Maps site allows. Roof of the tower has a conical shape and is covered with galvanized steel sheets. Joins between the sheets have the form of vertical ribs. The ribbed shape of the roof is visible even in the satellite image.



**Figure 2.** Image of the railway experimental railway ring from Google Maps free database.

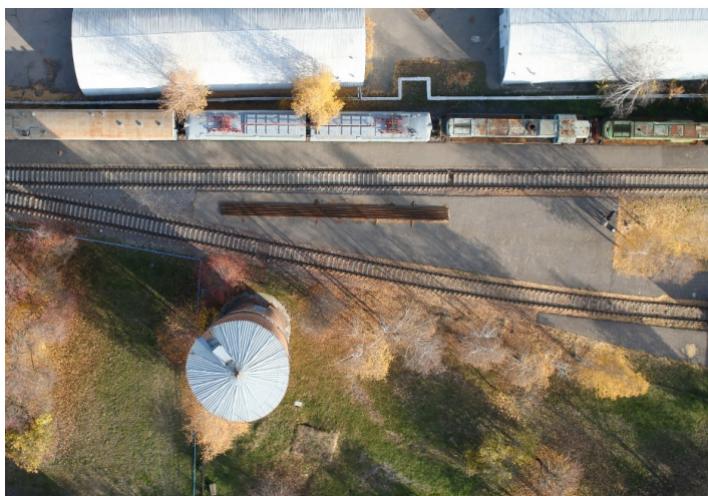


**Figure 3.** Large-scale satellite image of the water tower area on the experimental railway ring according to Google Maps.

The same area near the water tower was shot from the board of the UAV equipped with a Canon EOS 5D Mark II camera with EF-S 17-55 f/2.8 IS USM lens. Camera was mounted on a gimbal controlled by an operator, Figure 1. The result of the photographing from flight altitude about 70 m is shown in Figure 4. The difference between these two images consists in the location of the rail wagons on the tracks near the water tower that caused by different time of the shooting. The satellite image was recorded in summer, and drone flights were performed in autumn. They also differ

in size and direction of the local objects shadow, as the images were taken at different positions of the sun on the celestial sphere. This fact stresses the importance of sun-synchronous orbits, which were mentioned earlier and are commonly used for remote sensing satellites. Another advantage of using sun-synchronous orbits is the ability to determine whether an object is being built or not, and at what speed it is being built, judging by the length of the shadow at the images obtained from different orbit passes of the satellite. Usually this method is used for the interpretation of images recorded by military reconnaissance satellite, but it also can be used for cadastral surveying, when it is necessary to allocate from a large array of images new and illegally erected buildings. In that case, to determine the newly erected buildings and their height change it is enough to subtract the images.

If we compare the resolution of the two above photos, it is obvious that the UAVs image has a much better resolution than satellite one. This is clear since the altitude of the orbits of remote sensing satellites is about 600 km in comparison with the same parameter for drones that equals a few hundred meters or several kilometers. UAVs have another advantage over satellites because they are less dependent on weather conditions. In the presence of cloudiness, it is impossible to shoot in the visible spectrum from satellites. While UAVs can hover below the lower cloud boundary and take pictures of the terrain.



**Figure 4.** Photograph of the water tower area on the ERR recorded by the UAV.

As mentioned earlier, only radar imaging is weather-independent. Two low-resolution images of the Experimental Ring district obtained from the Google Maps website and the European Sentinel-1 radar satellite database (Sentinel-1, <https://sentinel.esa.int/web/sentinel/missions/sentinel-1/> / observation-scenario/archive) are presented in Figure 5.

At the lower part of both images Ostafievo airport (left in the pictures) and ERR (right) are seen.

Both presented images have approximately the same scale, but the visible-range image contains more detail than the radar one.

It is worth mention that radar image clearly depicts runways of the airport. The scale of both images can be easily estimated, because a diameter of the outer circle of the railway testing area is 1912 m.

The main advantage of radar, as it was mentioned earlier, is the ability to record images during the whole day and in all-weather conditions. In some cases, radar images may be the only source of information.

In this regard, it can be concluded that to solve various problems in the interests of railway safety, it is important to have access to different sources of remote sensing information.

## **RECONSTRUCTION OF 3D MODELS OF THE RAILWAY INFRASTRUCTURE OBJECTS FROM A SET OF 2D IMAGES**

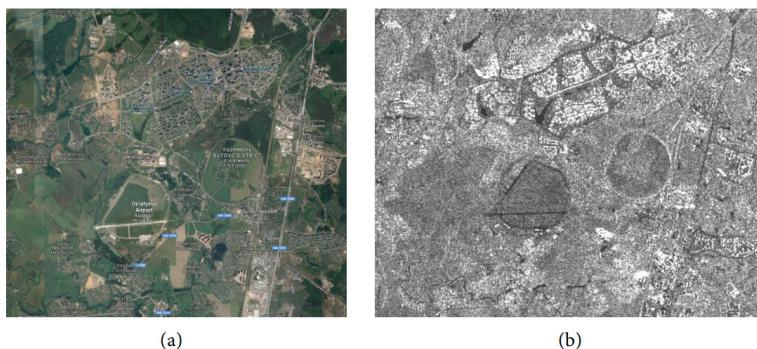
Modern image processing software allows 3D environment model reconstructing from a set of 2D images. This gives an ability to view objects from different angles and heights.

Such technology can be used in designing complex objects of the railway infrastructure (bridges, tunnels, etc.), as well as in solving problems of their safety.

In the recent time, the increasing interest in using 2D aerial images, collected by the UAV, in 3D model reconstruction is evident.

It is very important to note that the UAV simplifies the process of taking images because operator can set the trajectory of the vehicle individually [8].

The multiple images are acquired by on board high resolution camera from multiangle and multiposition during the flight of the UAV.



**Figure 5.** Satellite images of the Experimental Railway Ring neighborhood: (a) low-resolution visible spectrum image the according to Google Maps; (b) radar image recorded by the Sentinel-1 satellite, EU.

The essence of an image is a projection from a 3D scene onto a 2D plane, during which process the depth is lost. The 3D point corresponding to a specific image point is constrained to be on the line of sight. From a single image, it is impossible to determine which point on this line corresponds to the image point. If two images are available, then the position of a 3D point can be found as the intersection of the two projection rays. This process is referred to as triangulation.

The key for this process is the relations between multiple views which convey the information that corresponding sets of points must contain some structure and that this structure is related to the poses and the calibration of the camera. The task of converting multiple 2D images into 3D model consists of a series of processing steps which brief overview is given here. For a more extensive treatment the reader is referred to textbooks such as [9] [10].

Camera calibration consists of internal (focal length, coordinates of the principal point, distortion coefficients) and external (position and orientation) parameters, without which at some level no arrangement of algorithms can work. Internal parameters are independent of outer conditions and do not change from one snapshot to another snapshot.

In most situations, the internal parameters are calculated by shooting calibration objects, the geometry of which is pre-known, but also it can be computed without their use by applying a large number of images with common points [9]. External calibration is performed by triangulating key points common in two or more images.

To synthesize a 3D image, the problem of finding a match between key points in different images has to be solved. This task is accomplished by searching for points with minimum different descriptors-vector parameters describing the point. Thereby the SIFT algorithm describes each key point by a vector of 128 parameters identified on the basis of local gradients in the vicinity of the point [11].

One of the most efficient algorithms for filtering false correspondences of certain key points is RANSAC, Random Sample Consensus, [12]. The essence of the mentioned method is iterative search for the best transformation matrix (fundamental matrix) between N randomly selected key points in the first image and the corresponding points in the second one. From the initials set of key points RANSAC algorithm iteratively removes such points, deviation of which as a result of the applying the found transformation from the corresponding points in another image, exceeds the specified threshold.

Depth determination serves as the most challenging part in the whole process, as it calculates the 3D component missing from any given image—depth. The correspondence problem, finding matches between two images so the position of the matched elements can then be triangulated in 3D space is the key issue here. Once the multiple depth maps are determined they have to be combined to create a final mesh by calculating depth and projecting out of the camera. Camera calibration will be used to identify where the many meshes created by depth maps can be combined together to develop a larger one, providing more than one view for observation.

Complete 3D mesh may be the final goal, but applying the color from the original photographs to the mesh is usually required. In this paper the “mosaic” approach [13] was used to create the final texture which implies two-step approach: it does blending of low frequency component for overlapping images to avoid seamline problem (weighted average, weight being dependent on a number of parameters including proximity of the pixel in question to the center of the image), while high frequency component, that is in charge of picture details, is taken from a single image—the one that presents good resolution for the area of interest while the camera view is almost along the normal to the reconstructed surface in that point.

As the initial data for the reconstructing a 3D model of the earth surface with the objects of the railway infrastructure located on it, 176 drone photos were used. A few examples of the original photos are shown in Figure 6.

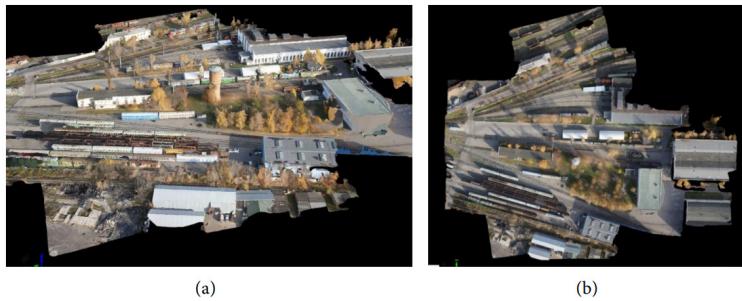
The resulting 3D model of the surface under different camera angles and from different heights is shown in Figure 7.

A more complete picture of the obtained model is demonstrated in the video, which could be downloaded from webpage [14]. Some limitations of the data, which have been used for reconstruction of 3D image, should be noted:

- A relatively small number of images;
- Non-optimal flight path of the UAV;
- Small overlap between images;
- Absence of information about the orientation of the UAV and camera, as well as data on GPS/GLONASS coordinates.



**Figure 6.** Examples of original images recorded by the UAV.



**Figure 7.** 3D reconstructed images of the railway depot.

Nevertheless, the resulting model sufficiently accurately reflects the features of the area where the photography was made. The model was constructed using the mathematical software Pix4D Mapper (Pix4Dmapper, <https://pix4d.com/product/pix4dmapper/>).

## CONCLUSION

The analysis of the possibilities of using remote sensing satellites and UAVs equipped with various sensors showed that they could be an

effective monitoring tool for ensuring traffic safety on the railways. Their information also can be used for solving other tasks, for example in the design and construction of railways infrastructure. The paper shows that the effectiveness of the Earth remote sensing can be increased by combining information recorded by satellites and UAVs, and with the use of the modern image processing means, it becomes possible to build synthetic three-dimensional images of railway infrastructure.

## **ACKNOWLEDGEMENTS**

The study was performed with financial support of the Russian Foundation for Basic Research, Project No. 17-20-02086. The authors express their gratitude to the staff of the Experimental Railway Ring for assistance in conducting of UAV flights for obtaining the necessary information.

## **CONFLICTS OF INTEREST**

The authors declare no conflicts of interest.

## REFERENCES

1. Flammini, F., Pragliola, C. and Smarra, G. (2016) Railway Infrastructure Monitoring by Drones. International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference, Toulouse, 2-4 November. <https://doi.org/10.1109/ESARS-ITEC.2016.7841398>
2. Eyre-Walker, R.E.A. and Earp, G.K. (2008) Application of Aerial Photography to Obtain Ideal Data for Condition Based Risk Management of Rail Networks. The 4th IET International Conference on Railway Condition Monitoring, Derby, 18-20 June 2008. <https://doi.org/10.1049/ic:20080353>
3. Francesco, F., Naddei, R., Pragliola, C. and Smarra, G. (2016) Towards Automated Drone Surveillance in Railways: State-of-the-Art and Future Directions. International Conference on Advanced Concepts for Intelligent Vision Systems, Lecce, 24-27 October 2016, 336-348. [https://doi.org/10.1007/978-3-319-48680-2\\_30](https://doi.org/10.1007/978-3-319-48680-2_30)
4. Schwank, M. and Naderpour, R. (2018) Snow Density and Ground Permittivity Retrieved from L-Band Radiometry: Melting Effects. *Remote Sensing*, 10, 354. <https://doi.org/10.3390/rs10020354>
5. Skofronick-Jackson, G., Kim, M., Weinman, J.A. and Chang, D.-E. (2004) A Physical Model to Determine Snowfall Over Land by Microwave Radiometry. *IEEE Transactions on Geoscience and Remote Sensing*, 42, 1047-1058. <https://doi.org/10.1109/TGRS.2004.825585>
6. Besada, J.A., Bergesio, L., Campaña, I., Vaquero-Melchor, D., López-Araquistain, J., Bernardos, A.M. and Casar, J.R. (2018) Drone Mission Definition and Implementation for Automated Infrastructure Inspection Using Airborne Sensors. *Sensors*, 18, 1170. <https://doi.org/10.3390/s18041170>
7. Koyama, C.N. and Sato, M. (2015) Detection and Classification of Subsurface Objects by Polarimetric Radar Imaging. *IEEE Radar Conference*, Johannesburg, 27-30 October 2015, 440-445. <https://doi.org/10.1109/RadarConf.2015.7411924>
8. Bulgakov, A., Evgenov, A. and Weller, C. (2015) Automation of 3D Building Model Generation Using Quadrotor. *Procedia Engineering*, 123, 101-109. <https://doi.org/10.1016/j.proeng.2015.10.065>
9. Hartley, R. and Zisserman, A. (2004) Multiple View Geometry in Computer Vision. 2nd Edition, Cambridge University Press,

- Cambridge. <https://doi.org/10.1017/CBO9780511811685>
10. McGlone, J.C. (2013) Manual of Photogrammetry. Sixth Edition, American Society for Photogrammetry and Remote Sensing, Bethesda, Maryland, USA, 1318 p.
  11. Lowe, D.G. (1999) Object Recognition from Local Scale-Invariant Features. Proceedings of the International Conference on Computer Vision, Kerkyra, 20-27 September 1999, Vol. 2, 1150-1157. <https://doi.org/10.1109/ICCV.1999.790410>
  12. Fischler, M. and Bolles, R. (1981) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM, 24, 381-395. <https://doi.org/10.1145/358669.358692>
  13. Maenpaa, T. and Pietikainen, M. (2005) Texture Analysis with Local Binary Patterns. In: Handbook of Pattern Recognition and Computer Vision, 3rd Edition, World Scientific, Singapore, 197-216. [https://doi.org/10.1142/9789812775320\\_0011](https://doi.org/10.1142/9789812775320_0011)
  14. <http://www.rslab.ru/pubdir/1.wmv>

# **CHAPTER**

# **16**

## **Assessment of UAV Based Vegetation Indices for Nitrogen Concentration Estimation in Spring Wheat**

---

**Olga S. Walsh<sup>1</sup>, Sanaz Shafian<sup>1</sup>, Juliet M. Marshall<sup>2</sup>, Chad Jackson<sup>2</sup>, Jordan R. McClintick-Chess<sup>1</sup>, Steven M. Blansct<sup>3</sup>, Kristin Swoboda<sup>4</sup>, Craig Thompson<sup>4</sup>, Kelli M. Belmont<sup>5</sup>, Willow L. Walsh<sup>6</sup>**

<sup>1</sup>Department of Plant Sciences, Southwest Research and Extension Center, University of Idaho, Moscow, ID, USA.

<sup>2</sup>Department of Entomology, Plant Pathology, and Nematology, Idaho Falls Research and Extension Center, University of Idaho, Idaho Falls, ID, USA.

<sup>3</sup>BASF-Chemical Co., Caldwell, ID, USA.

<sup>4</sup>Take Flight UAS, LLC, Boise, ID, USA.

<sup>5</sup>Seminis, Payette, ID, USA.

<sup>6</sup>Vallivue High School, Caldwell, ID, USA.

---

**Citation:** S. Walsh, O., Shafian, S. , M. Marshall, J., Jackson, C., R. McClintick-Chess, J., M. Blansct, S., Swoboda, K. , Thompson, C., M. Belmont, K. and L. Walsh, W. (2018) "Assessment of UAV Based Vegetation Indices for Nitrogen Concentration Estimation in Spring Wheat". *Advances in Remote Sensing*, 7, 71-90. doi: 10.4236/ars.2018.72006.

**Copyright:** © 2018 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

## ABSTRACT

Unmanned Aerial Vehicles (UAVs) have become increasingly popular in recent years for agricultural research. High spatial and temporal resolution images obtained with UAVs are ideal for many applications in agriculture. The objective of this study was to evaluate the performance of vegetation indices (VIs) derived from UAV images for quantification of plant nitrogen (N) concentration of spring wheat, a major cereal crop worldwide. This study was conducted at three locations in Idaho, United States. A quadcopter UAV equipped with a red edge multispectral sensor was used to collect images during the 2016 growing season.

Flight missions were successfully carried out at Feekes 5 and Feekes 10 growth stages of spring wheat. Plant samples were collected on the same days as UAV image data acquisition and were transferred to lab for N concentration analysis. Different VIs including Normalized Difference Vegetative Index (NDVI), Red Edge Normalized Difference Vegetation Index ( $NDVI_{red\ edge}$ ), Enhanced Vegetation Index 2 ( $EVI_2$ ), Red Edge Simple Ratio ( $SR_{red\ edge}$ ), Green Chlorophyll Index ( $CI_{green}$ ), Red Edge Chlorophyll Index ( $CI_{red\ edge}$ ), Medium Resolution Imaging Spectrometer (MERIS) Terrestrial Chlorophyll Index (MTCI) and Red Edge Triangular Vegetation Index (core only) ( $RTVI_{core}$ ) were calculated for each flight event.

At Feekes 5 growth stage, red edge and green based VIs showed higher correlation with plant N concentration compare to the red based VIs.

At Feekes 10 growth stage, all calculated VIs showed high correlation with plant N concentration. Empirical relationships between VIs and plant N concentration were cross validated using test data sets for each growth stage. At Feekes 5, the plant N concentration estimated based on  $NDVI_{red\ edge}$  showed one to one correlation with measured N concentration. At Feekes 10, the estimated and measured N concentration were highly correlated for all empirical models, but the model based on  $CI_{green}$  was the only model that had a one to one correlation between estimated and measured plant N concentration.

The observed high correlations between VIs derived from UAV and the plant N concentration suggests the significance of VIs deriving from UAVs for within-season N concentration monitoring of agricultural crops such as spring wheat.

**Keywords:** Unmanned Aerial Vehicles and Systems (UAV), Vegetation Indices (VIs), Plant Nitrogen Concentration

## INTRODUCTION

Nitrogen (N) is one of the essential factors for crop production in terms of plant growth and development and crop quality [1] [2]. Adequate supply of N is fundamental for optimizing wheat (*Triticum aestivum L.*) yield and grain quality [3] [4]. Nitrogen regulates plant growth processes and plays a vital role in chlorophyll (CL) production—the basis for the photosynthesis process [5]. Insufficient N supply can negatively affect photosynthesis process and result in crop yield and quality penalties [3]. On the other hand, excessive N application to agricultural crops has been associated with nitrate leaching, soil denitrification, ammonia volatilization, and nitrous oxide contamination of aquifers and aggravating the climate change [6] [7]. Dynamic and efficient fertilization (appropriate time and rate) is very important for optimizing crop yield and maintaining environmental quality [8]. Accurate estimation of crop N concentration is vital for developing effective fertilizer-N recommendations.

There is a strong correlation between N concentration and CL content at foliar and canopy scale because most of leaf N is localized within the CL molecules [9] [10] [11] [12]. Chlorophyll content is the main elements that govern the crop reflectance in the visible (VIS) and near infrared (NIR) regions of spectrum [8]. Thus, vegetation reflectance in these parts of spectrum is closely associated with N concentration. Remote sensing enables to acquire crop reflectance and provide diagnostic information on crop N concentration quickly and in a spatial context, compared to traditional destructive sampling techniques [13]. During the last few decades, scientists have proposed several vegetation indices (VIs) calculated from reflectance data to assess CL content and N concentration [8] [13] [14] [15]. These VIs are mostly a combination of NIR and VIS spectral bands, representing radiation scattering by canopy and radiation absorption by CL respectively [16]. Although these VIs accurately estimate CL and N concentration early in the growing season at lower CL values, they become less sensitive as the red spectral band is strongly absorbed by CL. Gitelson and Merzlyak [17] showed that red edge region is sensitive to a wide range of CL content values, and the use of this part of spectrum in VIs calculation can reduce the saturation effect due to lower absorption of the red edge region by CL. Several VIs based on this spectral region have been developed and used successfully to estimate CL and N concentration.

Gitelson and Merzlyak [17] replaced the red spectral band (675 nm) with red edge spectral band (705 nm) in Normalized Difference Vegetation

Index (NDVI) and developed a new index called Red Edge Normalized Difference Vegetation Index ( $NDVI_{red\ edge}$ ). They showed that traditional NDVI had a tendency to become saturated at higher CL level of senescent maple and chestnut leaves while  $NDVI_{red\ edge}$  continued to show strong linear correlation with CL content and observed no saturation issue. In a similar study, Gitelson et al. [18] showed that reciprocal of red edge spectral band is closely related to the CL content in leaves of all species. They proposed Red Edge Chlorophyll Index ( $CI_{red\ edge}$ ) and showed that  $CI_{red\ edge}$  is highly correlated with CL content (coefficient of determination  $R^2 > 0.94$ ). In another study, Dash and Curran [19] developed MERIS Terrestrial Chlorophyll Index (MTCI). They used data in three red/NIR wavebands centered at 681.25, 708.75 and 753.75 nm (bands 8, 9 and 10 in the MERIS standard band setting) to develop MTCI. They determined the relationship between MTCI and CL content using actual CL content for sites in the New Forest, United Kingdom (UK) and for small plots in a greenhouse experiment. Their results showed that MTCI is strongly and positively correlated to actual CL content. Li et al. [5] evaluated red edge based spectral indices for estimating plant N concentration and uptake of maize (*Zea mays L.*). They calculated chlorophyll content index (CCCI),  $NDVI_{red\ edge}$ ,  $CI_{red\ edge}$  and MTCI from hyperspectral narrow bands, simulated Crop Circle ACS-470 active crop canopy sensor bands and simulated WorldView-2 satellite broad bands. Their results showed that there is a positive strong correlation between red edge based VIs and N concentration in maize. Their results also indicated that CCCI performed the best across different bandwidths for estimating maize plant N concentration at the V6 and V7 and V10 - V12 stages. In another study, Wang et al. [20] compared broad-band and red edge based spectral VIs to estimate N concentration in corn, wheat, rice (*Oryza sativa L.*) and soybeans (*Glycine max L.*). They calculated various VIs from images acquired by the Compact Airborne Spectrographic Imager (CASI) sensor. Their result showed that NDVI performed the best compared to other VIs, and red edge based VIs did not show potential for accurate estimation of leaf N concentration data due to spectral resolution.

Unmanned aerial vehicles (UAVs) are remote sensing systems that can capture crop reflectance in the VIS-NIR region of spectrum and assess CL and N concentration. The UAVs, which have recently gained traction in number of studies, acquire ultra-high spatial resolution images by flying at low altitudes [21] [22]. Operational advantages such as low-cost systems, high flexibility in terms of flight planning and acquisition scheduling, and imaging below cloud cover make UAVs an appropriate tool to study crop

biophysical parameters including N concentration [23]. In a few studies, scientists have used reflectance data derived from UAVs to estimate CL or N concentration. Lu et al. [24] mounted a Mini Multi-Camera Array (Mini-MCA) imaging system on an octocopter UAV to estimate rice N status. They examined various VIs to estimate N concentration at panicle initiation and stem elongation growth stages in rice.

Their results showed that MTCI was best for estimating rice N concentration with  $R^2 = 0.48$ . In another study, Caturegli et al. [25] compared the spectral reflectance of three turfgrasses (*Cynodon dactylon* × *transvaalensis* (Cdxt) “Patriot”, *Zoysia matrella* (Zm) “Zeon” and *Paspalum vaginatum* (Pv) “Salam”) acquired with a UAV and using a ground based instrument.

They also tested the sensitivity of the two data acquisition sources in detecting induced variation among N application levels and for NDVI calculation. Their results showed that NDVI obtained with the ground based sensor was highly correlated with UAV based NDVI, with correlation coefficient values ranging from 0.83 to 0.97.

They also showed that UAV based NDVI was strongly correlated with N measured in the clipped plant biomass samples (correlation coefficient of 0.95). Similarly, Hunt et al. [26] used a UAV to monitor N status of irrigated potato (*Solanum tuberosum* L.). They used a small parafoil-wing UAV to acquire color-infrared images.

They showed that each of applied N treatments could be precisely distinguished in the images. Their results also concluded that NDVI and Green Normalized Difference Vegetation Index (GNDVI) were not useful for in-season N management in potato because the above-ground changes in leaf CL were not sufficiently large to be detected by remote sensing early in the growing season.

So far, very few studies have investigated the potential of using red edge based VIs from the UAV data for canopy CL or N concentration estimation.

To date, no studies on comparing red edge based VIs from the UAV data for wheat canopy CL or N concentration estimation have been reported. The main goal of this study was to evaluate the performance of UAV based VIs in estimating plant N concentration at canopy scale.

Specifically, we analyzed and statistically compared the performance of different red edge based VIs from UAV data to estimate spring wheat plant N concentration.

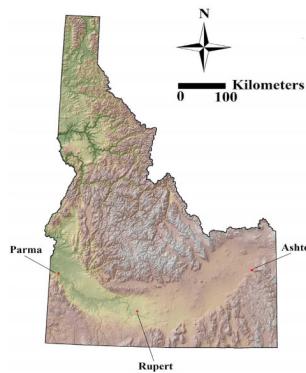
## MATERIALS AND METHODS

### Study Area

The experimental studies were conducted at five different locations in Idaho during 2016 growing season (Table 1 and Figure 1). The soil type, mean annual temperature, and mean annual precipitation for each study site are presented in Table 1.

**Table 1.** Latitude, longitude, soil type, mean annual precipitation, and mean annual temperature for five locations in Idaho.).

| Field  | Lat      | Lon        | Soil Type                                 | Mean annual        |                  |
|--------|----------|------------|---|--------------------|------------------|
|        |          |            |   | precipita-<br>tion | temper-<br>ature |
| Parma  | 43.80243 | -116.94291 | Green leaf-Owyhee silt loams              | 24 cm              | 10°C             |
| Ashton | 44.07127 | -111.39983 | Greentimber-Marystown-Robinlee silt loams | 43 cm              | 5°C              |
| Rupert | 42.72016 | -113.50641 | Sluka silt loam                           | 28 cm              | 10°C             |



**Figure 1.** Example of a figure caption (figure caption).

Hard red spring wheat (cv. Cabernet) was planted using a Hege 500 series drill at Rupert and Ashton, and soft white spring wheat (cv. Alturas) was planted at Parma using H&N Equipment research plot drill, at a density of approximately  $106.5 \text{ kg seeds ha}^{-1}$ . Row spacing was set at 17.78 cm using double disk openers. The plots were 1.52 wide by 4.27 m long, and then reduced to 3.05 m using glyphosate and tillage. Granular urea (46-0-0) was surface broadcasted immediately after planting at five different rates (0, 84,

168, 252, and 336 kg N  $\text{ha}^{-1}$ ). Each treatment was replicated four times in a randomized complete block design, resulting in a total of 20 plots at each location. Spring planting conditions were good for crop establishment. Soil moisture March through April were above average, which resulted in excellent early season growth and development. Early season precipitation provided excellent growing conditions until irrigation became available in April when all sites were irrigated every 7 to 10 days using sprinkle irrigation systems. Timely planting dates resulted in excellent tillering and a long growing period.

## Unmanned Aerial System

A quadcopter UAV 3DR Solo (3D Robotics, Inc., Berkeley, CA) shown in Figure 2 was selected to carry camera payloads to acquire ultra-high-resolution imagery. Solo is powered by four electric brushless 880 kV motors, two that spin clockwise, and two counterclockwise. Solo's arms are labeled 1 to 4 on the ends of the arms. Motors on arms #1 and #2 spin counterclockwise and use clockwise-tightening propellers with silver tops. Conversely, motors on arms #3 and #4 spin clockwise and use counter-clockwise tightening propellers with black tops. Solo's onboard computers control navigation, attitude, and communications in flight while sending real-time telemetry and video output and receiving control inputs over the 3DR Link secure WiFi network. A 14.8v dc 5200 mAh lithium polymer (LiPo) rechargeable battery is located next to the power button. The intelligent battery systems on the Solo tracks battery power and informs the pilot when the battery needs to be recharged. Solo includes a GoPro® the Frame (GoPro, Inc, San Mateo, CA) fixed mount to support a GoPro® HERO camera (GoPro, Inc, San Mateo, CA). Alternatively, the fixed mount could be replaced by the optional 3-Axis Gimbal (3D Robotics, Inc, Berkeley, CA) (Figure 2). Empty weight of the quadcopter is 1.74 kg and weight increased to 1.9 kg with compatible camera and Solo Gimbal.



**Figure 2.** The Unmanned Aerial System 3DR Solo.

A MicaSense Red Edge™ 3 Multispectral Camera (MicaSense, Inc, Seattle, WA) with an integrated Global Positioning System (GPS) with an accuracy of 2 - 3 meters, mounted on the UAV was used to obtain the imagery. The camera was mounted on a Gimbal and as the camera's weight was similar to GoPro camera's weight, there was no need to add balance weight. The camera acquires 1.3-megapixel images in five spectral bands (red edge, near infrared, red, green, and blue) with 12-bit Digital Negative (DNG) or 16-bit Tag Image File Format (TIFF) radiometric resolution. The ground spatial resolution of resulting images from the narrowband imager is 8 cm (per band) at 120 m above ground level. The MicaSense Red Edge™ 3 has a Downwelling Light Sensor (DLS) (MicaSense, Inc, Seattle, WA), which measures the ambient light during a flight for each of the five bands and records this information in the metadata of the images captured by the camera. The gain and exposure settings are automatically optimized for each capture and each band to prevent blurring or over-exposure, which results in properly exposed images.

## Multispectral Image Acquisition and Processing

The UAV images were captured within 2.0 hours of solar noon with flight duration ranging from 15 to 20 minutes in sunny and cloud free conditions. Mission planer software [27] was used to design the flight path and choose the flight and sensor parameters to ensure there is an adequate overlap between acquired images for mosaicking. Two flight missions performed at each location to coincide with Feekes 5 and Feekes 10 spring wheat growth stages resulted in six flight missions per season. These growth stages were chosen because N fertilizer applied at these growth stages has potential to maximize grain yield and quality.

Acquired multispectral images were imported to Micasense Atlas software (MicaSense, Inc, Seattle, WA) for mosaicking, georeferencing and radiometric calibration. Micasense Atlas has a partnership with Pix4D mapper image analysis software (Pix4D SA, Lausanne, Switzerland) to create aligned, mosaicked and georeferenced images from multispectral data captured with the MicaSense red edge camera. The Pix4D finds hundreds of tie-points between overlapped images and stiches the individual images together to build one ortho-rectified image of the whole study area. The accuracy of an outputted reconstructed images are usually 1-2 times the ground spatial resolution. In this study, we had different treatments in the adjacent rows (100 cm row spacing) (well separated plots), so a low level of mosaicking error is allowable when using UAV images for our purpose

[28]. Then, the mosaicked images were radiometrically calibrated using the Red Edge Camera Radiometric Calibration Model in Atlas software. Atlas software uses the calibration curve associated with a Calibrated Reflectance Panel (CRP) to perform calibration model and convert the raw pixel values of an image into absolute spectral radiance values. The CRP was placed adjacent to the study area during each flight mission, and an image of the CRP was captured immediately before and immediately after each flight. The output of radiometric calibration model is a 5-layer, 16-bit orthorectified GeoTIFF image.

## Field Sampling and Measurements

To obtain a representative plant sample, aboveground biomass was destructively sampled at Feekes 5 and Feekes 10 growth stages by cutting three randomly selected plants in the middle of each plot immediately after each UAV flight event. Plant samples were dried in the oven for 72 hours at 80°C and then were transferred to the lab for N content analysis. Samples' N content analysis was performed using the AOAC method 990.3 [29] at Brookside Laboratories, Inc (New Bremen, OH, USA) with extended uncertainty of  $\pm 5\%$ .

## Vegetation Indices

The UAV reflectance data were used for calculating eight VIs, many of which have been proposed as surrogates for canopy N concentration estimation. The VIs tested include the Normalized Difference Vegetation Index, NDVI [30], the Red Edge Normalized Difference Vegetation Index,  $NDVI_{red\ edge}$  [17], the Enhanced Vegetation Index 2, EVI2 [31], the Red Edge Simple Ratio,  $SR_{red\ edge}$  [32], the Green and Red Edge Chlorophyll Indices,  $CI_{green}$  and  $CI_{red\ edge}$ , respectively [18], the MERIS Terrestrial Chlorophyll Index, MTCI [19], and the Core Red Edge Triangular Vegetation Index ( $RTVI_{core}$ ) [33] (Table 2). For each study plot, a region of interest (ROI) was manually established by choosing the central two rows and mean of each VI value corresponding to that plot was extracted.

## Statistical Analysis

The study plots were randomly divided into test and training data sets. For the training data sets, simple regression analysis was performed to find the best relationship fit between N concentration and each UAV based VI. The determination coefficient ( $R^2$ ) and Root Mean Squared Error (RMSE) were

used to evaluate the predictive accuracy of each model. These parameters are widely used to evaluate the performance of empirical models. The RMSE are computed as shown in Equations (1):

$$RMSE = \sqrt{\left[ \frac{\sum_{i=1}^M (\hat{y}_i - y_i)^2}{M} \right]} \quad (1)$$

where,  $\hat{y}_i$  is predicted value of N concentration;  $y_i$  is measured N concentration, and M is total number of observations. In the next step, the test data set was used to evaluate the performance of developed model in the previous step. Predicted values of N concentration were plotted versus corresponding values of N concentration measured in the lab. The performance of regression models in estimating N for the training data set were evaluated by calculating the  $R^2$  and

**Table 2.** Vegetation indices (VIs) tested in this study to estimate nitrogen (N) content.

| Vegetation Index  | Equation  |
|---|---|
| Normalized Difference Vegetation Index (NDVI)                                       | (NIR – Red)/(NIR + Red)                               |
| Red Edge Normalized Difference Vegetation Index ( $NDVI_{red\ edge}$ )              | (NIR – Red Edge)/(NIR + Red Edge)                     |
| Enhanced Vegetation Index 2 ( $EVI_2$ )   | $2.5 \times (NIR - Red) / (NIR + 2.4 \times Red + 1)$ |
| Red Edge Simple Ratio ( $SR_{red\ edge}$ )  | (NIR) / (Red Edge)                                    |
| Green Chlorophyll Index ( $CI_{green}$ )  | (NIR/Green) – 1                                       |
| Red Edge Chlorophyll Index ( $CI_{red\ edge}$ )                                     | (NIR/Red Edge) – 1                                    |
| Medium Resolution Imaging Spectrometer (MERIS) Terrestrial Chlorophyll Index (MTCI) | (NIR – Red Edge)/(Red Edge + Red)                     |
| Core Red Edge Triangular Vegetation Index ( $RTVI_{core}$ )                         | $100(NIR - Red\ Edge) - 10(NIR - Green)$              |

RMSE. In addition, Student's t-tests were used to determine if the slope and the intercept of the regressions were significantly different from 1 and 0, respectively. If the values of slopes were not significantly different from 1 and the values of intercepts were not significantly different from 0, then it was concluded that the regression was not significantly different from the 1:1 line, and the empirical model could accurately predict N concentration.

## RESULTS AND DISCUSSIONS

### Variation of Plant N Concentration

Based on different N application rates, a wide range of N concentration (%) ranged from 0.76% to 1.58% were obtained at Feekes 5 (Table 3). In Parma and Rupert sites, the highest N concentration (%) were obtained at fertilizer N rate of 336 kg N ha<sup>-1</sup> while in Ashton site, the highest N concentration (%) was obtained at fertilizer N rate of 252 kg N ha<sup>-1</sup>. Similarly, at Feekes 10 a wide range of N concentration (%) ranged from 0.33% to 0.70% were obtained. In all sites, the highest N concentration (%) were obtained at fertilizer N rate of 336 kg N ha<sup>-1</sup>. Generally, observed variations in N concentration were due to differences in climate, inherent soil fertility, and N rate applications. When comparing N fertilizer rate applications, in Parma and Rupert sites at both Feekes 5 and Feekes 10 growth stages, the rate of N fertilizer significantly affected N concentration. In Ashton site at Feekes 5, the rate of N fertilizer did not make significant changes in N concentration while at Feekes 10, the rate of N fertilizer made significant changes in N concentration.

Across growth stages, applied N rates and locations, plant N concentration decreased at all locations (Table 3 and Table 4). These results indicate the “dilution effect” as the crop matured, as described in previous studies [34] [35]. Plant N concentration was lower for Ashton site at both Feekes 5 and Feekes 10 growth stages, compared to other sites. As all study sites were irrigated and had similar soil type (silt loam), the differences between plant N concentration for the same N application rates may be associated with differences in the amount of plant available water. Ashton site received 15 cm more precipitation compared to Rupert, and 17 cm more than Parma. The similar result was obtained for the semiarid grassland by Lu [36]. In that study, Lu showed that water additions significantly interacted to affect plant N uptake and N concentrations at the community level.

**Table 3.** Plant N concentration at Feekes 5, as affected by the applied N fertilizer rate

| N rate (kg N ha <sup>-1</sup> ) | Mean Plant N Concentration (%) |        |        |
|---------------------------------|--------------------------------|--------|--------|
|                                 | Parma                          | Ashton | Rupert |
| 0                               | 0.76d                          | 0.76a  | 1.31.b |

|     |       |       |        |
|-----|-------|-------|--------|
| 84  | 1.05c | 0.80a | 1.41b  |
| 168 | 1.18b | 0.79a | 1.44ab |
| 252 | 1.17b | 0.83a | 1.39b  |
| 336 | 1.32a | 0.82a | 1.58a  |

Means within each column followed by the same letter are not significantly different at  $p < 0.01$ , as determined by the Duncan's multiple range test.

**Table 4.** Plant N concentration at Feekes 10, as affected by the applied N fertilizer rate.

| N rate ( $\text{kg N ha}^{-1}$ ) | Mean Plant N Concentration (%) |        |        |
|----------------------------------|--------------------------------|--------|--------|
|                                  | Parma                          | Ashton | Rupert |
| 0                                | 0.33c                          | 0.35b  | 0.33c  |
| 84                               | 0.41bc                         | 0.39ab | 0.41bc |
| 168                              | 0.65a                          | 0.45a  | 0.65a  |
| 252                              | 0.49b                          | 0.35b  | 0.49b  |
| 336                              | 0.70a                          | 0.45a  | 0.70a  |

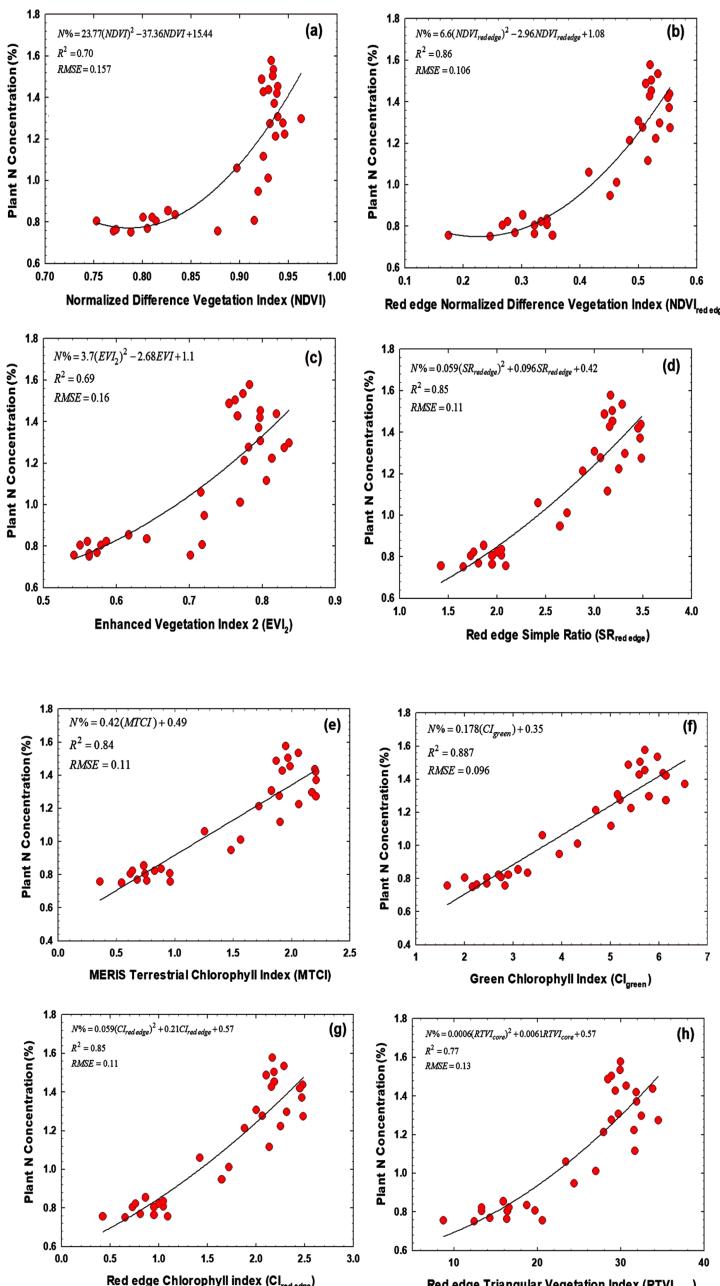
Means within each column followed by the same letter are not significantly different at  $p < 0.01$ , as determined by the Duncan's multiple range test.

## Plant N Concentration Models Development and Validation

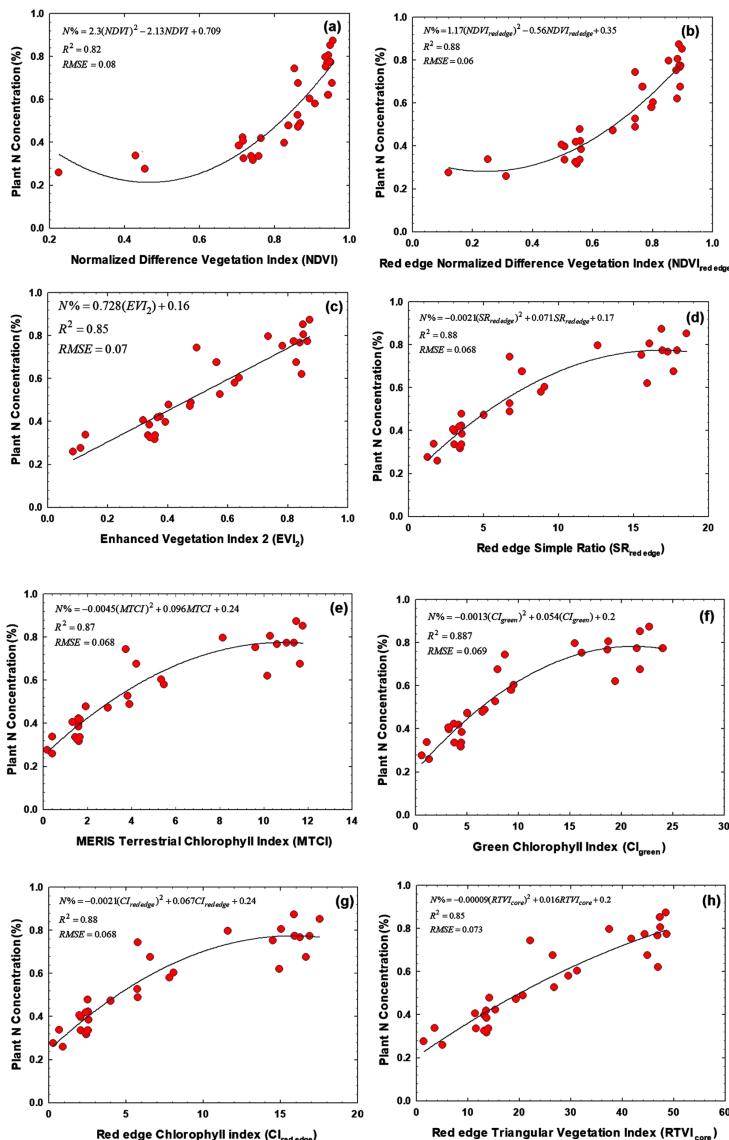
We used the training data set for establishing separate plant N content predictive models using UAV based VIs for Feekes 5 and Feekes 10 separately. The training data set includes a wide range of plant N content values due to differences in N application rates. Eight models were developed using the vegetation indices NDVI, NDVI<sub>red edge</sub>, EVI2, SR<sub>red edge</sub>, MTCI, CI<sub>green</sub>, CI<sub>red edge</sub> and RTVI<sub>core</sub>. Figure 3 and Figure 4 show the best relationship fit between N concentration and each UAV based VI for Feekes 5 and Feekes 10, respectively. At Feekes 5 (Figure 3), the  $R^2$  of these models ranged from 0.69 to 0.88 and RMSE ranged from 0.096 to 0.16. At this growth stage, the highest  $R^2$  between VIs and N concentrations was obtained for the CI<sub>green</sub>. Also, the developed model based on this index showed the lowest RMSE with N concentration. The best relationship fit between N concentration and most VIs at Feekes 5 were quadratic (Figure 3). CI<sub>green</sub> and MTCI were the

only VIs which their best relationship fit with N concentration were linear. At Feekes 10 (Figure 4), the  $R^2$  of the developed models ranged from 0.82 to 0.88 and RMSE ranged from 0.06 to 0.08. At this growth stage, the largest  $R^2$  between VIs and N concentrations again was obtained for  $CI_{green}$  while the developed model based on  $NDVI_{red\ edge}$  had the lowest RMSE with N concentration. The best relationship fit between N concentration and all VIs (except  $EVI_2$ ) at Feekes 10 were quadratic (Figure 4).  $EVI_2$  was the only VIs for which its' best relationship fit with N concentration was linear.

All UAV based VIs used in this study showed strong positive correlations with N concentration. In other words, all UAV based VIs used in this study were good indicators of spring wheat plant N concentration for both Feekes 5 and Feekes 10 growth stages. At Feekes 5 (Figure 3), red radiation based UAV indices (NDVI and EVI2) had lower  $R^2$  and higher RMSE as compared with other green and red edge based UAV indices. At this stage (lower fractional vegetation cover), soil background influence on research plots' reflectance could be strong and could negatively affect the red based VIs accuracy [37]. Red edge based VIs could minimize the soil reflectance and isolate crop signal from soil reflectance as a function of canopy cover changes [13]. This suggests that applying red edge based or green based VIs from UAVs data can improve plant N concentration prediction compared to the red based UAV VIs at Feekes 5 of wheat growth stage. At this stage,  $CI_{green}$  showed the highest  $R^2$  and lowest RMSE which suggests that the green based VIs can be a better indicator of plant N concentration than the red edge based VIs. This result is in consistent with the result of the previous study conducted by Li [38] who showed that red edge based VIs were more effective for N estimation at earlier growth stage of wheat. In addition,  $CI_{green}$  showed linear relationship with plant N concentration, which means sensitivity of the model did not change due to the wide range of variation in plant N concentration; it is straightforward to invert them between  $CI_{green}$  and plant N concentration to obtain a synoptic measure of N concentration. All the red edge based VIs showed comparable performance with similar  $R^2$  and RMSE values at Feekes 5. At Feekes 10 growth stage (Figure 4), all UAV based VIs used in the study performed very well with similar  $R^2$  and RMSE values. At this stage, the performance of NDVI and EVI2 for plant N concentration improved. All other UAV based VIs had similar performance compare to Feekes 5. Similar results have been reported by previous studies as well [38]. At Feekes 10, the crop canopy had fully developed, and soil background effect on research plots' reflectance had been reduced, so red based and red edge based VIs showed similar performance.



**Figure 3.** Relationships between measured plant N content (%) vs. (a) NDVI, (b)  $NDVI_{red\ edge}$ , (c)  $EVI_2$ , (d)  $SR_{red\ edge}$ , (e) MTCI, (f)  $CI_{green}$ , (g)  $CI_{red\ edge}$  and (h)  $RTVI_{core}$  at Feekes 5 growth stage.



**Figure 4.** Relationships between measured plant N content (%) vs. (a) NDVI, (b) NDVI<sub>red edge</sub>, (c) EVI<sub>2</sub>, (d) SR<sub>red edge</sub>, (e) MTCI, (f) CI<sub>green</sub>, (g) CI<sub>red edge</sub> and (h) RTVI<sub>core</sub> at Feekes 10 growth stage.

The performance of the developed models in the previous step were evaluated using test data set for Feekes 5 and Feekes 10, separately. For this purpose, we used the developed models in previous step to estimate the N concentration. Table 5 and Table 6 show the results of comparison between

measured N concentration and predicted N concentration retrieved from developed model in previous step for Feeks 5 and Feekes 10, respectively. At Feekes 5 (Table 5), the  $R^2$  of the relationship between measured and predicted N concentration ranged from 0.67 to 0.84 and RMSE ranged from 0.147 to 0.196. At this growth stage, the lowest RMSE between measured and predicted N concentrations were obtained using the developed models based on  $RTVI_{core}$  and  $NDVI_{red\ edge}$  (Table 5). At this growth stage, all the developed models performed similar in terms of  $R^2$  (Table 5). The results of a Student's t test showed that for the developed models based on  $NDVI_{red\ edge}$ ,  $SR_{red\ edge}$  and  $CI_{red\ edge}$  the slopes of those regression lines were not significantly different from 1 ( $t = 0.0279$ ;  $t = 0.43947$ ;  $t = 0.43706$  respectively) (Table 7), while a similar test showed that the intercepts of those lines were not significantly different from 0 ( $t = 1.026$ ;  $t = 0.730$ ;  $t = 0.731$ , respectively) (Table 7). Thus, one could conclude that these regression lines were not significantly different from the 1:1 line. These results indicated that the developed model based on  $NDVI_{red\ edge}$  can predict the plant N concentration at Feekes 5 with best accuracy compared to other developed models (Figure 4(a)).

**Table 5.** The results of algorithm cross validation for estimating plant N concentration (N%) at Feekes 5 growth stage. Best fit functions, determination coefficients ( $R^2$ ) and root mean square errors (RMSE) of plant N concentration estimation are given for eight vegetation indices.

| VI                 | Estimated N% = a (measured N%) + b | $R^2$ | RMSE (%) |
|--------------------|------------------------------------|-------|----------|
| NDVI               | $0.7265x + 0.3464$                 | 0.67  | 0.168    |
| $NDVI_{red\ edge}$ | $1.026x + 0.0815$                  | 0.84  | 0.148    |
| EVI <sub>2</sub>   | $0.8167x + 0.2694$                 | 0.78  | 0.165    |
| $SR_{red\ edge}$   | $1.0419x + 0.0799$                 | 0.84  | 0.179    |
| $CI_{green}$       | $1.2495x - 0.1527$                 | 0.84  | 0.196    |
| $CI_{red\ edge}$   | $1.0417x + 0.08$                   | 0.84  | 0.179    |
| MTCI               | $0.7932x + 0.3497$                 | 0.84  | 0.160    |
| $RTVI_{core}$      | $0.8434x + 0.2811$                 | 0.84  | 0.147    |

**Table 6.** The results of algorithm cross validation for estimating plant N concentration (N%) at Feekes 10 growth stage. Best fit functions, determination coefficients ( $R^2$ ) and root mean square errors (RMSE) of plant N concentration estimation are given for eight vegetation indices.

| VI <sub>s</sub>          | Estimated N% = a<br>(measured N%) + b | R <sup>2</sup> | RMSE (%) |
|--------------------------|---------------------------------------|----------------|----------|
| NDVI                     | 0.728x + 0.1509                       | 0.72           | 0.10     |
| NDVI <sub>red edge</sub> | 0.769 + 0.143                         | 0.74           | 0.091    |
| EVI <sub>2</sub>         | 0.7778x + 0.107                       | 0.73           | 0.095    |
| SR <sub>red edge</sub>   | 0.756x + 0.147                        | 0.75           | 0.091    |
| CI <sub>green</sub>      | 0.891x + 0.03                         | 0.76           | 0.090    |
| CI <sub>red edge</sub>   | 0.752x + 0.141                        | 0.75           | 0.091    |
| MTCI                     | 0.785x + 0.121                        | 0.75           | 0.090    |
| RTVI <sub>core</sub>     | 0.778x + 0.163                        | 0.72           | 0.104    |

**Table 7.** Regressions' analysis parameter, t value, to determine if the slope and the intercept of the regressions were significantly different from 1 and 0, respectively at Feekes 5.

| VI <sub>s</sub> | NDVI         | NDVI <sub>red edge</sub> | EVI <sub>2</sub> | SR <sub>red edge</sub> | CI <sub>green</sub> | CI- <sub>red edge</sub> | MTCI        | RTVI <sub>core</sub> |
|-----------------|--------------|--------------------------|------------------|------------------------|---------------------|-------------------------|-------------|----------------------|
| Slop            | -2.53<br>856 | 0.0279                   | -2.2772          | 0.43947                | 2.1633              | 0.43<br>706             | -1.9<br>776 | -1.7958              |
| Intercept       | 2.8066       | 1.026                    | 2.7678           | 0.730                  | -1.1557             | 0.730                   | 2.9279      | 2.8123               |

All t values were determined at 46 df and  $\alpha = 0.05$ .

Similarly, at Feekes 10, the R<sup>2</sup> of the relationship between measured and predicted N concentration ranged from 0.72 to 0.73 and RMSE ranged from 0.09 to 0.104. At this growth stage, the lowest RMSE between measured and predicted N concentrations were obtained using the developed models based on CI<sub>green</sub> and MTCI (Table 6). At this stage, all developed models had weaker performance in term of R<sup>2</sup>, but all models had smaller RMSE compared to Feekes 5. The results of a Student's t test showed that only for the model developed based on CI<sub>green</sub>, the slope of the regression line was not significantly different from 1 ( $t = -1.15722$ , 46 df,  $\alpha = 0.05$ ) (Table 8), while a similar test showed that the intercept of that line was not significantly different from 0 ( $t = 0.6554$ , 46 df,  $\alpha = 0.05$ ) (Table 8). Thus, one could conclude this regression line was not significantly different from the 1:1 line. These results indicate that the model developed based on CI<sub>green</sub> can predict the plant N concentration at Feekes 10 with greatest accuracy compared to other models, and can be successfully used to estimate plant N concentration of wheat crop in the future (Figure 4(b)).

In cross-validation process, results showed that at Feekes 5 developed model based on  $NDVI_{red\ edge}$  can predict the plant N concentration with best accuracy compared to other developed models. The developed model based on  $NDVI_{red\ edge}$  slightly underestimated the plant N concentration at lower values while overestimated the plant N concentration at higher values (Figure 5(a)). One possible reason for this is the soil type that was slightly different for each study site; it is possible that the total canopy reflectance was affected by slightly different background reflectance values at each location [39]. At Feekes 10, the developed model based on  $CI_{green}$  predicted the plant N concentration with highest accuracy compared to other models. The developed model based on  $CI_{green}$  at this growth stage did not have over- or underestimation issue as the crop canopy had been fully developed, which minimized the effect of different background reflectance soil values (Figure 5(b)).

At different growth stages, UAV based VIs showed different behaviors for estimating plant N concentration. Therefore, growth stage-specific models would be preferred for estimating plant N concentration. Mid-season plant N content estimation could significantly improve the opportunity for farmers to intervene with strategic fertilizer management. Ideally, the choice of an index for canopy N measurement should not depend on the geographical location where measurements are made.

The VIs that minimize the soil reflectance, such as red edge based VIs, have the best performance across different locations with different soil type. These kinds of VIs could be used to map N content across farmer fields without calibrations, allowing them to target N applications. The results mean that adding red edge band to UAV sensors can improve plant N concentration monitoring and estimation.

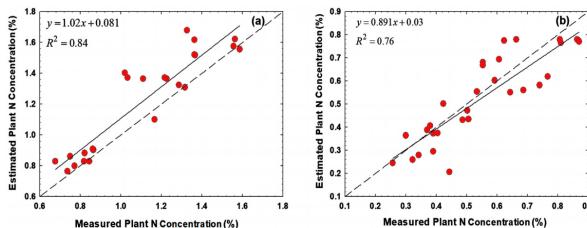
The main concern about developed models for plant N concentration estimation is that these models are crop-specific. Each crop has its' own unique spectral signature; thus, the same models cannot be used for various crops. Changes in plant characteristics (such as breeding improvements) may require the development of additional algorithms.

Mixed results about the effect of wheat cultivars on spectral reflectance have been reported in literature. Sembiring et al [40] found that wheat varieties did not have significant effect on spectral measurements. On the other hand, Sultana et al [41] documented that spectral reflectance (NDVI) values have varied significantly for a wide range of cultivars treated with the same N rates.

**Table 8.** Regressions' analysis parameter, t value, to determine if the slope and the intercept of the regressions were significantly different from 1 and 0, respectively at Feekes 5.

| VI <sub>s</sub> | NDVI    | ND-VI <sub>red edge</sub> | EVI <sub>2</sub> | SR <sub>red edge</sub> | CI <sub>green</sub> | CI <sub>red edge</sub> | MTCI    | RTVI <sub>core</sub> |
|-----------------|---------|---------------------------|------------------|------------------------|---------------------|------------------------|---------|----------------------|
| Slop            | -2.9701 | -2.9269                   | -2.7<br>852      | -3.2624                | -1.1572             | -2.876                 | -2.6159 | -2.4721              |
| Intercept       | 2.8194  | 3.1016                    | 2.2<br>996       | 3.3815                 | 0.6554              | 2.809                  | 2.5192  | 3.1225               |

All t values were determined at 46 df and  $\alpha = 0.05$ .



**Figure 5.** Cross validation of developed algorithms based on (a) NDVI<sub>red edge</sub> at Feekes 5 and (b) CI<sub>green</sub> at Feekes 10 in estimating plant N concentration of wheat crop in test data sets.

Remotely sensed VIs have been extensively used to quantify wheat crop N status. The UAV technology appears to provide a good complement to the current remote sensing platforms for N monitoring in wheat by capturing low-cost, high resolution images. These UAV technologies can bring a unique perspective to N management in wheat by providing valuable information on wheat N status. Time, labor and money can be saved using UAV data in crop monitoring.

Results presented in this paper show that high resolution images acquired with UAVs are a useful data source for in-season wheat crop N concentration estimation. At Feekes 5 growth stage, red edge and green based VIs had higher correlation with plant N concentration compared to red based VIs because red edge based VIs can reduce the soil background effect on crop reflectance. At Feekes 10 growth stage, all calculated VIs showed high correlation with plant N concentration, and there were no significant differences between red and red edge based VIs' performance. At this stage, crop canopy has been fully developed, and soil reflectance did

not have strong effect on the reflectance of research plots. At Feekes 5, the plant N concentration estimated based on  $NDVI_{red\ edge}$  showed 1:1 correlation with N concentration measured in the lab. At Feekes 10, the estimated and measured N concentration were highly correlated for all developed models, but the model based on  $CI_{green}$  was the only model that had a 1:1 correlation between estimated and measured plant N concentration. The observed high correlation between UAV based VIs with plant N concentration indicates the applicability of UAV for in-season data collection from agricultural fields.

## CONCLUSIONS

Remotely sensed VIs have been extensively used to quantify wheat crop N status. The UAV technology appears to provide a good complement to the current remote sensing platforms for N monitoring in wheat by capturing low-cost, high resolution images. These UAV technologies can bring a unique perspective to N management in wheat by providing valuable information on wheat N status. Time, labor and money can be saved using UAV data in crop monitoring. Results presented in this paper show that high resolution images acquired with UAVs are a useful data source for in-season wheat crop N concentration estimation. At Feekes 5 growth stage, red edge and green based VIs had higher correlation with plant N concentration compared to red based VIs because red edge based VIs can reduce the soil background effect on crop reflectance. At Feekes 10 growth stage, all calculated VIs showed high correlation with plant N concentration, and there were no significant differences between red and red edge based VIs' performance. At this stage, crop canopy has been fully developed, and soil reflectance did not have strong effect on the reflectance of research plots. At Feekes 5, the plant N concentration estimated based on  $NDVI_{red\ edge}$  showed 1:1 correlation with N concentration measured in the lab. At Feekes 10, the estimated and measured N concentration were highly correlated for all developed models, but the model based on  $CI_{green}$  was the only model that had a 1:1 correlation between estimated and measured plant N concentration. The observed high correlation between UAV based VIs with plant N concentration indicates the applicability of UAV for in-season data collection from agricultural fields.

## ACKNOWLEDGEMENTS

This study was supported in part by the Idaho Wheat Commission.

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

## REFERENCES

1. Biswas, D.K. and Ma, B.L. (2016) Effect of Nitrogen Rate and Fertilizer Nitrogen Source on Physiology, Yield, Grain Quality, and Nitrogen Use Efficiency in Corn. *Canadian Journal of Plant Science*, 96, 392-403. <https://doi.org/10.1139/cjps-2015-0186>
2. Jain, N., Ray, S.S., Singh, J.P. and Panigrahy, S. (2007) Use of Hyperspectral Data to Assess the Effects of Different Nitrogen Applications on a Potato Crop. *Precision Agriculture*, 8, 225-239. <https://doi.org/10.1007/s11119-007-9042-0>
3. Walsh, O.S., Shafian, S. and Christiaens, R.J. (2018) Evaluation of Sensor-Based Nitrogen Rates and Sources in Wheat. *International Journal of Agronomy*, 2018, Article ID: 5670479. <https://doi.org/10.1155/2018/5670479>
4. Ramoelo, A., Skidmore, A.K., Schlerf, M., Heitkönig, I.M., Mathieu, R. and Cho, M.A. (2013) Savanna Grass Nitrogen to Phosphorous Ratio Estimation Using Field Spectroscopy and the Potential for Estimation with Imaging Spectroscopy. *International Journal of Applied Earth Observation and Geoinformation*, 23, 334-343. <https://doi.org/10.1016/j.jag.2012.10.009>
5. Li, F., Mistele, B., Hu, Y., Chen, X. and Schmidhalter, U. (2014) Reflectance Estimation of Canopy Nitrogen Content in Winter Wheat Using Optimised Hyperspectral Spectral Indices and Partial Least Squares Regression. *European Journal of Agronomy*, 52, 198-209. <https://doi.org/10.1016/j.eja.2013.09.006>
6. Reay, D.S., Davidson, E.A., Smith, K.A., Smith, P., Melillo, J.M., Dentener, F. and Crutzen, P.J. (2012) Global Agriculture and Nitrous Oxide Emissions. *Nature Climate Change*, 2, 410. <https://doi.org/10.1038/nclimate1458>
7. Ravishankara, A.R., Daniel, J.S. and Portmann, R.W. (2009) Nitrous Oxide (N<sub>2</sub>O): The Dominant Ozone-Depleting Substance Emitted in the 21st Century. *Science*, 326, 123-125.
8. Bao, Y., Xu, K., Min, J. and Xu, J. (2013) Estimating Wheat Shoot Nitrogen Content at Vegetative Stage from in Situ Hyperspectral Measurements. *Crop Science*, 53, 2063-2071. <https://doi.org/10.2135/cropsci2013.01.0012>
9. Schlemmer, M., Gitelson, A., Schepers, J., Ferguson, R., Peng, Y., Shanahan, J. and Rundquist, D. (2013) Remote Estimation of

- Nitrogen and Chlorophyll Contents in Maize at Leaf and Canopy Levels. International Journal of Applied Earth Observation and Geoinformation, 25, 47-54. <https://doi.org/10.1016/j.jag.2013.04.003>
- 10. Baret, F., Houles, V. and Guérif, M. (2007) Quantification of Plant Stress Using Remote Sensing Observations and Crop Models: The Case of Nitrogen Management. Journal of Experimental Botany, 58, 869-880. <https://doi.org/10.1093/jxb/erl231>
  - 11. Filella, I., Serrano, L., Serra, J. and Penuelas, J. (1995) Evaluating Wheat Nitrogen Status with Canopy Reflectance Indices and Discriminant Analysis. Crop Science, 35, 1400-1405. <https://doi.org/10.2135/cropsc1995.0011183X003500050023x>
  - 12. Sage, R.F., Pearcy, R.W. and Seemann, J.R. (1987) The Nitrogen Use Efficiency of C<sub>3</sub> and C<sub>4</sub> Plants: III. Leaf Nitrogen Effects on the Activity of Carboxylating Enzymes in *Chenopodium album* (L.) and *Amaranthus retroflexus* (L.). Plant Physiology, 85, 355-359. <https://doi.org/10.1104/pp.85.2.355>
  - 13. Cammarano, D., Fitzgerald, G., Basso, B., O'Leary, G., Chen, D., Grace, P. and Fiorentino, C. (2011) Use of the Canopy Chlorophyll Content Index (CCCI) for Remote Estimation of Wheat Nitrogen Content in Rainfed Environments. Agronomy Journal, 103, 1597-1603. <https://doi.org/10.2134/agronj2011.0124>
  - 14. Xuefeng, L., Qiang, L., Shaolan, H., Shilai, Y., Deyu, H., Zhitao, W., Lie, D., et al. (2016) Estimation of Carbon and Nitrogen Contents in Citrus Canopy by Low-Altitude Remote Sensing. International Journal of Agricultural and Biological Engineering, 9, 149.
  - 15. Herrmann, I., Karnieli, A., Bonfil, D.J., Cohen, Y. and Alchanatis, V. (2010) SWIR-Based Spectral Indices for Assessing Nitrogen Content in Potato Fields. International Journal of Remote Sensing, 31, 5127-5143. <https://doi.org/10.1080/01431160903283892>
  - 16. Clevers, J.G. and Gitelson, A.A. (2013) Remote Estimation of Crop and Grass Chlorophyll and Nitrogen Content Using Red-Edge Bands on Sentinel-2 and -3. International Journal of Applied Earth Observation and Geoinformation, 23, 344-351. <https://doi.org/10.1016/j.jag.2012.10.008>
  - 17. Gitelson, A. and Merzlyak, M.N. (1994) Spectral Reflectance Changes Associated with Autumn Senescence of *Aesculus hippocastanum* L. and *Acer platanoides* L. Leaves. Spectral Features and Relation to Chlorophyll Estimation. Journal of Plant Physiology, 143, 286-292.

[https://doi.org/10.1016/S0176-1617\(11\)81633-0](https://doi.org/10.1016/S0176-1617(11)81633-0)

18. Gitelson, A.A., Gritz, Y. and Merzlyak, M.N. (2003) Relationships between Leaf Chlorophyll Content and Spectral Reflectance and Algorithms for Non-Destructive Chlorophyll Assessment in Higher Plant Leaves. *Journal of Plant Physiology*, 160, 271-282. <https://doi.org/10.1078/0176-1617-00887>
19. Dash, J. and Curran, P.J. (2004) The MERIS Terrestrial Chlorophyll Index.
20. Wang, Y., Liao, Q., Yang, G., Feng, H., Yang, X. and Yue, J. (2016) Comparing Broad-Band and Red Edge-Based Spectral Vegetation Indices to Estimate Nitrogen Concentration of Crops Using Casi Data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 137-143. <https://doi.org/10.5194/isprsarchives-XLI-B7-137-2016>
21. Maes, W.H., Huete, A.R. and Steppe, K. (2017) Optimizing the Processing of UAV-Based Thermal Imagery. *Remote Sensing*, 9, 476. <https://doi.org/10.3390/rs9050476>
22. Pádua, L., Vanko, J., Hruška, J., Adão, T., Sousa, J.J., Peres, E. and Morais, R. (2017) UAS, Sensors, and Data Processing in Agroforestry: A Review towards Practical Applications. *International Journal of Remote Sensing*, 38, 2349-2391. <https://doi.org/10.1080/01431161.2017.1297548>
23. Simelli, I. and Tsagaris, A. (2015) The Use of Unmanned Aerial Systems (UAS) in Agriculture. *HAICTA*, Kavala, 17-20 September 2015, 730-736.
24. Lu, J., Miao, Y., Huang, Y., Shi, W., Hu, X., Wang, X. and Wan, J. (2015) Evaluating an Unmanned Aerial Vehicle-Based Remote Sensing System for Estimation of Rice Nitrogen Status. 4th International Conference on Agro-Geoinformatics, Istanbul, 20-24 July 2015, 198-203.
25. Caturegli, L., Corniglia, M., Gaetani, M., Grossi, N., Magni, S., Migliazzi, M., Raffaelli, M., et al. (2016) Unmanned Aerial Vehicle to Estimate Nitrogen Status of Turfgrasses. *PLoS ONE*, 11, e0158268. <https://doi.org/10.1371/journal.pone.0158268>
26. Hunt, E.R. and Rondon, S.I. (2017) Detection of Potato Beetle Damage Using Remote Sensing from Small Unmanned Aircraft Systems. *Journal of Applied Remote Sensing*, 11, Article ID: 026013. <https://doi.org/10.1119/1.4982720>

- doi.org/10.1117/1.JRS.11.026013
- 27. Mission Planner Home. <http://ardupilot.org/planner>
  - 28. Shi, Y., Thomasson, J.A., Murray, S.C., Pugh, N.A., Rooney, W.L., Shafian, S., Rana, A., et al. (2016) Unmanned Aerial Vehicles for High-Throughput Phenotyping and Agronomic Research. *PLoS ONE*, 11, e0159781. <https://doi.org/10.1371/journal.pone.0159781>
  - 29. Official Methods of Analysis of AOAC International. [http://www.aoac.org/aoac\\_prod\\_imis/aoac/publications/official\\_methods\\_of\\_analysis/aoac\\_member/pubs/oma/aoac\\_official\\_methods\\_of\\_analysis.aspx?hkey=5142c47 8-ab50-4856-8939-a7a491756f48](http://www.aoac.org/aoac_prod_imis/aoac/publications/official_methods_of_analysis/aoac_member/pubs/oma/aoac_official_methods_of_analysis.aspx?hkey=5142c47 8-ab50-4856-8939-a7a491756f48)
  - 30. Rouse Jr., J., Haas, R.H., Schell, J.A. and Deering, D.W. (1974) Monitoring Vegetation Systems in the Great Plains with ERTS.
  - 31. Huete, A., Didan, K., Miura, T., Rodriguez, E.P., Gao, X. and Ferreira, L.G. (2002) Overview of the Radiometric and Biophysical Performance of the MODIS Vegetation Indices. *Remote Sensing of Environment*, 83, 195-213. [https://doi.org/10.1016/S0034-4257\(02\)00096-2](https://doi.org/10.1016/S0034-4257(02)00096-2)
  - 32. Gitelson, A.A., Vina, A., Ciganda, V., Rundquist, D.C. and Arkebauer, T.J. (2005) Remote Estimation of Canopy Chlorophyll Content in Crops. *Geophysical Research Letters*, 32, L08403. <https://doi.org/10.1029/2005GL022688>
  - 33. Nicolas, T., Philippe, V. and Huang, W.J. (2010) New Index for Crop Canopy Fresh Biomass Estimation. *Spectroscopy and Spectral Analysis*, 30, 512-517.
  - 34. Plénet, D. and Lemaire, G. (1999) Relationships between Dynamics of Nitrogen Uptake and Dry Matter Accumulation in Maize Crops. Determination of Critical N Concentration. *Plant and Soil*, 216, 65-82. <https://doi.org/10.1023/A:1004783431055>
  - 35. Muñoz-Huerta, R.F., Guevara-Gonzalez, R.G., Contreras-Medina, L.M., TorresPacheco, I., Prado-Olivarez, J. and Ocampo-Velazquez, R.V. (2013) A Review of Methods for Sensing the Nitrogen Status in Plants: Advantages, Disadvantages and Recent Advances. *Sensors*, 13, 10823-10843. <https://doi.org/10.3390/s130810823>
  - 36. Lü, X.T., Dijkstra, F.A., Kong, D.L., Wang, Z.W. and Han, X.G. (2014) Plant Nitrogen Uptake Drives Responses of Productivity to Nitrogen and Water Addition in a Grassland. *Scientific Reports*, 4, Article No. 4817. <https://doi.org/10.1038/srep04817>
  - 37. Viña, A., Gitelson, A.A., Nguy-Robertson, A.L. and Peng, Y. (2011)

Comparison of Different Vegetation Indices for the Remote Assessment of Green Leaf Area Index of Crops. *Remote Sensing of Environment*, 115, 3468-3478. <https://doi.org/10.1016/j.rse.2011.08.010>

38. Li, F., Miao, Y., Hennig, S.D., Gnyp, M.L., Chen, X., Jia, L. and Bareth, G. (2010) Evaluating Hyperspectral Vegetation Indices for Estimating Nitrogen Concentration of Winter Wheat at Different Growth Stages. *Precision Agriculture*, 11, 335-357. <https://doi.org/10.1007/s11119-010-9165-6>
39. Todd, S.W. and Hoffer, R.M. (1998) Responses of Spectral Indices to Variations in Vegetation Cover and Soil Background. *Photogrammetric Engineering and Remote Sensing*, 64, 915-922.
40. Sembiring, H., Raun, W.R., Johnson, G.V., Stone, M.L., Solie, J.B. and Phillips, S.B. (1998) Detection of Nitrogen and Phosphorus Nutrient Status in Winter Wheat Using Spectral Radiance. *Journal of Plant Nutrition*, 21, 1207-1233. <https://doi.org/10.1080/01904169809365478>
41. Sultana, S.R., Ali, A., Ahmad, A., Mubeen, M., Zia-Ul-Haq, M., Ahmad, S., Jaafar, H.Z., et al. (2014) Normalized Difference Vegetation Index as a Tool for Wheat Yield Estimation: A Case Study from Faisalabad, Pakistan. *The Scientific World Journal*, 2014, Article ID: 725326. <https://doi.org/10.1155/2014/725326>



# **CHAPTER**

# **17**

## **Research and Teaching Applications of Remote Sensing Integrated with GIS: Examples from the Field**

---

**Laura C. Loyola<sup>1</sup>, Jason T. Knowles<sup>1,2</sup>, Andrew J. Marx<sup>1</sup>, Ryan McAlinden<sup>1,3</sup>, Steven D. Fleming<sup>1</sup>**

<sup>1</sup>Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA.

<sup>2</sup>GeoAcuity, Los Angeles, CA, USA.

<sup>3</sup>Institute for Creative Technologies, University of Southern California, Los Angeles, CA, USA.

### **ABSTRACT**

Remote sensing is used in the Spatial Sciences Institute (SSI) across the full spectrum of the organization's teaching and research initiatives. From

---

**Citation:** Loyola, L. , Knowles, J. , Marx, A. , McAlinden, R. and Fleming, S. (2019) "Research and Teaching Applications of Remote Sensing Integrated with GIS: Examples from the Field". *Journal of Geographic Information System*, **11**, 670-682. doi: 10.4236/jgis.2019.116041.

**Copyright:** © 2019 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

undergraduate to graduate classes that utilize unmanned aerial systems (UAS) in data collection and graduate courses that incorporate remote sensing for a variety of applications, including Earth observation, to applied research via the Human Security and Geospatial Intelligence (HSGI) Lab projects and work being done with One World Terrain (OWT) at the Institute for Creative Technologies (ICT) to build a fully geo-referenced 3D planetary model, SSI recognizes the need to educate students about remote sensing techniques. This paper discusses faculty involvement in conducting a pilot study for the Catalina Island Conservancy (CIC) using UAS to survey local bison and deer populations. The team utilized an autonomous fixed-wing UAS with a thermal payload to collect data for a semi-automated detection workflow integrated within a GIS successfully identifying both deer and bison. Additionally, graduate students participate in a weeklong experiential learning opportunity on Catalina Island, CA during which they develop and conduct a research project integrating UAS and other remotely sensed data with primary data collection in a Geographic Information System (GIS). By extension, the Institute then reinforces that these educational opportunities, focused primarily on data acquisition, are instrumental in supporting the geographic information systems, science, and technology experiences in many diverse fields, including (but not limited to) human security, humanitarian relief, sustainable urban and rural planning, and public health.

**Keywords:** Unmanned Aerial Systems (UAS), Integrated Curriculum, Experiential Learning, Security, Sustainable Planning

## INTRODUCTION

Within spatial sciences, the University of Southern California (USC) Dornsife College of Letters, Arts and Sciences (Dornsife) Spatial Sciences Institute (SSI) recognizes the importance of remotely sensed data as an integral component of a Geographic Information System (GIS) and spatial analysis to a variety of disciplines. As is evident from this special issue alone, remotely sensed data integrated into a GIS can be applied to everything from environmental analysis to human rights monitoring in far to reach locales. We focus this paper on two research projects contributed by faculty and affiliate faculty that incorporate remotely sensed data, acquired both from satellite imagery and unmanned aerial systems, into GIS and virtual reality/augmented reality (VR/AR) to promote the well being of military and the monitoring of wildlife. These vastly differing projects underscore the variety of applications for remotely sensed data and different research that the

Spatial Sciences Institute undertakes, which also can be used in teaching and promoting the next generation of data acquisition and analysis specialists. Reinforcing concepts and scientific theories is best accomplished through active learning, when the creation of new knowledge occurs through the transformation of experience [1] [2]. We discuss these projects not only within the context of the integration of remote sensing with GIS, but also the broader context of integrating remote sensing into curricular advances within the SSI at both graduate and undergraduate levels. Situated in the heart of Los Angeles, CA, the online graduate programs in Geographic Information Sciences & Technology (GIST) at USC are currently the only US programs that conform to UNIGIS standards of design and delivery for distance learning in GIS and GISci. This places USC Dornsife SSI in a unique position at the forefront of curricular development.

Below, we briefly describe some of the work that Director of Modeling, Simulation, & Training at the Institute for Creative Technologies (ICT), Ryan McAlinden oversees, and work that Professors Jason Knowles and Andrew Marx have initiated with the Catalina Island Conservancy (CIC), a non-profit organization that privately holds and manages over 88% of the land on Catalina Island. The USC-ICT research utilizes novel geospatial techniques and advances in the areas of collection, processing, storage and distribution of geospatial data, while the pilot for wildlife monitoring of local bison and deer populations via UAS-based high resolution visible (RGB) and thermal imaging has not previously been conducted on Catalina Island. We also describe how projects such as these have been incorporated into specific spatial sciences courses over the years to provide support for research and platforms for enhanced student learning. Aspects of the pedagogical approaches were previously presented at the GIS-Pro & Cal-GIS 2018 conference [3], therefore, here we focus on newer research and faculty capacity building. We also provide the workflow that we have undertaken to increase the faculty capacity in operating UAS and working with remotely sensed data. We acknowledge this may not work for all academic units, but encourage academic departments to follow similar curricular advances.

The next section of this paper presents background, methodology, and results of the wildlife monitoring pilot study on Catalina Island from 2018. Section three describes the innovative research and application of remote sensing integrated with GIS for AR/VR products and 3D planetary modeling done at USC-ICT. Both research studies are presented within the context of a Dornsife Spatial Sciences Institute graduate spatial data acquisition course and the potential for building faculty capacity in integrating remote sensing

with GIS in this course. The fourth section focuses on novel integration of remote sensing into the undergraduate and graduate curriculum within spatial sciences and presents a potential workflow for other organizations to build faculty capacity in this domain.

## **WILDLIFE MONITORING ON CATALINA ISLAND, PILOT 2018**

### **Background**

The Dornsife Spatial Sciences Institute has long run an online spatial data acquisition course that affords students the opportunity to experience a weeklong field data excursion based at USC Wrigley Institute for Environmental Studies (WIES) on Catalina Island. This course has evolved from working solely with handheld GPS units to formally include unmanned platforms. This evolution was bolstered when trained Remote License Pilots (RPL) pilots, Professors Jason Knowles and Andrew Marx, joined the faculty of SSI in late 2017. Additionally, spatial students and faculty have intermittently previously worked with the Catalina Island Conservancy on a variety of projects, mostly small-scale projects focused on areas near the USC WIES campus or just beyond, during the week-long excursion, with results remaining internal.

Having practical experience from previous work, both Knowles and Marx were eager to continue their remote sensing work through SSI; the already forged contacts at the CIC and the spatial data acquisition course provided the opportunity to formally link their research with the student curriculum and vice versa, to link the students with this practical application of remote sensing. In April 2018, Knowles and Marx worked with the CIC to develop and execute a pilot study for conducting wildlife surveys of local bison and deer populations utilizing UAS-based high resolution visible (RGB) and thermal imaging. The study looked at both the feasibility of utilizing fixed wing UAS-based imagery to identify wildlife, but also at workflow maximization; could gains be made in the efficiency and efficacy of airborne counting and cataloging of wildlife in comparison to more traditional on the ground field survey methods [4]. In addition, it was also anticipated that this methodology would be less obtrusive and invasive to the observed species due to the collection altitude of the UAS. This is a vital component to wildlife monitoring and an especially high priority for ethical organizations such as the CIC, which are the sole guardians of wildlife in

an area. The pilot study was done flying a fixed wing sense Fly Ebee Plus with SODA (RGB) and Thermo MAP payloads (Figure 1) resulting in both ultrahigh resolution aerial photography (at 1.12 in/2.84cm group sample distance) and thermal data capture (at 9.49 in/24.11cm ground sample distance). The CIC Director and two field biologists joined the collection team for the duration of the project.

## Methodology

The study area selected by the CIC was Middle Ranch Meadows on Catalina Island (Figure 2), located near the center of the island in a valley surrounded by rich topography making on the ground field visual observations difficult. Two days of flying were completed on April 4 and 5, with both high-resolution RGB and thermal cameras for a total of five flights. Both pilots (Knowles and Marx) were holders of Civil Certificate of Waiver Authorization (COA), for commercial operations via Federal Aviation Administration (FAA) Code of Federal Regulations (14CFR) Part 107. The April 4 flights were for orientation and equipment shakedown/calibration, while April 5 flights were for wildlife data capture. On April 5, the first flight was completed with the thermal package, taking off 30-minutes before sunrise (FAA earliest allowed). This early flight was conducted to maximize the temperature differential between the cold evening ground and the wildlife. Immediately after the first, one-hour flight, a second flight was performed with a RGB sensor over the same study area. This, along with ground truth performed by the CIC personnel using binoculars from a vantage point to identify wildlife, was used to confirm the presence of wildlife and validate potential signatures identified in the thermal imagery capture which is shown to be the most effective [5] [6]. Immediately following the flights, datasets were preprocessed in the field and saved to secondary backup systems. Once back from the field, datasets were processed overnight via commercial photogrammetry software (Pix4DMapper v4.2) and the following datasets were produced within 48-hours of capture and integrated within GIS (ArcMap v10.5):

- Ultra-high resolution RGB (visible) aerial photography (at 1.12 in/2.84cm ground sample distance);
- LAS Point cloud (Figure 3);
- Thermal surface model (at 9.49 in/24.11cm ground sample distance);
- Digital surface model (DSM); and
- 3D Textured mesh (Figure 4).

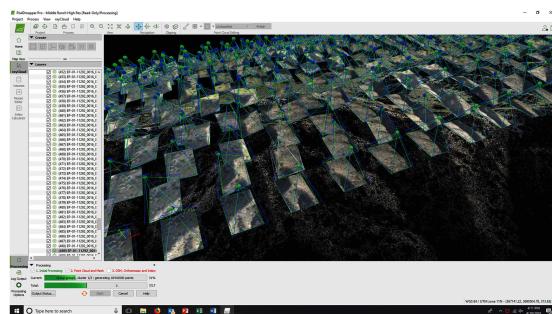
After processing the thermal imagery, it was added to a GIS, for manual analysis of the thermal imagery. Warm areas or literal “hotspots” from the thermal imagery capture were identified and correlated to the visible imagery and on the ground wildlife observations from the CIC field biologists.



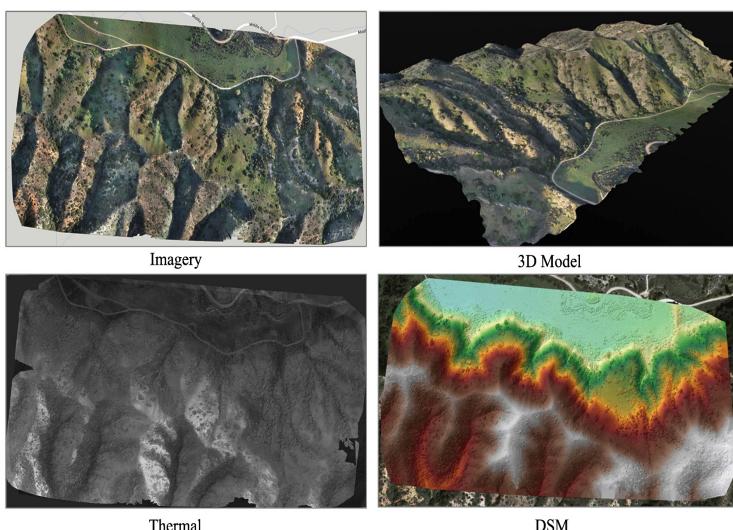
**Figure 1.** SenseFly eBee Plus, field set up prior to data collection.



**Figure 2.** Study area: flight coverage at Middle Ranch, Catalina Island, CA and Catalina Island with California coastline.



**Figure 3.** LAS point cloud processing in photogrammetry software.



**Figure 4.** UAS collected products: Orthoimage integrated with a Streets basemap (top left) and DSM integrated with Imagery basemap (bottom right) in GIS.

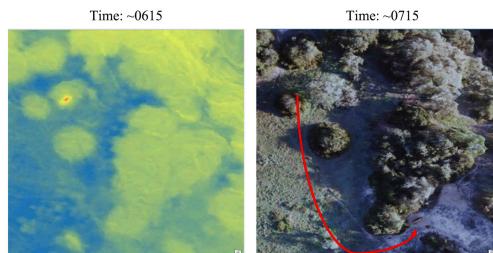
## Results

The preliminary results indicate that fixed wing UA appears to be a good platform for wildlife surveys both for its ability to cover large areas and host both RGB and thermal payloads. Initial surveys were a success with both bison and mule deer identified in the thermal imagery captured by the UAS survey (Figure 5 and Figure 6). Four mule deer and one bison were ultimately counted over the 1.41 km<sup>2</sup> study area.

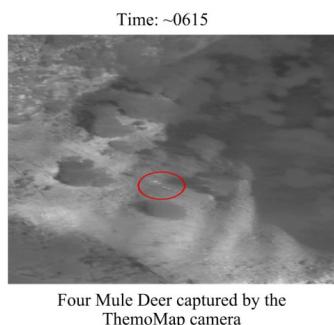
The field crew and the RGB imagery flown immediately following the thermal capture verified these signatures. The aerial collection methodology was unquestionably more efficient in terms of being able to cover more area (the eBee fixed wing has a flight time in excess of ~60-minutes and depending on the collection altitude can cover vast areas in a single flight) and visual observations of the wildlife during the flights indicated that there was no disturbance, with the animals seemingly unaware of the UAS high above them. In addition to the survey data, the derivative geospatial products produced by the programming process (Figure 4 above) were found to be extremely useful datasets for the CIC field biologists and GIS staff that would normally not be available.

## Summary

While we consider this initial pilot study a success, we believe that there can further improvements to the workflow and collection methodology. Collections and resultant thermal data capture would be significantly improved by flying predawn (or at night) in order to get a larger wildlife temperature differential signature between the environment and the wildlife. Even at sunrise, the head distribution was already much greater on east-facing slopes making detection of thermal signatures more difficult. Future studies will see the submission of an FAA Waiver to allow for predawn (or night) flying to maximize the temperature differential. Additionally, manual analysis of the thermal imagery, while doable, is time consuming and cumbersome. This process would benefit from the use of a scripted automation or semi-automated routine for entity detection [7]. A detection algorithm identify areas within the scenes where there are large temperature differences would enable the user to more rapidly identify and catalog the pertinent data from a large coverage area.



**Figure 5.** Thermal imagery from ThermoMap camera of bison (left) and RGB imagery (right) with movement of bison from time 0615 to 0715.



Four Mule Deer captured by the  
ThermoMap camera

**Figure 6.** Thermal imagery from ThermoMap camera of four mule deer (circled).

## AUTONOMOUS TERRAIN MODELING

Prior to fully incorporating UAS data collection and integration into the graduate level spatial data acquisition course, the Spatial Sciences Institute collaborated with the Institute for Creative Technologies to provide opportunities for UAS pilot training in difficult terrain and pilot workflow presentations that mutually benefited ICT pilots and students on Catalina Island. As such, we also describe the integration of aerial imagery performed by USC-ICT.

### Background

The USC-ICT's Terrain efforts focus on researching and prototyping capabilities that support a fully geo-reference 3D planetary model for use in the Army's next-generation training and simulation environments. USC-ICT research exploits new techniques and advances in the focus areas of collection, processing, storage and distribution of geospatial data to various runtime applications.

### Generalized Methodology

USC-ICT collects aerial images with Commercial off the Shelf (COTS) UAS using the ICT autonomous UAV path planning and imagery collection system. The software provides a user-friendly interface that encodes photogrammetry best practices. Unlike other commercially available UAV remote control software, the ICT solution was design for collecting aerial images that cover a large area of interest with multiple flights. Parameters that are required for data collections include a bounding box of the area of interest, flight altitude, the desired overlap between images, and camera orientation. An optimized flight path is then computed with these parameters and the imaging task can be automatically accomplished. With the acquired images, the 3D point clouds are reconstructed using commercial photogrammetry software.

The photogrammetric-generated point clouds/meshes (Figure 7) from a collection stage do not allow both user-level and system-level interaction, as they do not contain the semantic information to distinguish between objects. The workflow in previous works require either manually labeling the points into different parts, or re-training a new model for each new scene. USC-ICT has designed a fully automated process that utilizes deep learning to automatically extract relevant features and segmentation from point clouds. To train the feature attribution model, the points are first manually labeled

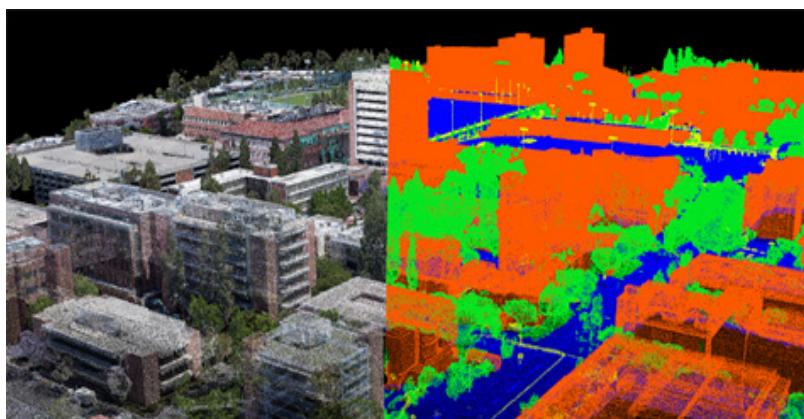
with the following labels: ground, man-made objects, vegetation, etc. These point clouds are then adapted to 3D voxel grids to produce a representation suitable for deep neural networks. ICT designed a simple yet effective 3D encoding and decoding network architecture based on 3D U-Net for point cloud segmentation.

During training, a straight and forward 3D data augmentation strategy was designed to perform rotation, translation, and crop on the input data at the same time.

This expands the amounts of data and allows for better generalization capabilities of the model. The resulting pipeline is able to extract building, ground, and vegetation in the raw point clouds automatically with high accuracy and produce accurate 3D models (Figure 8).

## Connection with Curriculum

With this technology and workflow in mind, Ryan McAlinden and pilot trainees demonstrated the mission planning and other considerations to the data acquisition classes. Pilot trainees were afforded the opportunity of a non-traditional, low-stakes preparation, training, and flight time at WIES on Catalina Island, while graduate students in the spatial acquisition course experienced the mission planning and data collection, and were able to utilize the aerial imagery collected and modeled in their project work. This endeavor, while beneficial to USC-ICT and SSI, was taken over by the general instructors in SSI, and is discussed below



**Figure 7.** Photogrammetric point cloud segmentation, University of Southern California site.



**Figure 8.** 4-layer transition image of the USC campus.

## INTEGRATING REMOTE SENSING AND UAS INTO CURRICULUM

As an academic unit, the SSI continues to grow and develop curriculum at the graduate and undergraduate levels. This is evident in our changing course offerings such as Spatial Data Collection Using Drones, an introductory undergraduate course that provides students with technical and practical experience with UAS, and program development such our Graduate Certificate in Remote Sensing for Earth Observation (RSEO). These courses and programs have undergone curricular review by our respective internal curriculum committees, and the appropriate Curriculum Offices at the College and University levels.

### Student Programs and Curriculum Development

The Spatial Sciences Institute initiated curricular updates concurrently for both undergraduate education and graduate education that incorporate remote sensing at a higher level than was previously encompassed. Some changes were considered minor, modifications to content of existing courses that meet learning objectives with regards to remote sensing and UAS data collection, while other changes involved the creation of new courses and programs. As discussed for this paper, students enrolled in the graduate level Spatial Data Acquisition course who are also new to integrating remotely sensed data, collected via instructor conducted UAS flights, gain experience in pre-flight mission planning and post-flight processing of the data. They also trouble shoot problems that may arise with integration of these data

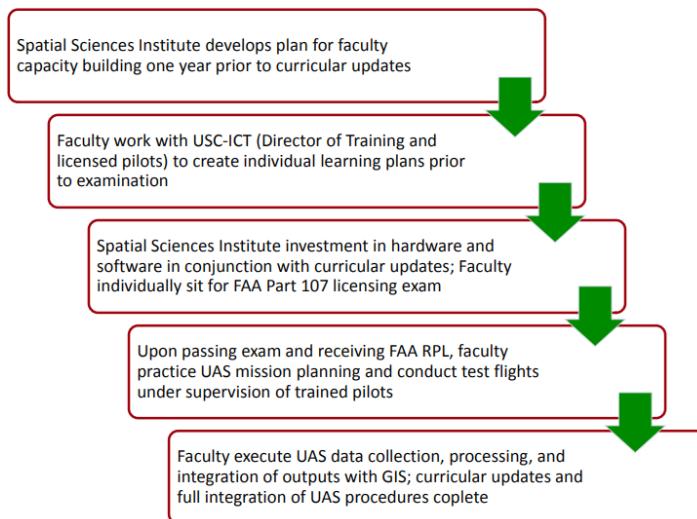
with other field collected data; we encourage students to work through the processes to solve issues such as mis-matched coordinate systems upon projecting both UAS imagery (collected in UTM Zone 11N) with data from high accuracy receivers (collected in WGS 1984 and projected in Web Mercator) under guidance from faculty. The students can then relate this practical experience with real world situations in which they will be utilizing these data and processes.

Additional curricular development culminated in the creation of a non-major undergraduate course in spatial data collection utilizing drone, in which students will, for the first time, work the UAS to plan, collect, and process imagery. This course revolves around applied and active learning experiences in which students can develop and demonstrate a deeper knowledge and understanding of the technological sciences behind the UAS-based collections, processing, and visualizations, and through germane examples. Additionally, this is a one-off course, meaning students across all disciplines are welcome to enroll in the course, no prior experience with GIS is required, and this is not limited to majors or minors. We anticipate that this course likely will draw students from the most diverse majors and academic disciplines. Lastly, we have developed a new graduate certificate program in Remotes Sensing for Earth Observations (RSEO), which leverages remotely sensed data from a multitude of sources such as Location Based Services (LBS), social media, and Internet-of-Things (IoT) devices for a variety of applications from weather and environmental observations to disaster management and recovery efforts. The program focuses on the acquisition, management, and integration of these data for the purpose of advanced trend analysis, with the aim that students and professionals develop proficiency working with these data and are able apply their use in decision-making processes.

## **Capacity Building of Faculty for Improved Student Outcomes**

In order to create a sustainable program that incorporated UAS and remote sensing technology at a greater level, SSI invested in building capacity of current faculty (Figure 9) beyond the abilities of Knowles and Marx. This entailed developing technical training and pilot certifications for the selected faculty that are, or would be, responsible for spatial data acquisition via UAS. Trainee faculty members worked in collaboration with USC-ICT and experienced pilot faculty to develop study plans and to review materials for the FAA RPL Part 107 exam. Faculty then created individual learning plans and study schedules. Due to fiscal considerations, faculty were required to sit for the Part 107 exam prior to the closing of fiscal year 2017.

In consultation with USC-ICT, Knowles, and Marx, SSI also invested in the necessary equipment and photogrammetry software for educational purposes. Equipment this included a quad copter UAS (DJI Phantom 4 Pro) with a RGB payload, no thermal or multispectral payloads are mounted currently. This was a budgetary consideration and we recommend that initial investment also include a multispectral payload. This will greatly increase the data collection possibilities to include vegetation and landscape analysis, such as Normalized Difference Vegetation Index (NDVI) or others, which can be applied to a variety of domains ranging from crop management in agricultural to tracking people and objects for human security, search and rescue, and other military operations. Additional equipment and accessories included extra Intelligent Flight batteries, back-up propellers, carrying case, and one tablet. Educational licensing for photogrammetry software (Pix4D) was purchased and is renewed annually. Additional mission planning software (AirMap, DJI-Go, etc) are available free of charge and downloadable to any mobile device.



**Figure 9.** Workflow for capacity building and training of faculty for attaining FAA RPL, Part 107.

Upon successful completion of the exam, all faculty pilots engaged in physical training with Marx and Knowles. A pre-flight check list, in-flight protocol, and post-flight image processing workflow were standardized for use with the current graduate spatial data acquisition course and are available to use for additional courses, such as the aforementioned undergraduate course

that specializes in spatial data collection using drones. Test flights, under the direction of Marx and Knowles, were conducted in open, unpopulated parks, in accordance with FAA regulations and recommendations that limit flights overhead of people in public spaces. Faculty also conducted test data collections, processing, and integration of outputs (DSM, 3D mesh) incorporated into a GIS platform to visualize georeferenced data to hone integration abilities.

## DISCUSSION

Ideally, students will develop projects that test not only the utility of remotely sensed imagery collected via UAS, but also efficiency and efficacy of their workflows. The weeklong Catalina experience can be used as a testing ground for implementation of this technology to new domains on a small scale, prior to large-scale implementation. Our goal is to provide students the opportunities to engage in active experimentation and improve learning experience and outcomes. This is accomplished through active collection of data, detailed data analysis, and product production via some geo-visualization outcome.

Having experienced faculty that can train additional faculty, and act as a resource throughout development of the programs and/or new student projects and novel flight paths, is key to the success of our programs. Successful pilot studies that apply remote sensing to wildlife tracking and monitoring, such as the first study highlighted above, and 3D models of the natural and built environment that are derived from UAS collected imagery and an automated computing process, such as accomplished by USC-ICT, are exemplars of the advantages of integrating remote sensing in a GIS. Additionally, faculty that can effectively communicate and demonstrate the possibilities of remotely sensed data acquired via UAS, are vital to improved student experiences and outcomes. SSI does not aim to train pilots, and in fact some of the students may already have experience working with UAS and product visualization through current jobs. Rather, courses and programs focus on the utility of remotely sensed data within a GIS, the science of photogrammetry and production of geo-referenced 3D models, and the variety of geospatial analyses that can be run for the diverse applications of remotely sensed data.

Lastly, while we have focused the case studies presented here on remotely sensed data mainly collected via unmanned aerial systems, our students interact with a variety of remotely sensed data (LiDAR, multi- and hyperspectral satellite imagery, etc) within a GIS during these courses and in the

progress of research projects that span the humanities and physical sciences. Through this work, SSI reinforces that these educational opportunities, focused primarily on data acquisition, are instrumental in supporting the application of geographic information systems, science, and technology in many diverse fields ranging from human security and humanitarian relief, to sustainable urban and rural planning and public health.

## **CONCLUSION**

We have presented two innovative research projects that integrate remotely sensed data collected via UAS in GIS for distinct purposes, but that share the common element of further integration of remote sensing into the curriculum of spatial sciences courses. We successfully demonstrated the potential for wildlife monitoring on Catalina Island utilizing UAS and made tangible recommendations the future work in this domain. We also presented the work of USC-ICT and development of an automated process to build a fully geo-referenced 3D model of the earth for the training and simulation needs as the impetus and model for faculty development at SSI. In order to achieve the curricular renovations referenced above, faculty must be properly equipped to guide student data acquisition, processing, and analysis; we presented the workflow of how the Spatial Sciences Institute achieved this and the importance for our student development.

## **ACKNOWLEDGEMENTS**

We would like to thank the Catalina Island Conservancy for engaging with the pilot study described here and permitting the public dissemination of the results. We also thank the staff of the Wrigley Institute for Environmental Studies on Catalina Island for their continue support of the field course.

## **CONFLICTS OF INTEREST**

The authors declare no conflicts of interest regarding the publication of this paper.

## REFERENCES

1. Kolb, D.A. (2015) Experiential Learning: Experience as the Source of Learning and Development. 2nd Edition, Pearson Education, Inc.
2. McLeod, S.A. (2017) Kolb-Learning Styles and Experiential Learning Cycle. Simple Psychology. <https://www.simplypsychology.org/learning-kolb.html>
3. Loyola, L.C., Marx, A.J. and Fleming, S.D. (2018) Combining Teaching, Partnerships, and Research in the Field: Lessons from the Spatial Data Acquisition Course (on Catalina Island). GIS-Pro & CalGIS 2018, Palm Springs, 8-12 October 2018.
4. Kays, R., Sheppard, J., Mclean, K., Welch, C., Paunescu, C., Wang, V., Crofoot, M., et al. (2019) Hot Monkey, Cold Reality: Surveying Rainforest Canopy Mammals Using Drone-Mounted Thermal Infrared Sensors. International Journal of Remote Sensing, 40, 407-419. <https://doi.org/10.1080/01431161.2018.1523580>
5. Hodgson, J.C., Mott, R., Baylis, S.M., Pham, T.T., Wotherspoon, S., Kilpatrick, A.D., Koh, L.P., et al. (2018) Drones Count Wildlife More Accurately and Precisely than Humans. Methods in Ecology and Evolution, 9, 1160-1167. <https://doi.org/10.1111/2041-210X.12974>
6. Chrétien, L.P., Théau, J. and Ménard, P. (2016) Visible and Thermal Infrared Remote Sensing for the Detection of White-Tailed Deer Using an Unmanned Aerial System. Wildlife Society Bulletin, 40, 181-191. <https://doi.org/10.1002/wsb.629>
7. Lhoest, S., Linchant, J., Quevauvillers, S., Vermeulen, C. and Lejeune, P. (2015) How Many Hippos (HOMHIP): Algorithm for Automatic Counts of Animals with Infra-Red Thermal Imagery from UAV. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 40, 355-362. <https://doi.org/10.5194/isprsarchives-XL-3-W3-355-2015>

# **CHAPTER**

# **18**

## **Development of Drone Cargo Bay with Real-Time Temperature Control**

---

**Sedam Lee, Yongjin Kwon**

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon, South Korea

### **ABSTRACT**

In order to deliver medical products (medicines, vaccines, blood packs, etc.) in time for needed areas, a method of transporting goods using drones is being studied. However, temperature-sensitive medical products may decay due to outside temperature changes. The time required to transport over the distance may vary a lot as well. As a result, the likelihood of the goods deteriorating is very high. There is a need for a study on cargo bay to prevent

---

**Citation:** Lee, S. and Kwon, Y. (2019), "Development of Drone Cargo Bay with Real-Time Temperature Control". *World Journal of Engineering and Technology*, 7, 612-621. doi: 10.4236/wjet.2019.74044.

**Copyright:** © 2019 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY). <http://creativecommons.org/licenses/by/4.0>

this and to protect the medical goods. In this paper, in order to protect the temperature sensitive medical goods, the inside cargo bay is equipped with the cooling fan device and the electric heating elements. These elements can be monitored and controlled according to the user's discretion. By using the web server built inside the cloud server, the temperature can be controlled in real-time from anywhere without the limitation of distance. We built the proposed device, and installed it on the drone cargo bay. The test results show that the cargo bay can be temperature-controlled, and the setting can be maintained over a great distance. The user can watch the temperature variations during the transport and ascertain the goodness of the medical supply with the data. It is expected that such development can greatly enhance the utility of the drone operations, especially for the medical supply transport applications.

**Keywords:** Real-Time Control, Cargo Bay Temperature, Cloud Server, Medical Transport, Drones

## INTRODUCTION

In many countries around the world, a fair accessibility to roads varies according to the geographic characteristics and weather conditions. As a result, it is difficult to transport items, such as medical supplies (medicines, vaccines, blood packs, etc.) when needed [1]. Currently, there is a variety of methods for transporting medical products by walking, transporting vehicles, helicopters, and airplanes. However, these methods have limitations in terms of distance, road accessibility, delivery costs, and geographic conditions [1]. To overcome these limitations, methods for transporting medical products using drones are being studied. The use of drones among various methods has already been extended to many industries [2] [3]. In addition, the transport of goods using drones has been steadily studied in recent years [4]. According to the papers that studied the effects of transporting blood by drones, the use of unmanned aircraft does not affect the quality of the products, such as coagulation and deterioration of medicines and biochemical samples [5]. However, the factor mostly affecting the goods in transport is the external environment. Depending on various factors such as region, season and weather, medical and various items stored in drone cargo bay may deteriorate. Therefore, there is a need for a method for protecting the medical supplies from outside temperature changes.

Currently, a common method for lowering the temperature inside the drone cargo bay is by arranging dry ice, and ice packs. Those include the

Rwanda's blood transport drone operations and a few other countries. This method does not have a means to raise or control the internal temperature to the desired setting. The existing method is simple, but due to various external requirements, it is difficult to avoid deterioration of stored items. This is because the type, place, time, and storage temperature of the stored items are all different. For this reason, there is a need for a method capable of controlling the cargo bay temperature. To solve this problem, we developed a technology to control the temperature inside the drone cargo bay in real-time. The structure of this paper shows Chapter 2 as related works, the configuration of temperature control system in Chapter 3, and hardware development in Chapter 4. Chapter 5 describes software development, and Chapter 6 describes the prototyping results. Finally, the last chapter includes concluding remarks.

## RELATED WORKS

In recent years, various approaches have been proposed for the delivery of medical items (blood, medical supplies, defibrillators, life-rings, etc.) using the drones [5]. One of these methods is to develop an efficient delivery system to overcome the hurdles from mountains and buildings during the flight, or to operate and manage drones using a modular design approach to effectively accommodate the various cargo types [6] [7]. However, the methods proposed or introduced in the papers have not considered the goodness of the transport items during the flight. In addition, the paper for improving the goodness of transport items is mainly dealt with cargo bay insulation to maintain the internal temperature. In most cases, the method focuses on maintaining a low temperature. It shows no result about transporting items that need to be stored at high temperature. It is difficult to find the related study that shows the flexible temperature control of the drone cargo bay [8] [9]. In this regard, the method to control the temperature of cargo bay by raising or lowering the temperature remotely can be viewed as quite innovative [10] [11].

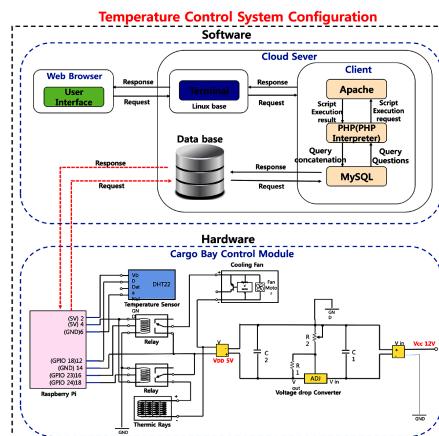
## CONFIGURATION OF TEMPERATURE CONTROL SYSTEM

In order to operate cargo bay module with real-time temperature control, the system is configured as in Figure 1. System configuration is divided into hardware and software sections. Hardware is composed of the cargo bay

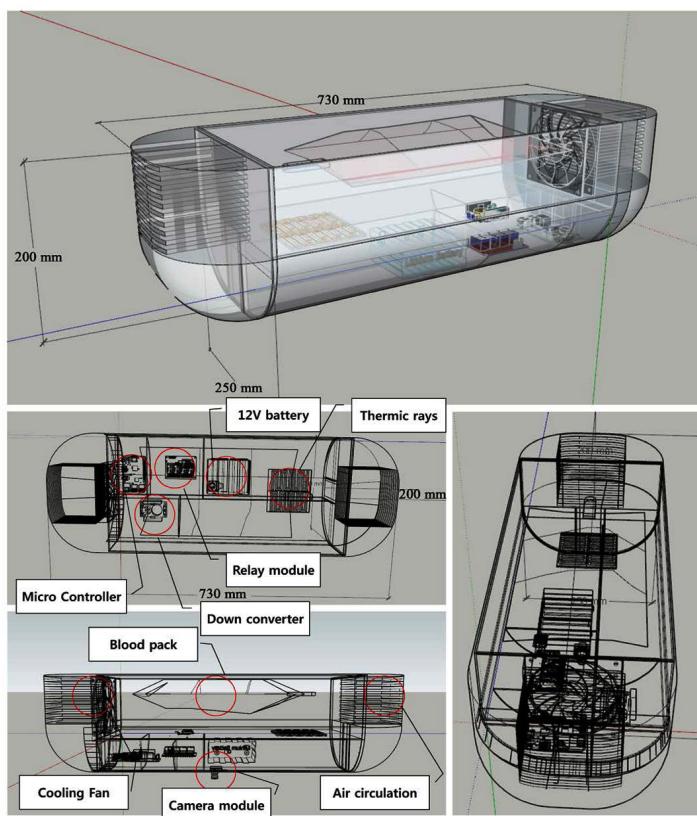
control module that exchanges data with the database, mainly on Raspberry Pi. This controller controls the attached devices and sensors. Software is largely divided into cloud server and web browser. Client and database that can send and receive data to and from the cloud server are configured. The data can be transmitted to the UI (user interface) of web browser through terminal. The real-time temperature control method proposed in this study is the method of building various servers for network connectivity through cloud sever, and controlling the cargo bay module using the deployed server, which can check and command the data of the cargo bay module using the UI.

## HARDWARE DESIGN

The temperature-controlled drone cargo bay proposed in this study is 730-mm wide, 250-mm long and 200-mm high, as shown in Figure 2. This is designed by considering the payload restrictions of the drone. In order to effectively lower the internal temperature during the flight, the air circulation port was fabricated. The cooling fan plays the role of taking the air to the cargo bay, which can lower the internal temperature. Thermic ray, which generates heat through the electricity supply, was used, and a barrier membrane was designed to maintain the cargo bay temperature efficiently. The barrier membrane also divided the cargo bay interior space into an interior compartment for storing goods and an exterior compartment for installing temperature control devices. Table 1 and Table 2 show the details.



**Figure 1.** Temperature control module system configuration.



**Figure 2.** Temperature controlled cargo bay schematic.

**Table 1.** Controller specification.

| Detail              | Raspberry Pi 3 b+   |
|---------------------|---|
| <b>Processor</b>    | <ul style="list-style-type: none"> <li>- Broadcom BCM2837B0, Cortex-A53 64-bit</li> </ul>   |
| <b>Memory</b>       | <ul style="list-style-type: none"> <li>- 1 GB LPDDR2 SDRAM</li> </ul>   |
| <b>Connectivity</b> | <ul style="list-style-type: none"> <li>- Dual band 2.4 GHz &amp; 5 GHz 802. 11b/g/n/ac</li> <li>- Gigabyte Ethernet over USB 2.0</li> </ul> |
| <b>Access</b>       | <ul style="list-style-type: none"> <li>- Extended 40-pin GPIO header</li> </ul>   |
| <b>Input Power</b>  | <ul style="list-style-type: none"> <li>- 5 V/2.5 A DC via micro USB connector</li> <li>- 5 V DC via GPIO header</li> </ul>                  |
| <b>Environment</b>  | <ul style="list-style-type: none"> <li>- Operating temperature, 0°C - 50°C</li> </ul>   |

**Table 2.** Temperature sensor, thermic rays, cooling fan specifications.

| Detail                 | DHT22 (sensor)    | Themic rasys       | Cooling fan       |
|------------------------|-------------------|--------------------|-------------------|
| <b>Input Power</b>     | DC 3.3 V - 5 V    | DC 5 V/570 mA      | DC 12 V/0.5 A     |
| <b>Measuring range</b> | -40°C - 80°C      | ~200°C             | -                 |
| <b>accuracy</b>        | ±0.5°C            | -                  | -                 |
| <b>Size</b>            | 43 * 26 * 18 (mm) | 90 * 60 * 0.3 (mm) | 97 * 97 * 33 (mm) |
| <b>Weight</b>          | 9.3 g             | 15 g (*2)          | 155 g             |

## SOFTWARE DESIGN

### Network Connection

It is implemented that a communication using RF sensor as a way to check and control the temperature inside cargo bay in real time [12]. However, RF communications have adopted a network connection method to address distance constraints. A network connection using a specific IP address by building a web server in the controller's memory is not possible due to the nature of the drone when the network is disconnected due to geographic distances. Therefore, we designed a static IP that plays an intermediate bridge role by building a web server through a cloud server. The web server was installed using “Apache, PHP, MySQL” on the cloud server, and data were running with the use of interlocks to receive data from the controller and send commands. Table 3 shows the data interlocking list for network connections. All six data linkage lists are implemented in PHP language through Terminal. It is designed to display and command data on the UI through interlocked data.

The control UI is designed to prevent users from receiving unnecessary information through the controller's kernel window or complicating the controller's command statements. It is designed to receive the necessary information quickly or issue commands. The first box of each configuration in Figure 3 shows the current temperature, as well as the upper and lower limits of the temperature. The second is the upper and lower limit input fields of temperature. The third is the operating status of thermic ray and cooling fan, and the on/off control switch of the controller power. The fourth box is a graph showing the cargo bay's internal temperature change, updated continuously over time to show the changes in temperature.

## Control Algorithm Design

The goods stored in cargo bay for the transport have different storage temperatures. Since the different storage temperatures must be considered for each item, the cargo bay internal temperature must be freely set. When a certain value is set for internal temperature control and there is no margin value of the specified temperature, the temperature control device adjusts frequently even with a minor temperature change of only 0.1°C. Therefore, the temperature should be set by specifying the range. In this study, the method of setting and controlling the temperature using upper and lower limits was used.

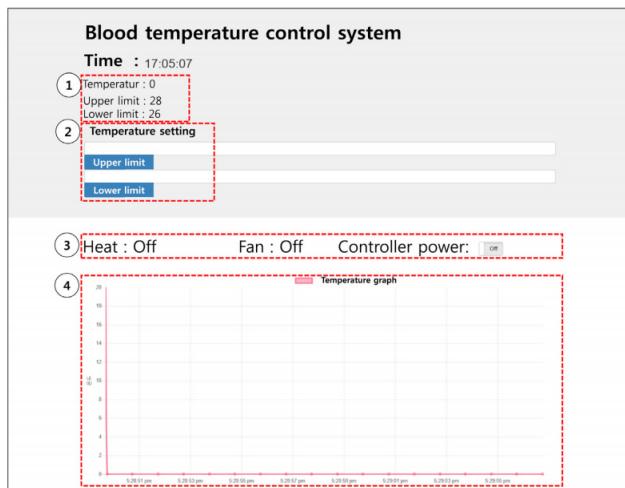
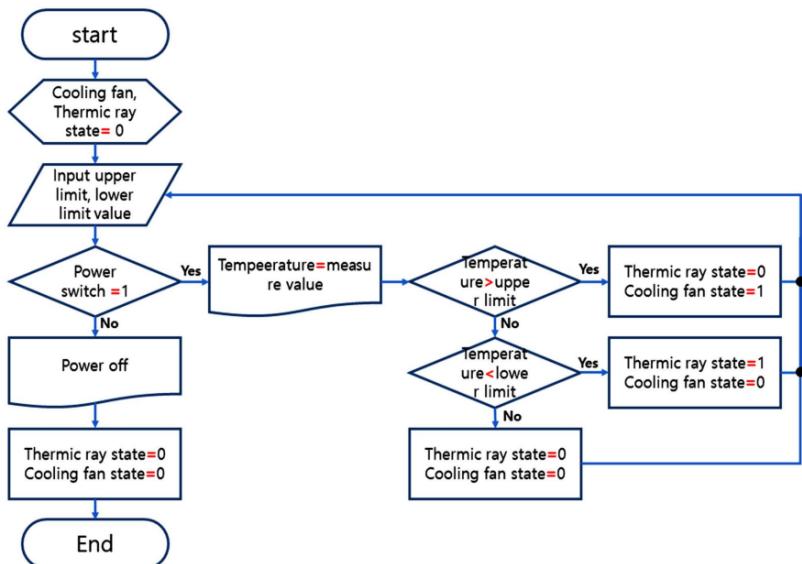


Figure 3. Temperature control user interface.

Table 3. Network data interlock list.

| Division                 | Contents   |
|--------------------------|--|
| Temperature sensor value | <ul style="list-style-type: none"> <li>Data value measured by temperature sensor is saved in database</li> <li>Keep 50 latest data and delete old data</li> </ul>                          |
| UI data value            | <ul style="list-style-type: none"> <li>Database and UI interworking, output values stored in database</li> <li>Retrieve sensor value from database every second</li> </ul>                 |
| Cooling fan              | <ul style="list-style-type: none"> <li>According to the upper and lower limits of the set temperature, whether or not operation is carried out to or from the database</li> </ul>          |
| Thermic rays             | <ul style="list-style-type: none"> <li>According to the upper and lower limits of the set temperature, whether or not operation is carried out to or from the database</li> </ul>          |
| Micro Controller         | <ul style="list-style-type: none"> <li>Pass the data value collected from the Micro Controller</li> <li>Passing device values controlled through the controller to the database</li> </ul> |
| User input value         | <ul style="list-style-type: none"> <li>Input the input value that user wants to control to UI and pass to database</li> </ul>  |

The temperature control algorithm set the basic operation of the thermal ray and cooling fan to zero as shown in Figure 4. After that, the upper and lower limit values are entered and the algorithm is repeated or terminated according to each condition. For example, in order to store an article having a storage temperature of 25°C to 26°C, the upper limit value is set to 26°C and the lower limit is set to 25°C. After inputting the upper and lower limit and the internal temperature is higher than the upper limit, the cooling fan becomes operated. If the internal temperature is lower than the lower limit, the Thermic ray becomes activated. If the internal temperature is between the upper and lower limits, the current temperature is maintained.



**Figure 4.** Temperature control algorithm pseudo code.

## PROTOTYPE DESIGN

Figure 5 shows the VTOL (vertical take-off and landing) drone equipped with a prototype temperature controlled cargo bay. Since the payload is about 2 kg, the weight of the medical product as a transport item is considered. The cargo bay prototype was built with a weight of around 1.5 kg. To check whether the temperature inside the cargo bay is controlled and maintained, two temperature sensors were used to measure the current temperature. The temperature inside the cargo bay was checked for 15 minutes in every 10 seconds. As shown in Figure 5, the upper and lower limits of the cargo bay

temperature were adjusted to 25°C and 24°C, respectively, and the cargo bay temperature was measured to be maintained at the desired setting.

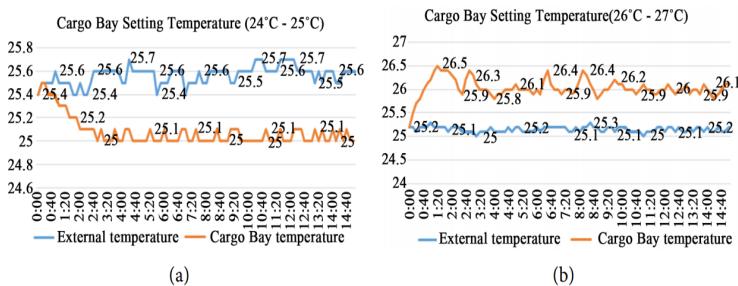
It took about three minutes for the cargo bay internal temperature to reach the set point at 25.4°C. After that, when the internal temperature is out of the set temperature, the cooling fan operates to maintain 25°C. In order to check whether the temperature is maintained by thermic ray, it is verified by setting the cargo bay internal temperature to 26°C - 27°C, as shown in Figure 6. Compared to the cooling fan, the set temperature was reached in about 40 seconds. If the temperature goes out of the set temperature, the thermic ray is operated to increase the temperature to maintain the internal temperature, and there is an error of about 0.1°C.

The controller can effectively recognizes and controls the setting measured by the temperature sensors. However, an error occurs between the set temperature and the holding temperature.

The error in temperature is due to the delay in the data and must react immediately when the temperature changes. In fact, the precise measurement, control, and data processing require the use of high-performance computers, sensors, and devices. In this study, however, due to the limitations of weight and payload of the drone, the use of high performance precision device was not feasible (Table 4).



**Figure 5.** Temperature module prototype.



**Figure 6.** External temperature vs. cargo bay internal temperature. (a) Graph of internal temperature and external temperature; (b) Graph of internal temperature and external temperature.

**Table 4.** Erro value with a 40 second interval.

| °Ct         | 40 (s)  | 80 (s)                        | 120 (s) | 160 (s) | 200 (s) | 240 (s) | 280 (s) |
|-------------|---------|-------------------------------|---------|---------|---------|---------|---------|
| 24°C - 25°C |         | Set temperature reach section |         | 0.1°C   | 0.1°C   | 0.1°C   | 0.1°C   |
| 26°C - 27°C |         | Set temperature reach section |         | 0.1°C   | -       | 0.2°C   | -       |
|             | 320 (s) | 360 (s)                       | 400 (s) | 440 (s) | 480 (s) | 520 (s) | 560 (s) |
| 24°C - 25°C | 0.1°C   | 0.1°C                         | 0.1°C   | 0.1°C   | -       | 0.1°C   | 0.1°C   |
| 26°C - 27°C | -       | 0.1°C                         | -       | 0.1°C   | -       | 0.2°C   | -       |
|             | 600 (s) | 640 (s)                       | 680 (s) | 720 (s) | 760 (s) | 800 (s) | 840 (s) |
| 24°C - 25°C | -       | 0.1°C                         | 0.1°C   | -       | 0.1°C   | 0.1°C   | 0.1°C   |
| 26°C - 27°C | -       | 0.1°C                         | -       | -       | 0.1°C   | 0.1°C   | 0.2°C   |

The temperature measured for a total of 15 minutes, and the interval was set as 40 seconds. Each error value is an absolute value that is the maximum error. In order to reduce error from the data delay, there are ways to increase the electronic device load capacity by using the high performance computers, sensors and devices, or to increase the network performance of the controller. Also, there is a way to use the LTE modem to increase the controller and network performance. If a modem to make the controller an access point (AP) is used to reduce the data delay, one can effectively reduce the error [13].

## CONCLUSION

This paper deals with the development of temperature controlled cargo bay to protect goods from decay due to external environmental factors. A small

controller, Raspberry Pi, was used to control the temperature control device, and a cloud server was used to solve the communication distance limitations. As a result of detecting the change of the cargo bay internal temperature for 15 minutes, it was confirmed that the set temperature was effectively maintained and monitored, despite some small errors. The developed device can be used to protect and safely transport items that are easily corrupted and deteriorated, such as medical blood products that are very sensitive to outside temperature variations. Even though the proposed device showed a small range of errors during the operation, this problem can be easily solved by using more accurate devices. The method provides a new approach to either raising or lowering the inside temperature of drone cargo bay, and such can be done remotely in real-time. By doing so, the temperature variations during the flight can be verified, and the end user can be assured that the transport items have arrived safely.

## **ACKNOWLEDGEMENTS**

This work was supported by the Hyundai-NGV Future Technology Research Fund.

## **CONFLICTS OF INTEREST**

The authors declare no conflicts of interest regarding the publication of this paper.

## REFERENCES

1. Laksham, K.B. (2019) Unmanned Aerial Vehicle (Drones) in Public Health: A SWOT Analysis. *Journal of Family Medicine and Primary Care*, 8, 342-349. [https://doi.org/10.4103/jfmpc.jfmpc\\_413\\_18](https://doi.org/10.4103/jfmpc.jfmpc_413_18)
2. Griffiths, F. and Ooi, M. (2018) The Fourth Industrial Revolution-Industry 4.0 and IoT. *IEEE Instrumentation & Measurement Magazine*, 21, 29-43. <https://doi.org/10.1109/MIM.2018.8573590>
3. Li, B., Fei, Z. and Zhang, Y. (2019) UAV Communications for 5G and Beyond: Recent Advances and Future Trends. *IEEE Internet of Things Journal*, 6, 2241-2263. <https://doi.org/10.1109/JIOT.2018.2887086>
4. Klinkmueller, K.M., Wieck, A.J., Holt, J.K. and Valentine, A.W. (2019) Airborne Delivery of Unmanned Aerial Vehicles via Joint Precision Airdrop Systems. *AIAA Scitech 2019 Forum*, San Diego, CA, 7-11 January 201, 2285. <https://doi.org/10.2514/6.2019-2285>
5. Amukele, T., Ness, P.M., Tobian, A.R., Boyd, J. and Street, J. (2016) Drone Transportation of Blood Products. *Transfusion*, 57, 582-588. <https://doi.org/10.1111/trf.13900>
6. Scott, J.E. and Scott, C.H. (2017) Drone Delivery Models for Healthcare. *Proceedings of the 50th Hawaii International Conference on System Sciences*, Hilton Waikoloa Village, HI, 4-7 January 2017, 3297-3304. <https://doi.org/10.24251/HICSS.2017.399>
7. Lee, J.H. (2017) Optimization of a Modular Drone Delivery System. *2017 IEEE International Systems Conference*, Montreal, 24-27 April 2017, 1-8.
8. Erdos, D., Erdos, A. and Watkins, S.E. (2013) An Experimental UAV System for Search and Rescue Challenge. *IEEE Aerospace and Electronic Systems Magazine*, 28, 32-37. <https://doi.org/10.1109/MAES.2013.6516147>
9. Xiang, G., Hardy, A., Rajeh, M. and Venuthurupalli, L. (2016) Design of the Life-Ring Drone Delivery System for Rip Current Rescue. *2016 IEEE Systems and Information Engineering Design Symposium*, Charlottesville, VA, 29April 2016, 181-186. <https://doi.org/10.1109/SIEDS.2016.7489295>
10. <https://www.hankookilbo.com/News/Read/201903201101062687>
11. <https://fortune.com/2019/01/07/delivery-drones-rwanda/>
12. Lee, S.-H., Yang, S.-H. and You, Y.-M. (2017) Design and Development of Agriculture Drone Battery Usage Monitoring System Using

- Wireless Sensor Network. *The International Journal of Advanced Smart Convergence*, 6, 38-44.
13. Krishna Chand, G.S.L., Lee, M. and Shin, S.Y. (2018) Drone Based Wireless Mesh Network for Disaster/Military Environment. *Journal of Computer and Communications*, 6, 44-52. <https://doi.org/10.4236/jcc.2018.64004>



# INDEX

---

## Symbols

3D cartesian paths 81, 86

## A

acceleration 80, 81, 83, 86, 90, 91, 93, 95, 96, 101, 108

access point (AP) 394

actuators 65

Adaptive control algorithms 25

Aerial Vehicle Control Language (AVCL) 80

aerodynamic forces 122

aircraft 322

algorithm 392

ammonia volatilization 345

angle of attack (AOA) 271

aquadcopter 61

attitude control 115, 131, 134, 136

Attitude Controller block 67

automatic cardioverter defibrillator (AED) 319

automatic/manual switching 131

Autonomous planning 41

autonomous takeoff 4

autonomous unmanned aerial systems (AUAS) 38

autonomy 39

## B

Backstepping 20, 25, 32, 33, 34

Backstepping control 25

battery 318, 319, 323

blade element theory (BET) 270

Bluetooth 4, 6, 7, 8, 9, 12, 13, 15, 16, 17

broadband wireless access 5

Brushless motors 65

## C

- Calibrated Reflectance Panel (CRP) 330  
 351
- Catalina Island Conservancy (CIC) 370, 371
- cell decomposition 143
- chlorophyll 345, 346
- Civil Certificate of Waiver Authorization (COA) 373
- clockwise (CW) 215
- cluster heads (CHs) 202
- Commercial off the Shelf (COTS) 377
- communication 390, 395
- Compact Airborne Spectrographic Imager (CASI) 346
- computer systems 114
- coordinate system 21, 22
- Coriolis terms 262, 265
- D**
- Data acquisition 127
- deconfliction 38, 39, 43, 49, 51, 52
- Disturbance 69, 71
- Downwelling Light Sensor (DLS) 350
- Dubins waypoint navigation (DWN) 209, 211, 224
- E**
- Earth remote sensing (ERS) 327
- Electronic Speed Controllers (ESC) 65
- elementary motion detector (EMD) 296
- environments 142
- Euler angles 62, 64
- Euler-Lagrange method 263
- Evolutionary algorithms 114
- Experimental Railway Ring (ERR) 330
- F**
- Federal Aviation Administration (FAA) 373
- feedback control algorithm 220
- feedback linearization (FBL) 26
- Finite-Difference Time-Domain (FDTD) Method 147
- finite state machine (FSM) 235
- fire detection 3
- Fixtures 319
- flight mission 4
- forest surveillance 3
- formation leader (FL) 202
- Fuselage 122
- fuselage body 9
- fuzzy logic 28, 29
- G**
- Gaussian process 244
- genetic algorithm 28
- Geographic Information Sciences & Technology (GIS) 371
- geographic information system (GIS) 232
- Global Positioning System receivers (GPS) 113
- graphical user interface (GUI) 303
- gravity 122, 123
- Green Normalized Difference Vegetation Index (GNDVI) 347
- guaranteed time slot (GTS) 6
- H**
- helicopter dynamics 114, 117
- I**
- inertial measurement unit (IMU) 9,

297, 305

Institute for Creative Technologies  
(ICT) 370, 371

interface Babelfish 305

Internet Communications Engine  
(ICE) 305

iterative feedback tuning (IFT) 284

iterative learning control (ILC) 283

## L

landing maneuvers 4

Linear Quadratic Regulator (LQR)  
20

logical architecture 10, 12

## M

machine learning 28

maneuverability 20

maneuvers 115, 127, 131, 134, 136,  
137, 139

marine pollution 230

Maritime activity 230

mass moments of inertia 22

master node (MM) 9

Mathematical models 117

Micasense Atlas software 350

microaerial vehicles (MAVs) 296

Mission Planner (MP) 84, 105

mobile cellular networks 5

mobile robot formation control 180

model free adaptive control (MFAC)  
283

momentum theory (MT) 270

Motion detection 296

## N

neural networks 28, 29

Newton-Euler formalism 21, 167

nitrate leaching 345

Nitrogen 343, 344, 345, 363, 364,  
365, 366, 367

nonlinear aerodynamic effects 181

Normalized Difference Vegetation  
Index (NDVI) 381

## O

Obstacle Field Navigation (OFN)  
82

One World Terrain (OWT) 370

Optimization algorithms 27

orientation vector 186, 187

Ostafievo airport 336

## P

packet loss ratio (PLR) 14

partial least squares (PLS) 283

Path Command block 67

PHP language 390

Position Controller block 67

principal component analysis (PCA)  
283

probability density 244

propeller 317, 318, 321, 322, 323

propellers 61, 65

proportional Integral Derivative  
(PID) 60

pseudorandom binary sequence  
(PRBS) 277

pulse width modulation (PWM) 20

## Q

Quadcopter Control mixing block  
67

Quadcopter Dynamics block 67

Quadcopters 60

quadrotor 255, 256, 257, 258, 259,  
260, 261, 262, 263, 264, 265,  
266, 267, 268, 269, 270, 271,

272, 273, 274, 275, 276, 277,  
 278, 279, 280, 281, 282, 283,  
 284, 285, 286, 287, 288, 289,  
 290, 291, 292, 293, 294  
 quadrotor aerial robot 163, 164, 165  
 quadrotor mass 166, 173  
 quadrotor system 297  
 quadrotor unmanned aerial vehicle  
 19, 20

**R**

radar antenna 147, 148  
 Radar Cross Section (RCS) 144  
 radar model 146  
 radars 142, 143, 148, 149, 152, 155  
 radar satellite information 331  
 railway transportation 328  
 Reichardt model 297, 300  
 relay nodes (RN) 202  
 Remote License Pilots (RPL) 372  
 robots 229, 230, 231, 234, 236, 237,  
 240, 244, 245, 246  
 Robust control algorithms 26  
 Root Mean Squared Error (RMSE)  
 351  
 rotor 20, 21, 22

**S**

satellite networks 5  
 satellite photography 332  
 scatternet 12  
 Sliding Mode Control (SMC) 24  
 snowfalls 328  
 soil denitrification 345  
 Spatial Sciences Institute (SSI) 369,  
 370  
 state vector 63  
 swarm robotics 229, 230, 253, 254

**T**

Tag Image File Format (TIFF) 350  
 Thermic ray 388, 392  
 Torque 118, 122  
 translational rotation matrix 182

**U**

Unmanned Aerial Systems (UAS)  
 79  
 Unmanned Aerial Vehicles (UAVs)  
 344

**V**

vegetation indices (VIs) 344, 345  
 vertical take-off and landing (VTOL)  
 20, 164, 256  
 virtual machine (VM) 9  
 virtual reality/augmented reality  
 (VR/AR) 370  
 virtual reference feedback tuning  
 (VRFT) 284

**W**

watershed pollution 3  
 waypoint navigation (WN) 209, 210  
 WiFi network 349  
 wireless local networks (WLAN) 5  
 wireless personal networks (WPAN)  
 5

Wrigley Institute for Environmental  
 Studies (WIES) 372

**Z**

zone surveillance 3

# Unmanned Aerial Vehicles (UAV) and Drones

Drones are defined as unmanned aerial vehicles (UAVs) that can fly over space from a few tens of meters - up to thousands of kilometers, or small (micro and nano) quadrotors that can move indoors. Today – there is a rising need of these types of flying vehicles - for civil and for military applications. There is also significant interest for development of new types of drones that can fly autonomously – around different locations and environments, and will accomplish different technical missions (agriculture, public events – sports/ concerts, space research etc.).

Within the past decade – wide spectra of applications have emerged that resulted in many variations in the size and the design of the UAV-s (drones). The year of 2015 was declared as a year that exponentially boosted the drone applications in many human-oriented fields and environments, especially in agriculture and forestry (75% of all applications).

The drones categories differ among themselves by their configuration, platform or application. There are 3 major classes defined in the literature, and the class I has 4 subclasses (a, b, c, d):

Class I-a) – Nano drones, weight < 200 g.

Class I-b) – Micro drones, weight between 200g – 2 kg.

Class I-c) – Mini drones, weight between 2 kg – 20 kg.

Class I-d) – Small drones, weight between 20 kg – 150 kg.

Class II – Tactical drones, weight between 150-600 kg.

Class III – Strike/ Hale drones, weight over 600 kg.

This edition covers different topics from UAV-s and drones – such as design and development of UAV-s, trajectory control techniques, small drones and quadrotors, and application scenarios of UAV-s and drones.

Section 1 focuses on design and development of UAV-s, describing design and development of a fly-by-wireless UAV platform, review of control algorithms for autonomous quadrotors, central command architecture for high-order autonomous unmanned aerial systems, and quadcopter design for payload delivery.

Section 2 focuses on trajectory control techniques, describing development of rescue material transport UAV, railway transport infrastructure monitoring by UAVs and satellites, assessment of UAV based vegetation indices for nitrogen concentration estimation in spring wheat, research and teaching applications of remote sensing integrated with GIS: examples from the field, and development of drone cargo bay with real-time temperature control.

Section 3 focuses on small drones and quadrotors, describing neural network control and wireless sensor network-based localization of quadrotor UAV formations, Dubin's waypoint, navigation of small-class unmanned aerial vehicles, modelling oil-spill detection with swarm drones, a survey of modelling and identification of quadrotor robot, and visual flight control of a quadrotor using bioinspired motion detector.

Section 4 focuses on application scenarios of UAV-s and drones, describing.



Dr. Zoran Gacovski has earned his PhD degree at Faculty of Electrical engineering, Skopje. His research interests include Intelligent systems and Software engineering, fuzzy systems, graphical models (Petri, Neural and Bayesian networks), and IT security. He has published over 50 journal and conference papers, and he has been reviewer of renowned Journals. Currently, he is a professor in Computer Engineering at European University, Skopje, Macedonia.