



UNMANNED AERIAL VEHICLES SURVEILLANCE AND INSPECTION USING OPTICAL IMAGING SENSORS

Carlos Alberto Moraes Correia

Thesis submitted to the Postgraduate Program in Instrumentation and Applied Optics of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Advisor: Diego Barreto Haddad
Co-advisor: Fábio Augusto de Alcântara Andrade

Rio de Janeiro,

September 14^{th}, 2022.

UNMANNED AERIAL VEHICLES SURVEILLANCE AND INSPECTION USING OPTICAL IMAGING SENSORS

Thesis submitted to the Postgraduate Program in Instrumentation and Applied Optics of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Carlos Alberto Moraes Correia

Examining jury:

President, Professor D.Sc. Diego Barreto Haddad (CEFET-RJ) (Advisor)

Professor Ph.D. Fábio Augusto de Alcântara Andrade (USN/Norway) (Co-advisor)

Dr. Agnar Holten Sivertsen, Ph.D. (NORCE/Norway)

Professor D.Sc. Alessandro Rosa Lopes Zachi (CEFET-RJ)

Dr. Anthony Reimer Hovenburg, Ph.D. (TracSense/Norway)

Professor D.Sc. Felipe da Rocha Henriques (CEFET-RJ)

Rio de Janeiro,

September 14^{th}, 2022.

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C824 Correia, Carlos Alberto Moraes
Unmanned aerial vehicles surveillance and inspection using optical imaging sensors / Carlos Alberto Moraes Correia – 2022.
149f : il. (algumas color.) , enc.

Tese (Doutorado). Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia : f. 142-149.
Orientador: Diego Barreto Haddad.
Co-orientador: Fabio Augusto de Alcântara Andrade.

1. Drones (Aeronaves). 2. Veículos remotamente pilotados. 3. Veículo aéreo não tripulado (VANT). 4. Inovações tecnológicas. 5. Controle preditivo .I. Haddad, Diego Barreto (Orient.). II Andrade, Fabio Augusto de Alcântara (Co-orient.). III. Título.

CDD 629.1326

Elaborada pelo bibliotecário Leandro Mota de Menezes – CRB/7 nº 5281

DEDICATION

I would like to dedicate this Doctorate Thesis to my best friend, my “comrade” and my “partner”, the person who taught me the most important things about life and who is the ultimate reason for everything I do: To my beloved son João Vitor.

Diese Doktorarbeit möchte ich meinem besten Freund, meinem “Kameraden” und meinem “Partner”, dem Menschen, der mir das Wichtigste über das Leben beigebracht hat und der der eigentliche Grund für alles ist, was ich tue, widmen: An meinen geliebten Sohn João Vitor.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Also, some people – many professors, friends and colleagues – contributed in many ways, in order that I could fulfill this task; their remarkable help was essential, without it I would certainly face much more difficulties, so I would like sincerely to show my great thanks to them.

Firstly, my friends, Professors Dr. Lourival José Passos Moreira and Dr. Jesse Costa, whom I had the privilege of teaching together in the Naval Academy and through whom I was introduced in the post-graduation program of Electrical Engineering, in CEFET-RJ;

Professor Dr. Ana Lúcia Ferreira Barros, of PPGIO, for the tireless aid and advises in many moments;

My friends Denys Pestana, Fábio “Diretor” Marujo da Silva and Fábio “Bahia” Oliveira da Silva, who accompanied me since the beginning and with their friendship, encouragement and partnership, made this doctorate course much more pleasant;

Professor Dr. Aline Gesualdi Manhães, my former advisor in PPGIO, for all the efforts, help and advises of crucial importance in many moments and different situations in my academic journey;

Professor Dr. Diego Barreto Haddad, for his attention and goodwill in accept to be my advisor, and for his support and wise advises;

Professor Dr. Alessandro Rosa Lopes Zachi, my former advisor in the Master course, in PPEEL, CEFET-RJ, and my friend, for his competent support and for his decisive encouragement when I was giving up, that made me carry on and fulfill my Master degree, and for all his friendship over these years;

And, specially, my great friend Professor Dr. Fabio Augusto de Alcântara Andrade, who shared with me many challenges over the years, since we taught together in the Naval Academy, for all his loyalty and friendship, that encouraged me in many troubled moments, whose wise and friendly words made me glimpse other ways and new horizons; but, also, for his remarkable professional skill – “a point out of the curve” –, that indubitably turned possible for me to arrive at this point.

RESUMO

Vigilância e inspeção com Veículos Aéreos Não-Tripulados com uso de sensores de imagem

Veículos Aéreos Não-Tripulados (VANTs) apresentam vasto e crescente emprego em diversas áreas, como aplicações militares, indústria, agricultura, entretenimento etc. Inserida neste escopo do Estado da Arte, esta tese apresenta duas partes, com soluções que atendem a missões autônomas de VANTs. Primeiramente, é proposto um algoritmo, com o emprego do Model Predictive Control (MPC), para utilização em missões de busca e salvamento com VANTs. Busca e salvamento (SAR) é uma área proeminente para uso de VANTs, em substituição de missões tripuladas. Nesta parte, é apresentada uma solução em tempo real de planejamento de trajetória, empregando múltiplos cooperativos VANTs. A técnica de otimização do enxame de partículas é utilizada no problema em Model Predictive Control (MPC), com o objetivo de realizar a busca em determinada área de interesse, seguindo a diretiva de padronização internacional de SAR. Um modelo cinemático coordenado para voo nivelado, com o efeito do vento, é incluído no MPC. A solução é implementada para ser embarcada no computador de bordo do VANT com DUNE, um software de bordo para navegação. O desempenho é avaliado usando o software-in-the-loop do Ardupilot, com o modelo de simulações da dinâmica de voo JSBSim. Os resultados mostram que, quando são empregados três VANTs, o grupo alcança 50% de Probabilidade de Sucesso, 2.35 vezes mais rápido do que apenas um VANT. A outra parte desta tese desenvolve um método de georeferenciamento direto de imagens, obtidas por sensores ópticos a bordo de VANTs. Neste contexto, a localização em um sistema geográfico de referência de uma imagem é uma tarefa necessária na maior parte das aplicações. Este processo é denominado georeferenciamento direto, quando não são empregados pontos de controle no solo. O processo completo de georeferenciamento direto envolve várias informações, como características dos sensores, medidas de construção, atitude e posição do VANT, entre outras. Além disso, há várias rotações e translações entre os diferentes sistemas de referência, fazendo do processo uma operação complexa. Outro problema é que fabricantes e softwares podem utilizar sistemas referenciais diferentes, acrescentando dificuldade à implementação. Para suprir esta demanda, esta pesquisa apresenta um método para georeferenciamento direto de imagens aéreas, obtidas de câmeras em VANTs, em que são explicadas em detalhes toda a informação requerida, operações matemáticas e etapas da implementação. Finalmente, para demonstrar a praticidade e a acurácia do método, são realizadas simulações e voos reais, em que as imagens obtidas são diretamente georeferenciadas.

Palavras-chave: Veículo Aéreo Não-Tripulado; planejamento de trajetória; busca e salvamento; model predictive control; Ardupilot; DUNE; software-in-the-loop; JSBSim; georeferenciamento direto; modelo de câmera “pinhole”

ABSTRACT

Unmanned Aerial Vehicles surveillance and inspection using optical imaging sensors

Unmanned Air Systems (UASs) have large and still increasing employment in many areas, such as military applications, industry, agriculture, entertainment etc. Inserted in this state of the art scope, this thesis presents two parts with solutions that will match many UASs autonomous missions. Firstly, it is proposed an algorithm, using Model Predictive Control, for UAS applications in search and rescue autonomous missions. Search and Rescue (SAR) is one of the most prominent areas for the employment of UASs, in place of a manned mission. In this part, a real-time path-planning solution, using multiple cooperative UASs for SAR missions, is proposed. The technique of Particle Swarm Optimization is used to solve a Model Predictive Control (MPC) problem, that aims to perform search in a given area of interest, following the directive of international standards of SAR. A coordinated turn kinematic model for level flight, in the presence of wind, is included in the MPC. The solution is fully implemented to be embedded in the UAS on-board computer with DUNE, an on-board navigation software. The performance is evaluated using Ardupilot's Software-In-The-Loop, with JSBSim flight dynamics model simulations. Results show that, when employing three UASs, the group reaches 50% Probability of Success, 2.35 times faster than when a single UAS is employed. The other part of this Thesis develops a method for direct georeferencing of images, gotten from optical sensors aboard the UAS. Optical image sensors are the most common remote sensing data acquisition devices present in Unmanned Aerial Systems (UASs). In this context, assigning a location in a geographic frame of reference to the acquired image is a necessary task in the majority of the applications. This process is denominated direct georeferencing when ground control points are not used. The complete direct georeferencing process involves much information, such as camera sensor characteristics, mounting measurements, attitude and position of the UAS, among others. In addition, there are many rotations and translations between the different reference frames, among many other details, which makes the whole process a considerable complex operation. Another problem is that manufacturers and software tools may use different reference frames, posing additional difficulty when implementing the direct georeferencing. In order to supply this demand, this research presents a comprehensive method for direct georeferencing of aerial images, acquired by cameras mounted on UAS, where all the required information, mathematical operations and implementation steps are explained in detail. Finally, in order to show the practical use of the method and to prove its accuracy, both simulated and real flights were performed, where objects of the acquired images are direct georeferenced.

Keywords: UAS; path planning; unmanned aerial vehicles; search and rescue; model predictive control; Ardupilot; DUNE; software in the loop; JSBSim; direct georeferencing; pinhole camera model

LIST OF FIGURES

Figure 1 – Official number of UAS registrations of use, among the Brazilian aviation agency (ANAC – Agência Nacional de Aviação Civil), since 2017 up to April, 2022. The graphics presents the quantities of registrations made by legal persons and by individual (natural, physical) persons. (Source: produced with information extracted from ANAC publication of data, actualized up to April , 2022) 14

Figure 2 – Total number of UAS in Brazil, separated by the kind of its use, in professional use and for entertainment. (Source: produced with information extracted from ANAC publication of data, actualized up to April , 2022) 15

Figure 3 – Structure of the thesis schematized. 22

Figure 4 – MPC controller structure and operation. 29

Figure 5 – Prediction horizon, measured from the actual instant K and extending to $K + p$. 30

Figure 6 – A set of $n = 3$ plant predicted scenarios, each one with p predicted states. 31

Figure 7 – The Error e_{K+i} from the steps $y\prime_{K+i}$ of the chosen best predicted scenario and the set point r. 32

Figure 8 – The new prediction horizon, shifted by one step of time. 32

Figure 9 – The first step of the selected scenario applied to the new shifted prediction horizon. 33

Figure 10 – MPC controller computes the plant feedback, the constraints and, produces new predicted scenarios for the new prediction horizon. 33

Figure 11 – Vertical (**left**) and horizontal (**right**) forces in the coordinated turn maneuver. 35

Figure 12 – Simplified System block diagram. 38

Figure 13 – Control task 39
Figure 14 – Unmanned Aerial System (UAS) agents flowchart. 40
Figure 15 – Unmanned Aerial System (UAS) agents communication. 41
Figure 16 – Software-In-The-Loop setup. 42
Figure 17 – X8 UAV. Source: NTNU 42
Figure 18 – Reward of cells. 44

Figure 19 – Angles between $X'Y'Z'$ and XYZ axes. 49
Figure 20 – Airplane yaw, pitch and roll maneuvers. 52
Figure 21 – Euler sequence: (a) Yaw; (b) Pitch; and (c) Roll. 52
Figure 22 – Camera Model. 54
Figure 23 – Pinhole Camera Model: (a) X axis view; and (b) Y axis view. 55
Figure 24 – Equivalent Camera Model: (a) X axis view; and (b) Y axis view. 55
Figure 25 – Equivalent Camera Model, in a more detailed perspective view, with the Image Frame top left orientation in the two axis U and V. 56
Figure 26 – Camera and Gimbal frames. 59
Figure 27 – UAS and Gimbal frames. 60
Figure 28 – NED and UAS frames. 61
Figure 29 – ENU and NED frames. 62
Figure 30 – Object's height in the ENU frame (z_{ENU} coordinate). 64
Figure 31 – Direct Georeferencing pseudocode 68
Figure 32 – The UAS path-planning autonomous mission control for a virtual reality solar plant inspection. 71
Figure 33 – The direct georeferencing of a defective solar panel image, taken from UAS RGB camera. 72
Figure 34 – The USN SeaDrone maritime autonomous vessel, showing the RGB-D camera, GPS and IMU on board sensors. 73
Figure 35 – The collision avoidance task in the AUTODRONE competition: the USN SeaDrone must make the decision between “stand on”, which means to continue in the planned route, or “give away”, changing the planned route. 74
Figure 36 – The scheme of the collision avoidance algorithm, employed in the USN SeaDrone, in the AUTODRONE collision avoidance task. 74

Figure 37 – Snapshot of a mission with three UASs being monitored with Neptus. 77
Figure 38 – Snapshot of the beginning of a mission. 78
Figure 39 – Time to reach 50% of Probability of success (POS). 79
Figure 40 – Average Probability of Success in time. 80
Figure 41 – Probability of success (POS) in 20 min of mission. 82
Figure 42 – Probability of success (POS) in 20 min of mission. 82
Figure 43 – Spiral path. 83
Figure 44 – Probability of Success. 83
Figure 45 – UAS in simulation environment. 85
Figure 46 – Camera placement on UAS in the simulation application. 85
Figure 47 – Picture acquired in the simulation environment. 86
Figure 48 – Parameters used for the direct georeferencing of the aerial imaged acquired in the simulation environment. 87
Figure 49 – ROS and georeferencing algorithm frames. 88
Figure 50 – Aerial image taken by camera mounted on UAS. 89
Figure 51 – Parameters used for the direct georeferencing of the aerial imaged acquired in a real flight. 89
Figure 52 – Pixel and georeferenced positions of table's top shelf's four corners. 90
Figure 53 – Georeferenced positions of bottom right corner of the top shelf and bottom right corner of the bottom shelf. 91
Figure 54 – Inspection product with pixel coordinates of defective area 93
Figure 55 – Location of the defective panels in the solar plant layout 93
Figure 56 – SeaDrone moving straight tracking a steady boat 95
Figure 57 – SeaDrone rotating around the tracked boat 95
Figure 58 – SeaDrone moving when tracking multiple moving objects 96

Figure 59 – Initial probability table example. 103
Figure 60 – Cells grid example. 105
Figure 61 – Mesh of the virtual solar panel 112
Figure 62 – Testure of the virtual solar panel 112
Figure 63 – Blueprint setup for the rotation function 113
Figure 64 – The unreal environment 114
Figure 65 – Drone flying in a virtual environment 115

Figure 66 – System flowchart 116
Figure 67 – Virtual Solar Plant 118
Figure 68 – Mission Planning. Gray line represents the solar panels pile; yellow line represents the calculated flight path; and red line is the normal between the drone flight path and the solar panels pile. 119
Figure 69 – Guidance algorithm flowchart 119
Figure 70 – Georeferencing in three steps 123
Figure 71 – Collision Avoidance Mission layout. 129
Figure 72 – System's block diagram. 130
Figure 73 – SeaDrone 131
Figure 74 – Direct georeferencing steps. 134
Figure 75 – Pseudo-code 136
Figure 76 – SeaDrone moving straight tracking a steady boat 138
Figure 77 – SeaDrone heading during the tracking 139
Figure 78 – SeaDrone rotating around the tracked boat 139
Figure 79 – SeaDrone heading when rotating around the tracked boat 140
Figure 80 – SeaDrone moving when tracking multiple moving humans 140
Figure 81 – SeaDrone heading when tracking multiple moving humans 140

LIST OF TABLES

Table 1 – Calculated length between shelf corners. 90

CONTENTS

1	**Introduction**	**14**
1.1	Background and Motivation	15
1.1.1	The problem of Search and Rescue	16
1.1.2	The problem of georeferencing	17
1.2	Objectives and Main Contributions	18
1.2.1	Search and Rescue Solution	18
1.2.2	Georeferencing Solution	19
1.3	Structure of the Thesis	20
1.4	Publications	22
1.5	Technical production	24
2	**Search and Rescue with Model Predictive Control**	**25**
2.1	Bibliographic revision and the state of the art	25
2.1.1	Researches employing UAS in SAR missions	25
2.1.2	Researches employing MPC	26
2.2	Model Predictive Control	28
2.2.1	MPC operation	28
2.2.2	The prediction horizon and the future plant states	29
2.2.3	The cost function and best scenario	30
2.3	Search and Rescue	34
2.3.1	Optimal Control Problem	34
2.3.2	Embedded System	37
2.3.3	Software-In-The-Loop Simulations Environment	41
2.3.4	Mission Simulation Scenario and Parameters	42
3	**Georeferencing**	**45**
3.1	Direct Georeferencing	45

3.1.1 Direct Georeferencing concept 46
3.2 Bibliographic revision and the state of the art 46
3.3 Mathematical Fundamentals 47
3.3.1 Rotation Matrix 47
3.3.2 Euler Angles Sequence for Aerodynamics: Yaw, Pitch and Roll 51
3.4 Direct georeferencing algorithm implementation 53
3.4.1 Image Frame to Camera Frame 53
3.4.2 Camera Frame to Gimbal Frame 58
3.4.3 Gimbal Frame to UAS Frame 60
3.4.4 UAS Frame to NED Frame 61
3.4.5 NED Frame to ENU Frame 62
3.4.6 Determination of z_C 63
3.4.7 Camera Matrix 65
3.4.8 Lens Camera Distortion 66
3.5 Software Implementation 67
3.5.1 MATLAB Implementation 67
3.5.2 Python Implementation 69
3.6 Direct georeferencing applications 70
3.6.1 Direct georeferencing of images of defective solar panels, taken by UAS cameras, in solar plants inspections 70
3.6.2 Direct georeferencing of images applied in a maritime collision avoidance system for small autonomous vessels 72

4 Results 76
4.1 Search and Rescue Results 76
4.1.1 Mission Scenario 76
4.1.2 Results 79
4.2 Direct Georeferencing Results 84
4.2.1 A simulation environment example with UE4 and AirSim and a practical example with ROS in an UAS real flight simulation 84
4.2.2 Direct georeferencing of RGB camera images in solar panels inspections in virtual environment 92

4.2.3 Results of the direct georeferencing of RGB-D images in collision avoidance systems for small autonomous vessels 94

5 Conclusions 97

5.1 Search and Rescue conclusions 97

5.2 Direct Georeferencing method conclusions 98

5.3 Proposals of future works 99

Appendix A - Search and Rescue Cost Function and Simulation Environment 100

Appendix B - Solar Plants Inspection Simulation Environment 107

Appendix C - Detection and tracking of crossing vessels for small autonomous vessels equipped with stereo camera 127

References 141

1- Introduction

The large use of Unmanned Air Systems (UASs) as a tool for professionals of many areas, and also for entertainment, is a reality worldwide, and shows enormous potential of increasing. In Brazil, this fact is confirmed by the Figures 1 and 2.

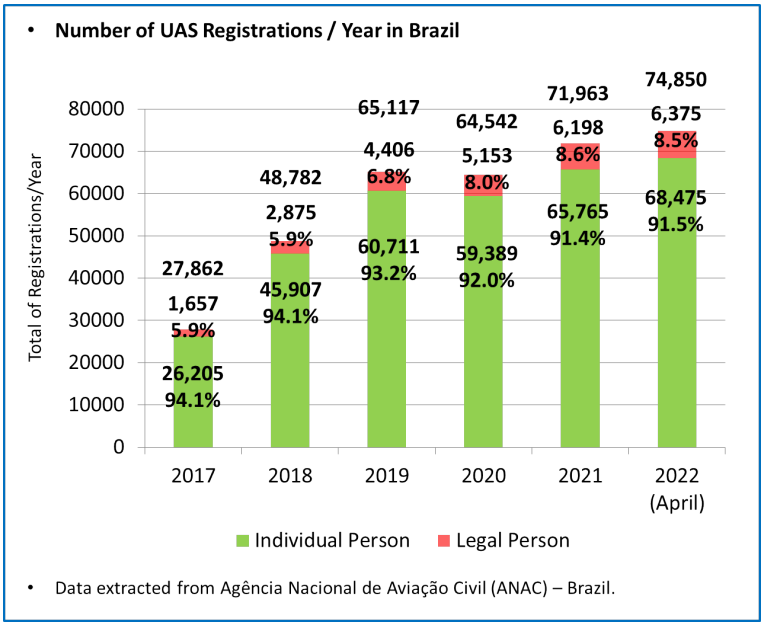


Figure 1 – Official number of UAS registrations of use, among the Brazilian aviation agency (ANAC – Agência Nacional de Aviação Civil), since 2017 up to April, 2022. The graphics presents the quantities of registrations made by legal persons and by individual (natural, physical) persons. (Source: produced with information extracted from ANAC publication of data, actualized up to April , 2022)

Figure 1 shows that mostly registrations in Brazil are proceeded by natural persons (individual or physical persons), and also indicates that its numbers, both of natural and legal persons registrations, are in increasing trend. In the same way, Figure 2 shows the increasing of the number of machines in operation, and that the greatest part of UAS in Brazil is allocated for professional applications, revealing its importance in economy. The two graphics demonstrate a clear trend of increasing in UAS applications, giving an idea of its respectable magnitude and importance in Brazilian society.

In the same way, (ANDRADE, 2019) enumerates a set of applications of UAS worldwide, in areas such as public security, agriculture, industry, as well as in military area, in strike-defense and surveillance missions.

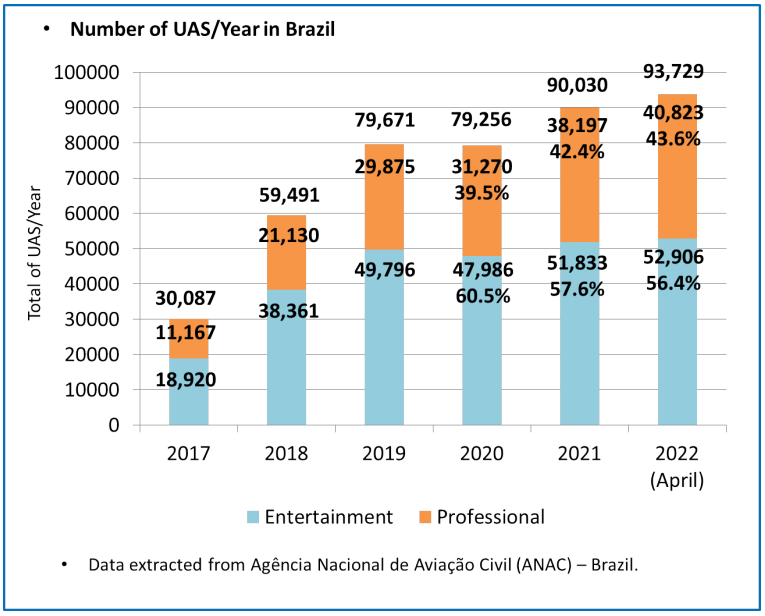


Figure 2 – Total number of UAS in Brazil, separated by the kind of its use, in professional use and for entertainment. (Source: produced with information extracted from ANAC publication of data, actualized up to April , 2022)

For all these uses, there is a great interest is in UAS autonomous operation, in order to obtain the advantages that automation brings, such as: precision, economy and safety.

1.1- Background and Motivation

In the many fields of activities that UAS are been employed, a wide set of missions are assigned, in order to take advantage of UAS features, such as friendly management, accurate data obtaining, robustness, facility in adapt onboard a great variety of devices and sensors, which allows enormous versatility to the UAS, making this equipment to become a very useful tool and, especially, with very acceptable costs and considerable benefits. Two of these applications - maritime Search and Rescue (SAR) missions and photogrammetry, with the direct georeferencing of the images, obtained by on board optical sensors, such as RGB or infrared cameras - are motivating many researches, seeking for solutions in many related problems. Inserted in this promising scope, this thesis in Instrumentation and Applied Optics presents the development of solutions for problems of UAS operations

in autonomous missions, using data obtained by the sensors commonly aboard (as Global Positioning System – GPS, gyroscopes, accelerometers, magnetometers, Pitot tubes etc), as well as optical sensors (cameras). This study consists in two main contributions, which brings its respective proposed unprecedented novelties for the UAS scientific community.

The first main contribution is a solution, using Model Predictive Control (MPC), that optimizes a target search in a given area, for practical application in real search and rescue UAS autonomous missions.

The other main contribution of this thesis is a method for direct georeferencing of images, obtained by UAS onboard cameras. It pays special attention in explaining in details all the steps of the complete process, in order to provide a helpful tool for researchers that may need to apply the method in their works, since the scientific works in this theme do not bring enough information about the complete direct georeferencing process.

1.1.1 The problem of Search and Rescue

Search and Rescue (SAR) is one of the fields where the employment of UASs brings many advantages over manned missions, such as its reduced costs, lower use of human resources, and mental and perception limitations of human operators. The process involves many variables, such as timing to find the target, equipment physical and technical limitations, as well as weather conditions etc, all of then respectively contributing in their own way to the success of the mission. In addition, real time circumstances permanently changes, demanding new decisions for each case. This scope of the problem requires a mathematical optimization algorithm capable to deal with many variables and constraints in real time.

In order to seek applicable mathematical solutions, MPC is a receding horizon control technique, where the motion constraints are integrated in the control problem, which is particularly interesting for problems with fixed-wing UASs. Also, as the optimization is done for a finite time horizon, the technique is proper for real-time problems where the environment can dynamically change during the mission execution. For this main reason, MPC is a useful tool in missions with such characteristics, such as maritime SAR missions

with the employment of UASs.

1.1.2 The problem of georeferencing

Traditionally, the georeferencing of aerial images was performed with the use of Ground Control Points (GCP) (ANDERSON, 1999; SKALOUD, 1999), which are marked points on the ground that have a known geographic location. However, the need of GCP imposes a considerable limitation in such operations, because, most of the time, the operator cannot place and measure the GCP prior to the mission. For example, in (ZHOU; LI; CHENG, 2005), the authors developed a large-scale video fast geo-registration method for forest fire monitoring, using high-resolution camera mounted in UAS. The developed system uses selected terrain points as references, called “landmarks ground control points”, LGCPs, whose positions and geographic coordinates are known from official geographic maps and loaded in the system, in order to precede the obtained image georeferencing. Despite the method may be satisfactory for land monitoring, it is limited for terrains where the mapping was not accurate or known, as well in case there will be no possibility to chose LGCP, as deserts or maritime regions, for example.

The process of georeferencing aerial images without the use of GCPs is called Direct Georeferencing. This process relies on using information about the position, attitude, image sensor and lens characteristics to calculate the position of the aerial image object in the geographic frame. This involves many challenges, such as that there is no range information available, i.e., direct distance between the camera and the target.

One may say that this could be simply solved by equipping the UAS with laser scanners or stereo cameras. However, these sensors are costly for long ranges and would also increase the weight of the UAS payload, reducing its endurance.

1.2- Objectives and Main Contributions

1.2.1 Search and Rescue Solution

In this thesis, a multiple UAS cooperative Nonlinear Model Predictive Control solution to search a given area is proposed. The technique of Particle Swarm Optimization is used to find the control inputs that solve the optimal control problem. A coordinated turn kinematic model is implemented considering the effects of wind. The search area is divided into cells and each cell has an associated reward that, in this work, is defined according to the international Search and Rescue directives. A Software-In-The-Loop (SITL) environment with flight dynamics simulations is used to test the solution. SITL simulations are more realistic numerical simulations that run in real-time and closed loop with the control system software, in which aircraft sensor data is simulated by a flight dynamics model.

In addition, it is assumed that there is always connection between UASs in the search area so that they can always share information between them. It is beyond the scope of this work to discuss network design. Research about communication and networks for UASs in maritime missions is available at (ZOLICH et al., 2015) and (ZOLICH et al., 2018), where communication technologies, standards and protocols are discussed and system architectures are proposed, including hardware description.

The main contributions of this research to the field are:

1. Research about the use of receding horizon techniques in exploration problems is limited. This research contributes to the field by bringing a novel multiple UASs solution to an exploration problem, using Model Predictive Control, where a finite time horizon grid search cost function with cells rewards and terminal cost is proposed.

2. The solutions found in the literature are developed with simplified vehicle models, in which the effects of wind are not considered when UAS platforms are employed. In small UAS missions, wind can easily exceed half of the UAS's airspeed, significantly affecting the UAS performance. In this work, a coordinated turn kinematic model, that takes the wind into consideration, is designed. The model is developed so that

the control inputs are the same as the autopilot control unit controls.

3. Usually, the solutions in the literature are only simulated in environments without embedded programming restrictions and where vehicle dynamics are not simulated. This makes results not close enough to what is expected in real-life applications. Implementing the solution in software for real-time applications brings additional challenges such as communication delays, processing time, actuator limitations, among others. In this research, the algorithm is fully implemented in an embedded software and tested in a real-time SITL environment that also simulates the flight dynamics, bringing results that are very close to reality.

4. In the literature, the SAR scenarios created for testing the proposed solutions are not based on standard directives. Therefore, making a fair comparison between different solutions is difficult. In this work, international SAR directives are considered to define the mission scenario and the performance indicators. This also allows testing the proposed solution in a relevant case.

1.2.2 Georeferencing Solution

Currently, optical image sensors are ubiquitous in UAS. These acquisition devices can be RGB, infrared or hyperspectral cameras, for example. There are a variety of applications using such sensors, e.g., in the fields of inspection (DE ALCANTARA ANDRADE et al., 2021), mapping (REES, 2019), search and rescue (RODIN et al., 2018), tracking (HELGESEN et al., 2021), border patrol (NOVAK, 2018), sea surveillance (SISK, 2018), agriculture (HONKAVAARA et al., 2013), and even recreation (LUNA, 2015). Although the method of direct georeferencing of images, taken by cameras mounted on UAS, demands mathematical concepts in linear algebra theory, the entire process involves many other details, resulting in a complex and tricky operation. Therefore, a researcher may face it difficult to find and understand these details, as they are treated separately and pulverized in many works in the scientific literature, increasing the risk of making small mistakes, which lead to big errors in the results. In fact, there is absolutely no paper in the literature that explains the overall process in a comprehensive way. This thesis adds to the

field a direct georeferencing method, detailing the complete implementation of the entire process, clarifying the concepts, and the reasons behind the choices. In this regard, the main contributions of this research are:

1. A georeferencing method employing ground control points (GCPs) is not suitable in some kind of environments, such as seas, deserts or very rough terrains. The direct georeferencing method proposed in this research dismisses the use of GCPs, allowing it to perform the georeference of images without such environments limitations;

2. The method is tested both in real and simulated environments, as there has been significant growth on the use of simulation tools for UAS applications;

3. The detailed determination of the z_C factor, briefly, or imprecisely, mentioned or not mentioned at all by the other works. In this research, the calculations involving the z_C factor is completely elucidated.

4. The inclusion of platform mounting measurements (lever arms) in the calculations. This is often neglected in the UAS community. Since the lever arms are in the range of centimeters, they are usually not considered. However, this can result on significant errors for high resolution sensors, or larger UAS.

Finally, the challenge that this work tries to overcome is the lack of centralized and standardized information about the methods to perform the georeferencing of aerial images, taken by cameras mounted on UAS. It is important to well understand the methods because there are many different frames involved in the process and the tools provided by sensors and systems manufacturers may use different frames and axis orientations. Therefore, it is important to understand the mathematical fundamentals and the implementation steps to be able to consciously make the required conversions and adaptations.

1.3- Structure of the Thesis

This thesis is divided into 5 chapters.

Chapter 2: In this chapter, the Search and Rescue with Model Predictive Control is presented. Search and Rescue (SAR) is one of the most prominent areas for the employment

of UASs in place of a manned mission, specially because of its limitations on the costs, human resources, and mental and perception of the human operators. In this section, a real-time path-planning solution using multiple cooperative UAVs for SAR missions is proposed. The technique of Particle Swarm Optimization is used to solve a Model Predictive Control (MPC) problem that aims to perform search in a given area of interest, following the directive of international standards of SAR. A coordinated turn kinematic model for level flight in the presence of wind is included in the MPC. The solution is fully implemented to be embedded in the UASs on-board computer with DUNE, an on-board navigation software. The performance is evaluated using Ardupilot's Software-In-The-Loop with JSBSim flight dynamics model simulations.

Chapter 3: In this chapter, the technique of direct georeferencing is presented. Optical image sensors are the most common remote sensing data acquisition devices present in UAS. In this context, assigning a location in a geographic frame of reference to the acquired image is a necessary task in the majority of the applications. This process is denominated direct georeferencing when ground control points are not used. The complete direct georeferencing process involves much information, such as camera sensor characteristics, mounting measurements, attitude and position of the UAS, among others. In addition, there are many rotations and translations between the different reference frames, among many other details, which makes the whole process a considerable complex operation. Another problem is that manufacturers and software tools may use different reference frames, posing additional difficulty when implementing the direct georeferencing. As this information is spread among many sources, researchers may face difficulties on having a complete vision of the method. In fact, there is absolutely no paper in the literature that explains this process in a comprehensive way. In order to supply this demand, this thesis presents a comprehensive method for direct georeferencing of aerial images acquired by cameras mounted on UAS, where all required information, mathematical operations and implementation steps are explained in detail.

Chapter 4: In this chapter, the results are presented. Results of the MPC in Search and Rescue show that when employing three UASs, the group reaches 50% of Probability of Success 2.35 times faster than when a single UAS is employed. Regarding the direct georeferencing, in order to show the practical use of the method and to prove its accuracy, both simulated and real flights were performed, where objects of the acquired images

were georeferenced.

Chapter 5: In this chapter, the concluding remarks are presented, as well as proposes for future works, leveraged by the research performed in this thesis.

Figure 3 presents a scheme of the complete structure of this thesis.

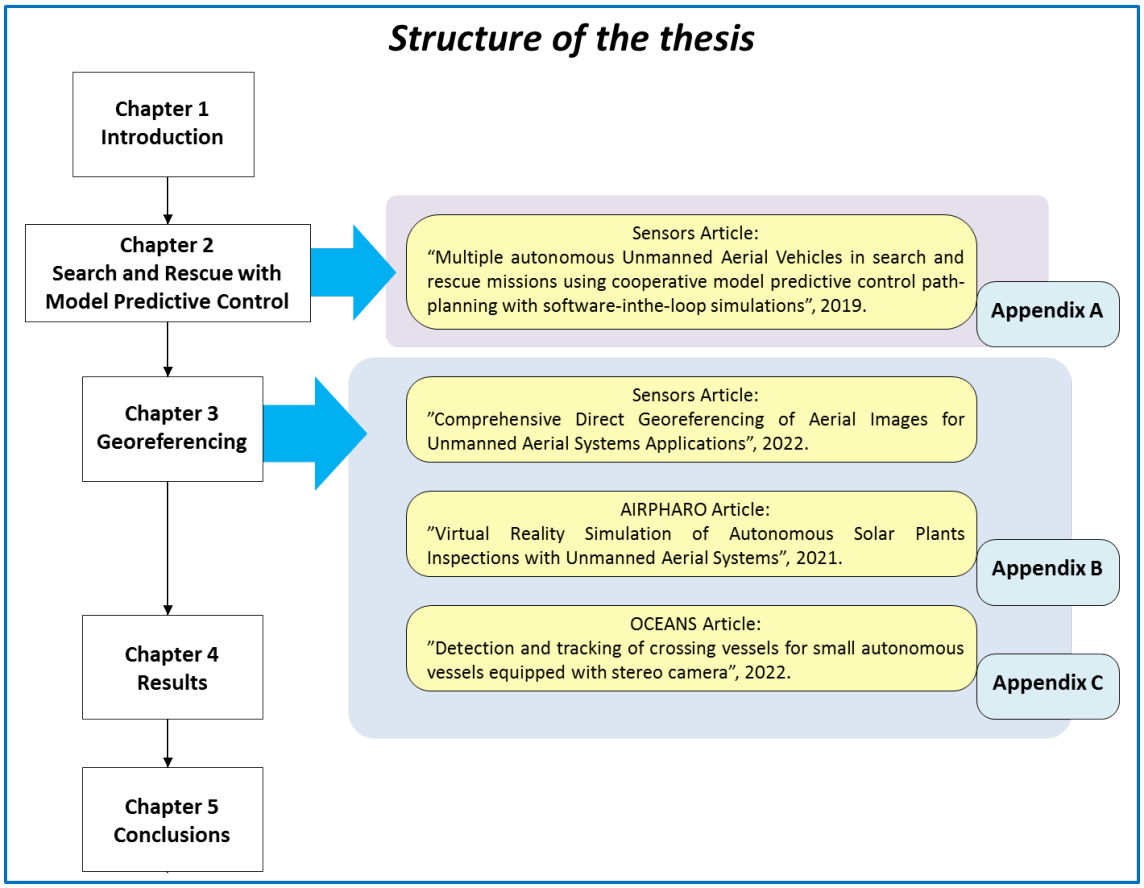


Figure 3 – Structure of the thesis schematized.

1.4- Publications

This thesis produced four scientific publications. Two of these articles were published in the reputed international journal Sensors (Impact Factor: 4.050; A1 - CAPES, Brasil), and the other two papers were published in the international conferences. This scientific production is listed bellow.

- Carlos A. M. Correia,Fabio A. A. Andrade, Agnar Sivertsen, Ihannah Pinto Guedes,

Milena Faria Pinto, Aline Gesualdi Manhães and Diego Barreto Haddad. “Comprehensive Direct Georeferencing of Aerial Images for Unmanned Aerial Systems Applications”. Sensors, 2021.

- Almost the entire article is described in Chapter 3 and the simulations and its results are described in Chapter 4.

- Fabio A. A. Andrade, Anthony Hovenburg, Luciano Netto de Lima, Christopher Dahlin Rodin, Tor Arne Johansen,Rune Storvold, Carlos A. M. Correia and Diego Barreto Haddad. “Multiple autonomous Unmanned Aerial Vehicles in search and rescue missions using cooperative model predictive control path-planning with software-in-the-loop simulations”. Sensors, 2019.

  - Parts of this publication that present the introduction and the bibliographic review are included in Chapter 1; parts about the optimal control problem, the coordinated turn model, the model predictive control problem, the embedded system and the simulation environment are included in Chapter 2; parts that describes the optimization technique, the SAR directives applied to UAVs equipped with remote sensing, search area, probability map and cost function, are exposed in Appendix A; and the results are included in the Chapter 4.

- Fabio Augusto De Alcantara Andrade, Agnar Holten Sivertsen, Carlos Alberto Moraes Correia, Nabil Belbachir, Lucas Costa de Sousa, Victor Müller Pereira Rufino, Eduardo Pimenta Petrópolis, Erick Rodrigues e Silva, Victor Hugo Rinaldi Fortes Henriques. “Virtual Reality Simulation of Autonomous Solar Plants Inspections with Unmanned Aerial Systems”. Aerial Robotic Systems Physically Interacting with the Environment. AIRPHARO, 2021.

  - This research is briefly described in Chapter 3, its results presented and commented in Chapter 4, and the full text of the publication is exposed in Appendix B.

- Fabio A. A. Andrade, Marius Tannum, Adrian Knutsen, Nina Tran, Torben Falleth Olsen, Tobias Tufte, Carlos A. M. Correia. “Detection and tracking of crossing vessels for small autonomous vessels equipped with stereo camera”. OCEANS, 2022.

  - This research is briefly described in Chapter 3, its results presented and commented in Chapter 4, and the full text of the publication is exposed in the Appendix C.

1.5- Technical production

This thesis produced a MATLAB algorithm for the computational implementation of the direct georeferencing method, proposed and described in Chapter 3. This technical production is available in the following repository:

https://github.com/professorfabioandrade/georef

2- Search and Rescue with Model Predictive Control

2.1- Bibliographic revision and the state of the art

In order to provide a notice in the main themes of this Chapter, allowing an overview of the state of the art in the area, this bibliographic revision brings some researches, produced among the UAS community, about solutions for UASs in SAR missions, employing different algorithms and techniques. As well, some works employing MPC are discussed.

2.1.1 Researches employing UAS in SAR missions

The work (SENGUPTA et al., 2010) was one of the first works to perform experimental tests of a complete autonomous single UAS SAR solution. A probability density function (PDF) that expressed the likelihood of the target's location was one of the main inputs of the system. Video data from the UAS was transmitted to the ground station, that processed it in real time using computer vision techniques to detect the presence of the target and update the PDF. Paths were generated by the ground station to maximize the probability of finding the targeted object. The experimental flights showed satisfactory results in searching and detecting the target. The main necessary improvements identified by the authors were to implement on-board computing and to use multiple UASs in the future.

SAR missions with autonomous UASs are usually defined as an exploration problem. Exploration approaches can be used in a wide range of applications. For example, ice management, such as proposed by (HALS; SKJONHAUG, 2017), where a Centralized Model Predictive Search Software was used for surveillance and tracking of ice, using multiple UASs. In the solution proposed by this reference, the algorithm finds a set of optimal waypoints that are sent to the autopilot control unit. The solution was tested in a Software-In-The-Loop environment and the results were evaluated for a different number

of UASs. (ALOTAIBI; ALQEFARI; KOUBAA, 2019) compared five exploration algorithms for SAR missions, using a team of multiple UASs. A centralized mission planner was used for the path-planning, whose objective was to find a sequence of waypoints for each UAS to follow. In (SAN JUAN; SANTOS; ANDUJAR, 2018), an off-line algorithm to find a sequence of waypoints, to be followed by a team of UASs in a SAR mission, is proposed. The cost function is built to minimize the travelled distance and the risk that people are exposed to. The authors created a non-uniform “risk map” and ”probability of people map” for the simulated scenario.

A broad literature review about the persistent surveillance problem was done by (NIGAM, 2014), focusing on the use of multiple UASs. Persistent surveillance is a type of exploration problem where the areas must be revisited over time. Among the many topics that the literature review covers, grid decomposition and path-planning techniques are the ones of the most interest for this work. The author reviews the most common types of grid decomposition, classifying the rectangular one, that is also used by the enhanced solution proposed by this work, as the most popular. Regarding the path-planning techniques, the author states that the most common methods are classical search methods such as A* (JAKOB et al., 2010), decision theoretic methods such as Mixed Integer Linear Programming (MILP) (FORSMO et al., 2013), and Spanning Tree Coverage (STC) methods (GABRIELY; RIMON, 2001). Model Predictive Control (MPC) (CAMACHO et al., 2010) is mentioned as a topic less studied, compared to the other planning techniques, but with significant advantages because it directly incorporate dynamic constraints, it is less heuristic and can react to changes in the environment.

2.1.2 Researches employing MPC

In (SKJONG et al., 2015), MPC was used for sea Search and Track missions using an autonomous UAS. Hardware-In-The-Loop tests were performed. Waypoints were optimized and sent to the autopilot. Gimbal attitude was also optimized and sent to the servo system. The MPC optimization was not run on-board but on a dedicated computer in the ground control station. In (WU et al., 2018), a MPC solution was proposed for target tracking using a solar-powered UAS. Energy harvesting and consumption were also

included in the cost function. A three-dimensional dynamic model was used with the thrust, roll angle and lift coefficient as control inputs. The model also included horizontal winds. (WU et al., 2017) describes a tracking application using multiple solar-powered UASs in urban areas. A two-dimensional kinematic model is used, without considering the wind. The cost function includes the energy harvesting and considers the shadows made by the buildings. Energy consumption is also considered. In addition, a distance constraint was included to guarantee the communication quality. The cooperative solution used is similar to the one described in this study, where the UASs share their planned states. The meta-heuristic technique Grasshopper Optimization Algorithm is used for the optimization. A performance comparison, between optimization techniques, is presented, including the performance of Particle Swarm Optimization algorithm, which is used in this study. In this reference, the cost function is composed by the feasibility cost, the mission cost, the energy cost, the safety cost, the collision cost and the communication cost. (CHAO et al., 2012) proposed a decentralized MPC for formation flight with obstacle avoidance. The chosen formation strategy was to keep a desired distance and angle between the UASs and a moving reference point, representing a UAS, which follows the reference trajectory. Collision avoidance was also included in the cost function, combined with a priority strategy where lower priority UASs treat other UASs as obstacles, if they are within the protected radius. A two-dimensional UAS kinematic coordinated turn model without considering the wind was used. Numerical simulations were done using MATLAB, where flight dynamics simulations neither control system software closed loop were considered. In (HOY; MATVEEV; SAVKIN, 2012), a collision avoidance decentralized MPC solution was proposed for multiple wheeled vehicles. External disturbances were also included in the form of translational, rotational, and speed alterations. A high level trajectory planning module was responsible for finding the planned path, based on a nominal model and a low level trajectory tracking module, was responsible for ensuring that the vehicles follow the planned trajectory facing the disturbances. MATLAB simulations were done for a wide variety of scenarios and number of vehicles and experiments were done using two robots in order to show real-time implementation. In (FARROKHSIAR; PAVLIK; NAJJARAN, 2013), a similar strategy of dual control MPC was used. The solution was applied to a scenario with a single robot that aims to reach a given destination. Static and dynamic obstacles were simulated and measurement noise and process disturbance were included. Performance and real-time implementation were tested by simulations and experiments

with a ground mobile robot. In (ALCANTARA ANDRADE et al., 2018), a cooperative multiple UASs solution, using MPC, was used to close the communication link between a moving Autonomous Surface Vehicle and the ground station. Each UAS had to minimize a local cost function that took into consideration the planned states of the adjacent UASs. In (DI; ZHOU; DUAN, 2015), a multiple UASs receding horizon strategy was proposed for a cooperative surveillance problem. A potential field method was used for collision avoidance and network topology control management. The cooperative searching model was established based on the detection probability of the UASs on targets in cells. In addition, a forgetting factor was included to indicate how fast the detection efforts are forgotten, so the UASs can revisit the areas that were searched before. Simulations for different parameters were compared. Also, the performance of the proposed method was compared to the performance of a parallel sequence search. In (TRODDEN; RICHARDS, 2008), a multi-vehicle cooperative search solution was proposed using MPC. Decoupled, centralized, cooperative and greedy approaches were compared.

2.2- Model Predictive Control

Model predictive control (MPC) is a control algorithm that employs simulations of the future states of the system and selects the state that is the most approximate to the reference, which are the desired conditions for the system behavior. The model is designed for multiple input and multiple output (MIMO) systems and deals with constraints as well.

2.2.1 MPC operation

The MPC algorithm employs two main parts: an optimizer and the plant model.

The optimizer produces a number of n control input sequences of p ($p \in \mathbb{N}$) steps, that when are input to the model, produces a number of n simulations of the system behavior over p futures steps, which are denominated the prediction horizon.

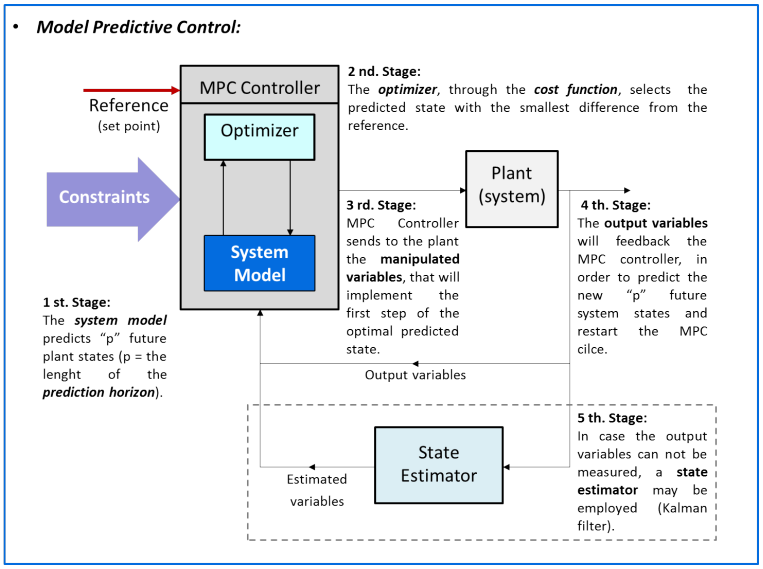


Figure 4 – MPC controller structure and operation.

The optimizer selects the simulated behavior with the smallest difference to the reference, or set point. The optimizer does this choice through the cost function. The MPC controller, then, transmits to the plant only the first state of the chosen simulation. The MPC cycle is, then, restarted, with the creation of a new prediction horizon, whose the first stage is the next step of the previous prediction horizon; for example, if the first prediction horizon started in moment K and finish in moment $K + p$, then the next prediction horizon will start in moment $k' = K + 1$ and finishes in moment $K' + p = K + p + 1$. The controller will compute the plant feedback and the constraints, and the plant model will produce new n simulated behaviors of the plant, for the next p steps of the new prediction horizon. This loop will be repeated until the plant state is equal, or in an acceptable difference of, the set point. Figure 4 summarizes the MPC controller operation previously explained.

All the steps of the algorithm, described in Figure 4 will be explained below, giving a complete view of the MPC operation.

2.2.2 The prediction horizon and the future plant states

The MPC controller determines a period of time where a set of predicted states of a plant will be simulated. Its length is p and is defined considering the plant dynamics and hardware limitations. For each of the n behavior simulations, a number p of states are

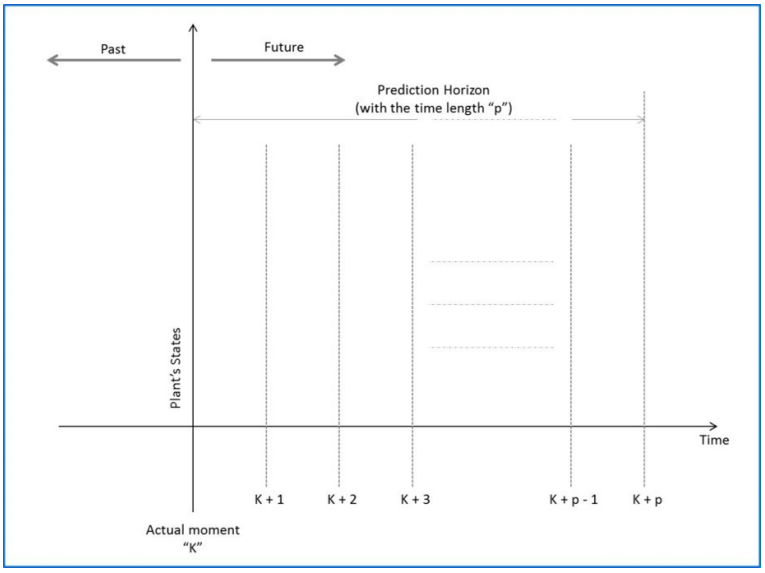


Figure 5 – Prediction horizon, measured from the actual instant K and extending to $K + p$.

predicted. Figure 5 schematizes a prediction horizon, with p steps.

From the actual instant K, the MPC controller determines an interval with length p where a set of n plant behaviors will be simulated, each one with p predicted states, which correspond to each step of the prediction horizon. These predictions are possible because the MPC has a plant model, which provides the states simulations and consider the constraints. Figure 6 shows a set of three plant simulations, or scenarios, with p states each for the p steps of the prediction horizon.

2.2.3 The cost function and best scenario

The optimizer calculates the best predicted scenario, which is the one with the lowest difference to the set point. The quality of the scenario is given by the cost function J. For example, if a metaheuristic optimization technique is used, it calculates the error $e(N)_{K+i}(1 \leq i \leq p)$ between each step $y\prime(N)_{K+i}(1 \leq N \leq n)$, of all the n predicted scenarios produced by the system model, to the set point r.

$$e(N)_{K+i} = \left| y\prime(N)_{K+i} - r \right| \tag{1}$$

where $y\prime(N)_{K+i}$ is the simulated predicted state $y\prime$ of scenario $N \in n$ in step $K + i$, r is

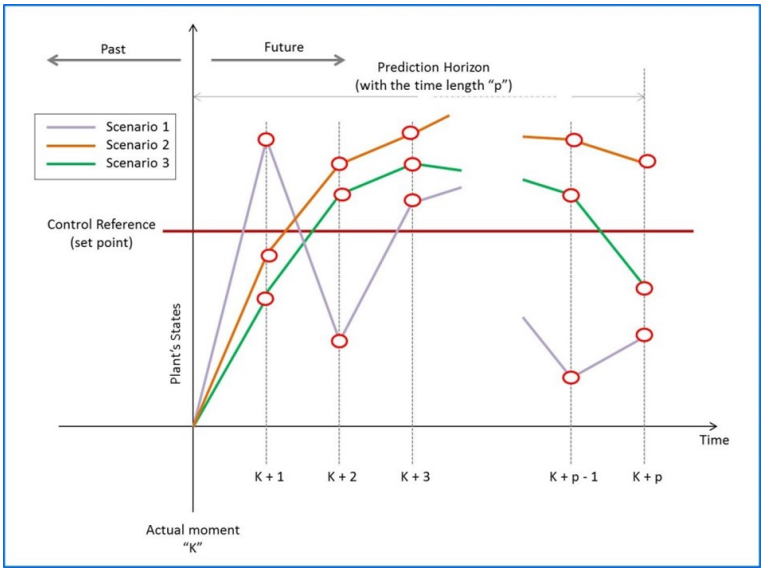


Figure 6 – A set of $n = 3$ plant predicted scenarios, each one with p predicted states.

the set point and $e(N)_{K+i}$ is the error of scenario N in the step $K + i$, $i \in p$.

Figure 7 shows the error e_{K+i} in a chosen predicted scenario.

The cost function may also include the constraints, as well as other terms, in order to calculate the best scenario. The chosen best predicted scenario will be the one which presents the minimum value of the weighted square sums of the errors and of the other terms (constraints), as shown in the equation:

$$J = \ min \left[\sum_{i=1}^{p} w_e.(e_{K+i})^2 + \sum_{m=1}^{r} w_C.(C_m)^2 \right] \tag{2}$$

where w_e is the weight of the error e_{K+i} and w_C is the weight of the constrain C_m.

Once the best scenario is chosen by the optimizer, based on the cost function, the prediction horizon shifts forward by one time step, and the MPC controller applies only the first step of this chosen scenario, disregarding the rest of the steps. Figure 8 presents the new shifted prediction horizon and Figure 9 shows the first step of the chosen scenario applied to this new scenario.

Following, the MPC controller restarts the cycle. The controller receives new sensor measurements of the plant's outputs or estimates the states if no sensor measurements are available. Beginning from the first step of the previously selected scenario, the new predicted scenarios are produced for the present (shifted) prediction horizon. Figure 10 shows the MPC cycle restart.

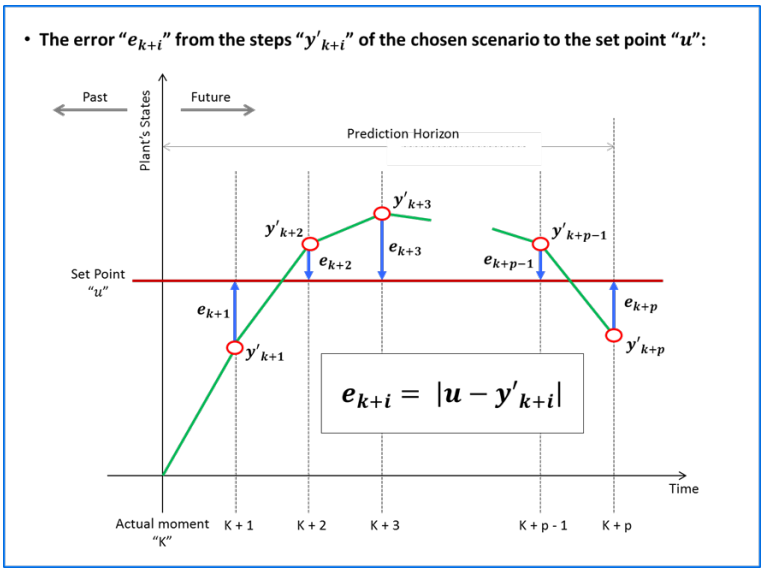


Figure 7 – The Error e_{K+i} from the steps $y\prime_{K+i}$ of the chosen best predicted scenario and the set point r.

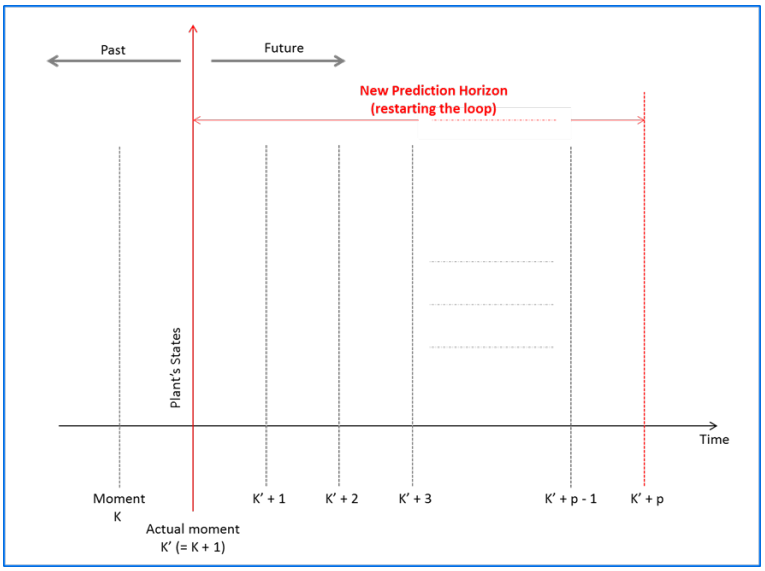


Figure 8 – The new prediction horizon, shifted by one step of time.

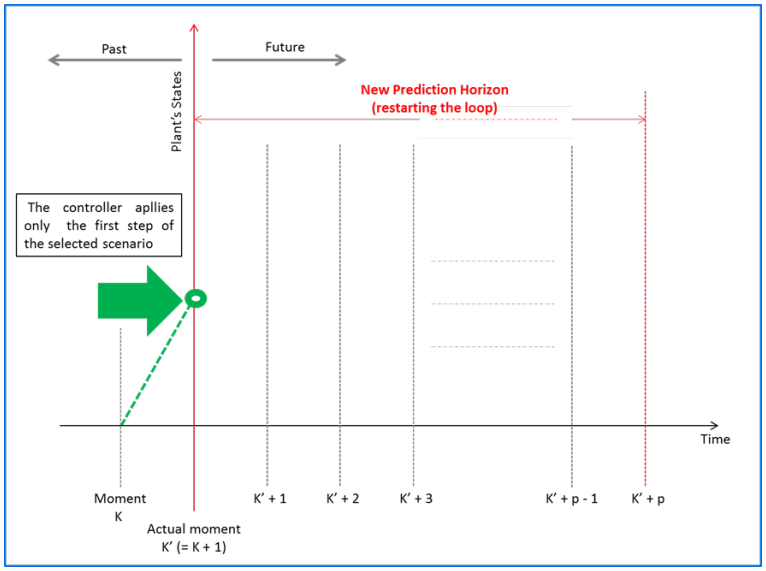


Figure 9 – The first step of the selected scenario applied to the new shifted prediction horizon.

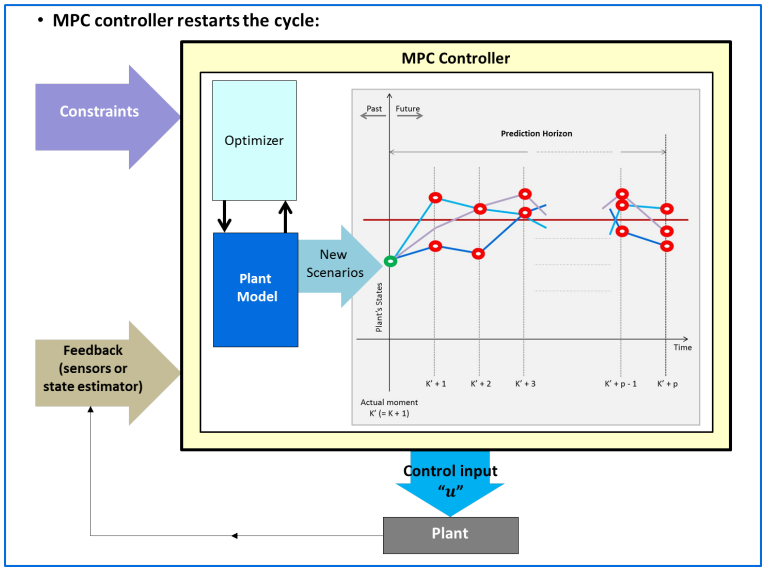


Figure 10 – MPC controller computes the plant feedback, the constraints and, produces new predicted scenarios for the new prediction horizon.

2.3- Search and Rescue

2.3.1 Optimal Control Problem

Coordinated Turn Model

In view of the optimal control problem solution, two-dimensional kinematic model was developed based on the Coordinated Turn model for level flight (BEARD; MCLAIN, 2012). In this model, the UAS turns by changing its roll angle so that there is no net side force acting on the UAS. As the effects of the increased angle of attack to compensate the lift losses when rolling do not have a significant impact on the kinematics, they are not included in the model for simplicity. Therefore, it is possible to relate the course rate and the roll angle by making the centripetal force acting on the UAS equal to the horizontal component of the lift force acting in the radial direction (Figure 11):

$$F_{lift} \sin u_\phi \cos(\chi - \psi) = m\frac{v_g^2}{R} = m\frac{\dot{\chi}^2 R^2}{R} = m\dot{\chi}^2 R = m v_g \dot{\chi}, \tag{3}$$

where R is the radius of the trajectory in [m], F_{lift} is the lift force in [N], u_ϕ is the roll control input in [rad], χ is the course angle in [rad], ψ is the aircraft heading in [rad], m is the aircraft mass in [kg], v_g is the ground speed in [m/s] and $\dot{\chi}$ is the course rate in [rad/s].

In addition, the vertical component of the lift force should be equal and opposite to the projection of the gravitational force (Figure 11):

$$F_{lift} \cos u_\phi = mg, \tag{4}$$

where g is the gravitational acceleration of 9.81 m/s^2.

By combining Equations (3) and (4), it is possible to find the relation between the course rate and the roll control input:

$$\dot{\chi} = \frac{g}{v_g} \tan u_\phi cos(\chi - \psi). \tag{5}$$

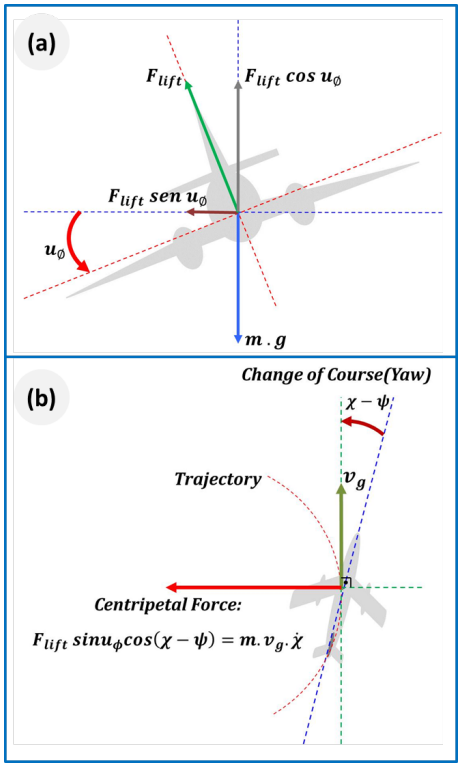


Figure 11 – Vertical (**left**) and horizontal (**right**) forces in the coordinated turn maneuver.

As wind is a major issue on UAS missions as it can likely reach more than half of the UAS's maximum airspeed, the Coordinated Turn model used in this project was developed to consider the influence of wind on the UAS kinematics.

Therefore, the coordinated turn kinematic model for level flight in the presence of wind is given by:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\chi} \end{pmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} v_g \cos\chi \\ v_g \sin\chi \\ \frac{g}{v_g} \tan u_\phi cos(\chi - \psi) \end{pmatrix}, \tag{6}$$

where $\mathbf{x} = (x, y, \chi)$ are the north and east positions in the NED (North-East-Down) frame in [m] and the course angle in [rad], respectively. $\mathbf{u} = (u_v, u_\phi)$ are the airspeed control input in [m/s] and roll control input in [rad], respectively, and with the ground speed (v_g in [m/s]):

$$v_g = \sqrt{(u_v \cos\psi + v_w \cos\psi_w)^2 + (u_v \sin\psi + v_w \sin\psi_w)^2}, \tag{7}$$

where v_w is the wind speed in [m/s], ψ_w is the wind heading in [rad] and with the aircraft

heading (ψ in [rad]) calculated using the law of sines:

$$\psi = \chi - \arcsin\Big(\frac{v_w}{u_v}\sin\left(\psi_w - \chi\right)\Big). \tag{8}$$

The model is discretized by the forward Euler method:

$$\mathbf{x}_{k+1} = f_d(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + T_s f(\mathbf{x}_k, \mathbf{u}_k), \tag{9}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ are the state and control inputs vector, respectively, in the discretization step k and T_s is the sampling period.

Model Predictive Control Problem

To reach the mission goal, a centralized optimization approach might not be feasible because the problem would be too complex with too many control inputs. In a non-convex problem with a very long vector of variables to optimize, falling very early in a local minimum is a common issue. In addition, the necessary processing power to optimize so many control inputs would be difficult to achieve by the on-board processing unit of the UAS. In contrast, optimizing the controls of all UASs in a ground station would not be an ideal solution, due to communication range limitations and because, in case of a communication failure, the UASs would not receive its controls, which could compromise the mission.

Therefore, in this research, the problem is addressed as a cooperative control problem, where each UAS optimizes its own control inputs to update its state so that a local cost function is minimized. The cost function also takes into consideration the planned states of the other UASs. As each UAS follows the same process, it is expected that the global mission goal is achieved cooperatively. Collision avoidance between UASs is also considered.

Considering I UASs ($\mathbf{x}^i, \forall i \in \{0, ..., I-1\}$), the algorithm finds a control input sequence $U_k^i = \{\mathbf{u}_0^i, \mathbf{u}_1^i, ..., \mathbf{u}_{K-1}^i\} \in R^{2\times K}$ for the ith UAV, which solves the following

optimal control problem:

$$\text{minimize} \quad \delta(\mathbf{C}\bar{\mathbf{x}}_K) + \sum_{k=0}^{K-1} L^i(\mathbf{C}\bar{\mathbf{x}}_k, \mathbf{u}_k^i), \tag{10}$$

$$\text{subject to} \quad \mathbf{x}_{k+1}^i = f_d(\mathbf{x}_k^i, \mathbf{u}_k^i), \tag{11}$$

$$v_{a_{min}} \leq u_{v_k}^i \leq v_{a_{max}}, \tag{12}$$

$$\phi_{min} \leq u_{\phi_k}^i \leq \phi_{max}, \tag{13}$$

$$|\mathbf{C}(\mathbf{x}_k^i - \mathbf{x}_k^j)| > r_c, \forall j \in \{0, ..., I-1\} \backslash \{i\}, \tag{14}$$

where

$$\delta(\mathbf{C}\bar{\mathbf{x}}_K) = F(\mathbf{C}\mathbf{x}_K) - aJ(\mathbf{C}\bar{\mathbf{x}}_K), \tag{15}$$

and

$$L^i(\mathbf{C}\bar{\mathbf{x}}_k, \mathbf{u}_k^i) = aJ(\mathbf{C}\bar{\mathbf{x}}_k) + b(u_{v_k}^i - u_{v_{k-1}}^i)^2 + c(u_{\phi_k}^i - u_{\phi_{k-1}}^i)^2. \tag{16}$$

Consider $u_{v_{-1}}^i$ and $u_{\phi_{-1}}^i$ as the commanded airspeed and roll angle for the ith UAS, respectively, in the previous optimization loop, $\bar{\mathbf{x}}_k = [\mathbf{x}_k^0, ..., \mathbf{x}_k^{I-1}]$ as the states of all UASs, K as the number of horizon steps and r_c as the minimum safe distance between the UASs to avoid collision. a, b, c are constant weighting factors and $\mathbf{C} \in R^{2\times 3}$ is used to define that only the x (north) and y (east) positions are used from the state vector:

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{17}$$

In Equation (15), the function J represents the grid search function, which is the sum of the rewards of visited cells, and F is the terminal cost (cost-to-go) function, which is the distance from the terminal position to the unvisited cell with highest reward. Both functions are described in detail in Section 5.3.

2.3.2 Embedded System

The path-planning algorithm was implemented as a task in DUNE: Unified Navigation Environment (PINTO et al., 2013). It allows the operation of a wide variety of robots

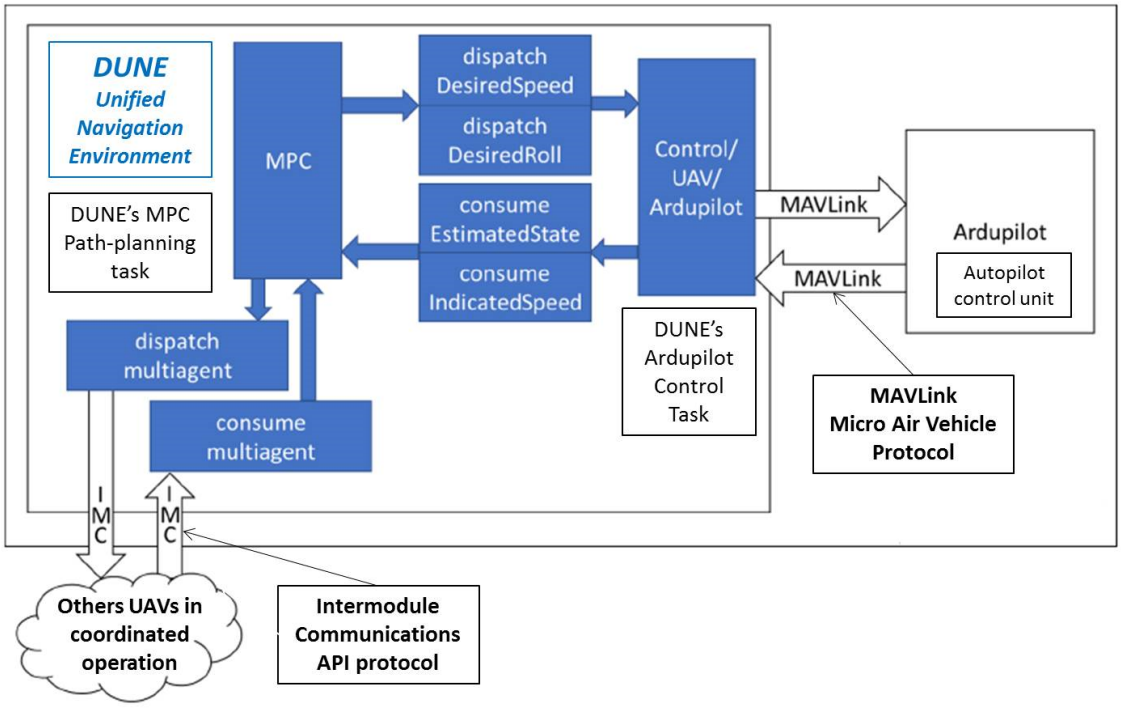


Figure 12 – Simplified System block diagram.

using the same environment. This facilitates the development because the communication between DUNE and the different control units is transparent to the user.

The embedded system is outlined in Figure 12 and Figure 13. The communication between DUNE and the Ardupilot (ARDUPILOT. . . , 2019) autopilot control unit is done via MAVLink Micro Air Vehicle Protocol (MAVLINK. . . , 2019). To command the Ardupilot, DUNE's MPC task must dispatch a message with the desired command, which will be interpreted by DUNE's Ardupilot control task and then sent to the Ardupilot via MAVLink.

The communication between DUNE tasks is done via the IMC: Intermodule Communication API protocol (PINTO et al., 2013). This protocol basically operates by dispatching and consuming messages. Thus, if a message is dispatched by a task, another task that is waiting for that message will consume it.

The communication with other UASs in DUNE's cloud and with the ground station is also done via IMC protocol. The vehicles must be in the same network. Therefore, IP addresses and TCP and UDP ports must be properly configured and included in each vehicle's DUNE configuration files. To allow the communication between vehicles, the InterVehicleCommon UDP transport module is used.

The MPC task is inside DUNE. The commands to control the UAS are given by the DesiredSpeed and DesiredRoll IMC messages, which carry, respectively, the airspeed and

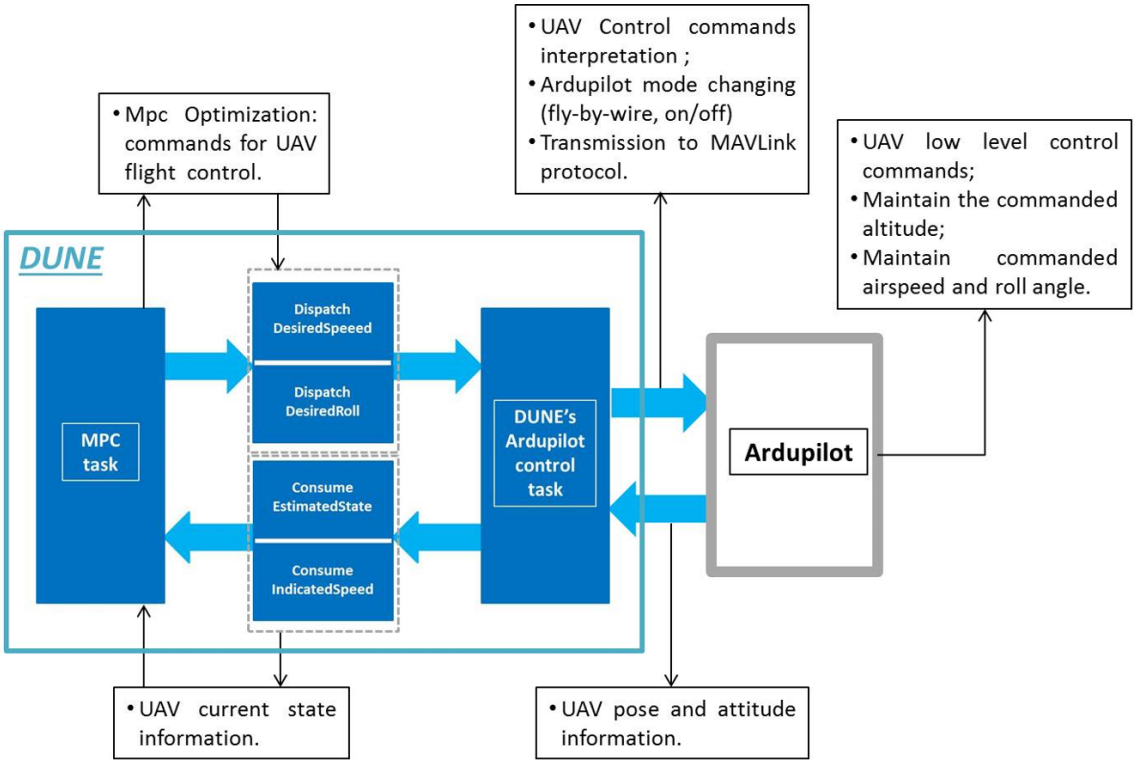


Figure 13 – Control task

roll control inputs given by the optimization. These messages are interpreted by DUNE's Ardupilot control task that sends the correspondent MAVLink message to the Ardupilot unit. In addition, DUNE's Ardupilot control task is responsible for changing the Ardupilot flight mode to the Fly-By-Wire-B (FBWB) mode when the mission is started. In this flight mode, the Ardupilot control unit is responsible for holding the aircraft's altitude and compensating the loss of lift caused by the rolling. Airspeed, roll angle and altitude external controls are accepted by the Ardupilot in the FBWB mode. The Ardupilot control unit is responsible for the low level control loops to maintain the commanded altitude, airspeed and roll angle. In this thesis, only roll and airspeed are controlled by the optimization algorithm; therefore, the altitude control is fixed at the desired altitude during the whole mission.

DUNE's Ardupilot control task is also responsible for receiving the UAS's pose and attitude information and dispatching it in the IMC messages EstimatedState and IndicatedSpeed. These messages are consumed by the MPC task to be used as the current state of the UAS.

The communication between the UASs is done by the multiagent message, which was created and included in the IMC messages list specifically for this application. This message carries the information that the UASs need to share between them, such as

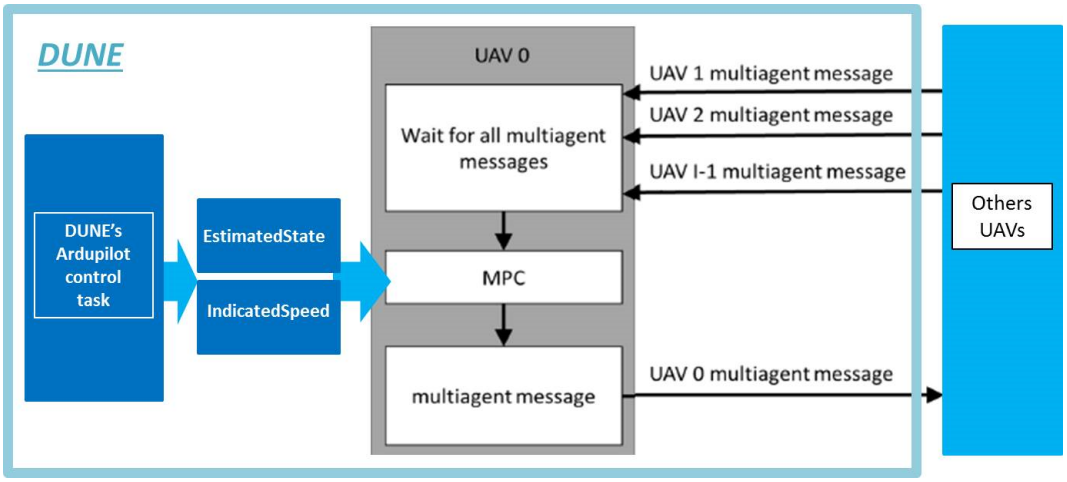


Figure 14 – Unmanned Aerial System (UAS) agents flowchart.

planned control inputs and current state.

Each UAS waits for the multiagent messages from all other UASs before running the MPC optimization, e.g., if a team of three UASs is employed, UAS 0 only runs the optimization algorithm after receiving the multiagent messages with the needed information from UASs 1 and 2. This flow is described in Figures 14 and 15. Once the messages are received and the optimization loop is over, the UAS dispatches its multi agent message containing all information that need to be shared with the other UASs.

In order to prevent control issues caused by communication and processing delays, the UAS states used by the MPC optimization are predicted according to the previously commanded control inputs and the measured delays by using Equation (9). Therefore, in the end of the optimization loop, the real states of the UASs are expected to be close to the ones considered by the optimization algorithm when finding the optimal control inputs. This results in a superior behavior since the control inputs are obtained for UASs attitudes and positions that are closer to reality.

Note that, in an extension of the method proposed by this work, measures can be employed to protect the system from communication failures, so that the UASs do not wait for delayed messages for too long but use predictions instead. In addition, distance constraints can be included in the MPC in order to guarantee that the UASs do not separate further than the maximum distance that enables the communication.

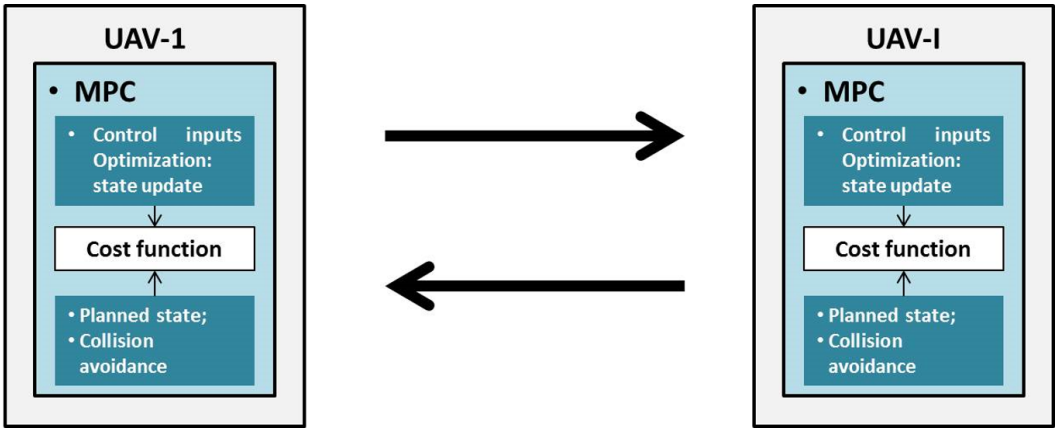


Figure 15 – Unmanned Aerial System (UAS) agents communication.

2.3.3 Software-In-The-Loop Simulations Environment

To evaluate the proposed solution, a Software-In-The-Loop (SITL) environment was set up using Ardupilot SITL simulator. This simulator allows for testing the behavior of the Ardupilot code by running the Ardupilot in any computer without the Ardupilot hardware. The aircraft sensor data are simulated by JSBSim (BERNDT, 2004), an open source Flight Dynamics Model. Therefore, it is able to compute the UAS dynamics according to the actuator controls given by the Ardupilot code.

An aircraft platform model must be chosen for JSBSim flight dynamics calculations. In this work, the Skywalker X8 UAS platform is used. The X8 is a UAS with around 4.0 to 4.5 kg of weight including the payload, 2.1 m of wingspan and 35.7 cm of mean aerodynamic chord. The aerodynamic model used to feed the JSBSim configuration file was developed based on wind tunnel tests (GRYTE et al., 2018). In addition, the JSBSim source code was modified to use in its calculations the same wind map that is used by the MPC optimization.

Figure 16 shows the interconnection between modules. For each UAS, an Ardupilot SITL instance must be started linked to a JSBSim module. Each Ardupilot instance uses a different TCP port. Therefore, one DUNE module must be started for each UAS, configured with the correspondent TCP port. Finally, Neptus (PINTO et al., 2013), a command and control software, is used to visualize the UASs telemetry and location and to give commands to the UASs, such as take off, loiter and to start/stop the Search and Rescue mission.

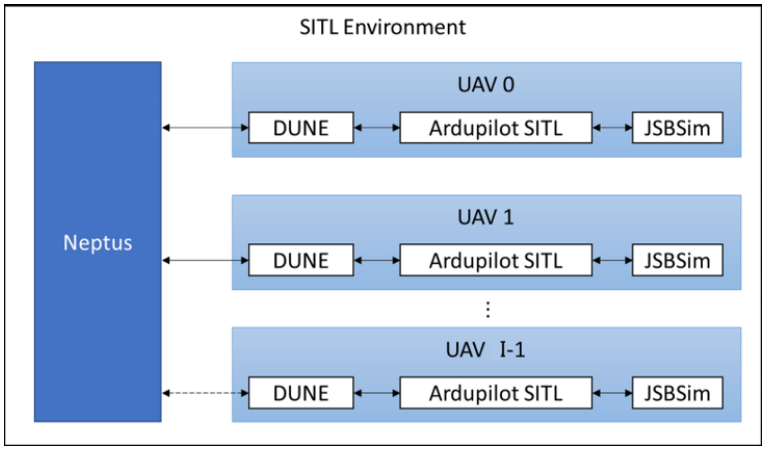


Figure 16 – Software-In-The-Loop setup.



Figure 17 – X8 UAV. Source: NTNU

2.3.4 Mission Simulation Scenario and Parameters

In this section, the parameters that define the mission scenario are described.

Aircraft and Remote Sensing Platform

The X8 (Figure 17) is a battery powered small UAS which can fly for around 80 min with the automated remote sensing payload and proper battery. The radius of the remote sensor's range is equal to 200 m, which is half of the width of the sensor's footprint. This footprint was chosen assuming that a computer vision algorithm, such as the one described by (RODIN et al., 2018), can detect the target in images captured at 400 m of altitude by an infrared camera with 7.5 mm of lens focal length, 640×480 pixels of resolution and 17 μm of pixel size.

Search Domain

The reference search area used in this work is equivalent to the Search Effort (Z_a) of one X8 UAS, calculated by Equation (88). Considering the total endurance of 80 min, the On-Scene Endurance (T) is equal to 60 min (85% of the total endurance). The Search Facility Speed (V) is equal to the average airspeed of the aircraft, in this case 16 m/s. The Sweep Width (W) is equal to 400 m, which is the lateral length of the required sensor footprint. Therefore, the search area is equal to 23.04 km^2, which gives a length and width equal to 4.8 km as the area has a squared shape because the datum is a single point.

Cells Grid

The grid was built with a cell width of 100 m. Therefore, the 23.04 km^2 of search area were divided into 48×48 cells. A two-dimensional normal distribution curve was fitted to the 12×12 single point datum reference table provided by the IAMSAR manual (IAMSAR, 2007). The fitted curve of the probability (Figure 18) that gives the reward of each cell is given by:

$$\phi_{mn} = 0.002946 \exp\Big(- \Big(\frac{(m - 23.5)^2}{108.28} + \frac{(n - 23.5)^2}{108.28} \Big) \Big), \tag{18}$$

where m and n are the horizontal and vertical indexes of the cell, respectively.

MPC Parameters

The airspeed range was chosen to be between 12 to 22 m/s. The reason for this choice was to keep the airspeed around the cruise speed, so that the battery consumption did not reach values that were too high. The roll angle range was chosen to be between –45 and 45 degrees so that the aircraft performed smoother maneuvers but still with freedom. The safe distance between UAVs was chosen to be 100 m and a wind of 9.9 m/s pointing to 45 deg was considered in the flight dynamics simulation.

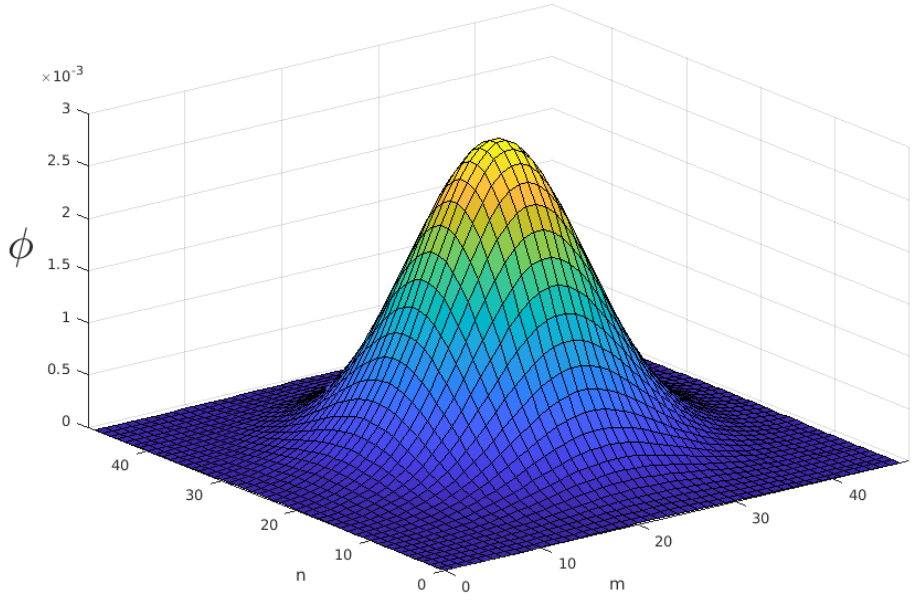


Figure 18 – Reward of cells.

A time horizon of 20 s and 20 horizon steps were the parameters chosen for the MPC problem. With, for example, a ground speed of 17 m/s, this means 340 m of straight distance, or a 180 deg turn. The weighting factor a was chosen to be 10,000 because the rewards are of a very low value (the sum of all cells rewards is equal to one). The weighting factors b and c were chosen to be 1, in order to avoid unnecessary aggressive maneuvers.

Regarding the PSO parameters for the optimization, a total of 384 particles was used and the algorithm runs 35 iterations with local and global coefficients of 1.

3- Georeferencing

3.1- Direct Georeferencing

This chapter presents the direct georeferencing method proposed in this thesis, which provides the localization of an image, acquired from optical sensors onboard an UAS, in a geographic frame of reference, completely dismissing the employment of ground control points (GCPs), which is the main contribution and novelty of this research and generated a paper published in the international journal SENSORS. Following, the definition of direct georeferencing, as well as its applications, will be presented. In addition, the most important mathematical tools used in the georeferencing, such as rotations and aircraft maneuver angles, will be formulated, in order to guarantee a self-contained material. The implementation steps will be described. Finally, will be presented two study cases, where the direct georeferencing algorithm was applied in situations with different physical characteristics, which shows the versatility of the method proposed in this thesis, demonstrating the importance of its scientific contribution. These researches generated two publications in international conferences (AIRPHARO and OCEANS).

Besides offering an useful tool in direct georeferencing of images for the UAS researche community, another intention of this research was to provide a complete and comprehensive overview of the entire method, presenting and explaining each concept and step in a friendly and detailed form, in order to supply the lack of such theoretical material in this theme in the scientific literature. The idea is to help researches that may face difficulties to deal with this process, which turns in a very complex problem, because its amount of details and operations.

3.1.1 Direct Georeferencing concept

Georeferencing is the process of assigning to an object or target a location in a geographic frame of Reference (KOBAYASHI, 2019). In this thesis, the objects of interest, or the targets, are initially available in an aerial image acquired by a monocular camera mounted on an UAS. The goal is, therefore, to calculate the location of the object in a geographic frame, such as the Universal Transverse Mercator frame or World Geodetic System 1984 frame. To perform these calculations, information about the camera sensor, mounting characteristics, and attitude and position of the UAS in the geographic frame is used, among other information. Considering that this process does not involve Ground Control Points, it is defined as Direct Georeferencing.

3.2- Bibliographic revision and the state of the art

In the literature, there are some works that perform direct georeferencing, giving an idea of the state of the art in this theme. The work (HEMERLY, 2014) has the merit of being one of the main references in the field, bringing a brief explanation of the direct georeferencing process, which is tested in a real UAS flight; the paper also mentions the camera calibration process. Leira et al. (LEIRA et al., 2015) implements a light-weight thermal camera for small fixed-wing UAS, with a georeferencing algorithm. The georeferencing process is explained and a real flight test is performed, where eighty images are georeferenced.

In (HELGESEN et al., 2019), the authors proposed a real time direct georeferencing system for maritime targets, employing a thermal camera mounted on an UAS. The system is tested for a static target and for a moving vessel. All the steps of the georeferencing process are discussed. The algorithm is later used in (HELGESEN et al., 2021) to calculate the target's position for further tracking in two of the four tracking solutions presented with the use of UAS. Real flight tests were taken to evaluate the solutions. In (XIANG; TIAN, 2011), a rapid aerial image georeferencing method for UAS for agricultural activities was implemented. Low cost Inertial Measurement Unit (IMU) sensors, Global Positioning

System (GPS) and sensor fusion techniques were employed, and GCPs were not used. The method obtains the georeferenced position by treating navigation data and the camera lens distortion model. Results were compared with sixteen positions given by GCP and the accuracy was considered satisfactory for agriculture applications.

In (BARBER et al., 2006), the authors implement a method for estimate the coordinates of a target from the image obtained by a camera, mounted in a small fixed-wing UAS, given the UAS position and altitude, and the camera posed angles. The georeferencing calculations are similar to those presented in our research, which are detailed and fully explained here. The paper also brings four techniques to reduce the estimated georeferenced position error.

3.3- Mathematical Fundamentals

In this section, the mathematical support used in the direct georeferencing equations are presented. These comprises concepts, such as rotation matrices, and orientation angles fundamentals.

3.3.1 Rotation Matrix

Figure 19 shows two different coordinates systems, represented by the orthogonal reference frames XYZ and $X'Y'Z'$. It also presents the angles between axis X' and each axis of the XYZ frame, which will be used in the coordinates transformations between the frames. The others angles that each axis forms with the axes of the other frame are not shown in Figure 19 for didactic purposes.

From Figure 19, it can be seen that the unit basis vectors of X' Y' and Z' axes ($\hat{i'}$, $\hat{j'}$ and $\hat{k'}$) projections in the XYZ frame are given by the following Equations (CURTIS, 2013):

$$\widehat{i'} = \cos A_{X'X}\hat{i} + \cos A_{X'Y}\hat{j} + \cos A_{X'Z}\hat{k}, \tag{19}$$

$$\widehat{j'} = \cos A_{Y'X}\hat{i} + \cos A_{Y'Y}\hat{j} + \cos A_{Y'Z}\hat{k}, \tag{20}$$

$$\widehat{k'} = \cos A_{Z'X}\hat{i} + \cos A_{Z'Y}\hat{j} + \cos A_{Z'Z}\hat{k}, \tag{21}$$

where $A_{\text{axis}\prime-\text{axis}}$ are the angles between the axis of the two frames, and $\cos\ A_{\text{axis}\prime-\text{axis}}$ denotes the direction cosines of the vector on the axis.

These equations can be written in the matrix form as:

$$\begin{bmatrix} \widehat{i'} \\ \widehat{j'} \\ \widehat{k'} \end{bmatrix} = \begin{bmatrix} \cos A_{X'X}\hat{i} + \cos A_{X'Y}\hat{j} + \cos A_{X'Z}\hat{k} \\ \cos A_{Y'X}\hat{i} + \cos A_{Y'Y}\hat{j} + \cos A_{Y'Z}\hat{k} \\ \cos A_{Z'X}\hat{i} + \cos A_{Z'Y}\hat{j} + \cos A_{Z'Z}\hat{k} \end{bmatrix}, \tag{22}$$

$$\begin{bmatrix} \cos A_{X'X}\hat{i} + \cos A_{X'Y}\hat{j} + \cos A_{X'Z}\hat{k} \\ \cos A_{Y'X}\hat{i} + \cos A_{Y'Y}\hat{j} + \cos A_{Y'Z}\hat{k} \\ \cos A_{Z'X}\hat{i} + \cos A_{Z'Y}\hat{j} + \cos A_{Z'Z}\hat{k} \end{bmatrix} = \begin{bmatrix} \cos A_{X'X} & \cos A_{X'Y} & \cos A_{X'Z} \\ \cos A_{Y'X} & \cos A_{Y'Y} & \cos A_{Y'Z} \\ \cos A_{Z'X} & \cos A_{Z'Y} & \cos A_{Z'Z} \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix}, \tag{23}$$

$$\begin{bmatrix} \cos A_{X'X} & \cos A_{X'Y} & \cos A_{X'Z} \\ \cos A_{Y'X} & \cos A_{Y'Y} & \cos A_{Y'Z} \\ \cos A_{Z'X} & \cos A_{Z'Y} & \cos A_{Z'Z} \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} = R_{XYZ\to X'Y'Z'} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix}, \tag{24}$$

where the matrix $R_{XYZ\to X'Y'Z'}$ is known as the Directional Cosine Matrix (DCM), or the rotation matrix, from the XYZ frame to the $X'Y'Z'$ frame, which is given by:

$$R_{XYZ\to X'Y'Z'} = \begin{bmatrix} \cos A_{X'X} & \cos A_{X'Y} & \cos A_{X'Z} \\ \cos A_{Y'X} & \cos A_{Y'Y} & \cos A_{Y'Z} \\ \cos A_{Z'X} & \cos A_{Z'Y} & \cos A_{Z'Z} \end{bmatrix}. \tag{25}$$

Now, let V be a vector in the frame XYZ:

$$V = a\hat{i} + b\hat{j} + c\hat{k} = [a\ b\ c] \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix}. \tag{26}$$

Therefore, the vector V', which is the representation of the vector V on the frame $X'Y'Z'$ can be obtained by:

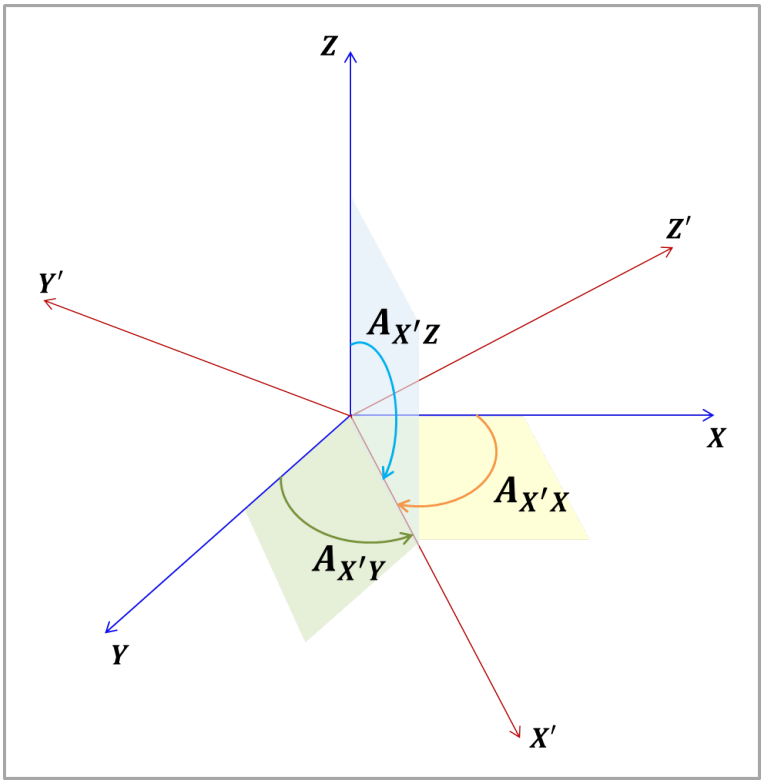


Figure 19 – Angles between $X'Y'Z'$ and XYZ axes.

$$V' = \begin{bmatrix} a'\hat{i} \\ b'\hat{j} \\ c'\hat{k} \end{bmatrix} = R_{XYZ \to X'Y'Z'} V = \begin{bmatrix} \cos A_{X'X} & \cos A_{X'Y} & \cos A_{X'Z} \\ \cos A_{Y'X} & \cos A_{Y'Y} & \cos A_{Y'Z} \\ \cos A_{Z'X} & \cos A_{Z'Y} & \cos A_{Z'Z} \end{bmatrix} \begin{bmatrix} a\hat{i} \\ b\hat{j} \\ c\hat{k} \end{bmatrix}. \tag{27}$$

or:

$$V' = R_{XYZ \to X'Y'Z'} V. \tag{28}$$

To perform the inverse transformation, that is, from the $X'Y'Z'$ frame to XYZ frame, the rotation matrix $R_{X'Y'Z' \to XYZ}$ is given by:

$$R_{X'Y'Z' \to XYZ} = \begin{bmatrix} \cos A_{XX'} & \cos A_{XY'} & \cos A_{XZ'} \\ \cos A_{YX'} & \cos A_{YY'} & \cos A_{YZ'} \\ \cos A_{ZX'} & \cos A_{ZY'} & \cos A_{ZZ'} \end{bmatrix}. \tag{29}$$

Considering the congruence of the angles between axis of the two frames, given by:

$$A_{X'X} = A_{XX'};\ A_{X'Y} = A_{YX'}\ ; A_{X'Z} = A_{ZX'}, \tag{30}$$

$$A_{Y'X} = A_{XY'};\ A_{Y'Y} = A_{YY'}\ ; A_{Y'Z} = A_{ZY'}, \tag{31}$$

$$A_{Z'X} = A_{XZ'};\ A_{Z'Y} = A_{YZ'}\ ; A_{Z'Z} = A_{ZZ'}, \tag{32}$$

one may conclude that:

$$R_{X'Y'Z'\to XYZ} = \begin{bmatrix} \cos A_{XX'} & \cos A_{XY'} & \cos A_{XZ'} \\ \cos A_{YX'} & \cos A_{YY'} & \cos A_{YZ'} \\ \cos A_{ZX'} & \cos A_{ZY'} & \cos A_{ZZ'} \end{bmatrix} \tag{33}$$

$$= \begin{bmatrix} \cos A_{X'X} & \cos A_{Y'X} & \cos A_{Z'X} \\ \cos A_{X'Y} & \cos A_{Y'Y} & \cos A_{Z'Y} \\ \cos A_{X'Z} & \cos A_{Y'Z} & \cos A_{Z'Z} \end{bmatrix} = R^{\intercal}_{XYZ\to X'Y'Z'}, \tag{34}$$

or:

$$R_{X'Y'Z'\to XYZ} = R^{\intercal}_{XYZ\to X'Y'Z'}. \tag{35}$$

So, given the rotation matrix from one frame, said XYZ, to another, said $X'Y'Z'$, in order to perform the inverse transformation, the new rotation matrix will be given by its transpose matrix:

$$V' = R_{XYZ\to X'Y'Z'}V \tag{36}$$

$$V = R_{X'Y'Z'\to XYZ}V' = R^{\intercal}_{XYZ\to X'Y'Z'}V'. \tag{37}$$

It can also be demonstrated that, in the specific case of rotation matrices, the following statement is true:

$$R^{\intercal}_{XYZ\to X'Y'Z'}R_{XYZ\to X'Y'Z'} = R^{\intercal}_{X'Y'Z'\to XYZ}R_{X'Y'Z'\to XYZ} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I, \tag{38}$$

where I canonically represents the identify matrix.

3.3.2 Euler Angles Sequence for Aerodynamics: Yaw, Pitch and Roll

There is a mathematical method to transform coordinates from one frame to another, that is, to perform the rotations of vectors between two frames. This technique is known as Euler angles sequence, and consists to choose a sequence of three angles to project one frame into another. There are many possibilities for these sequences. In aerodynamics, a specific angles sequence is more convenient to use, because its relationship with the aircraft movements: yaw, pitch and roll.

The yaw movement is the turn to the left or right of the nose of the airplane, provoked by the action of the rudder. In other words, it is a rotation in the axis normal to the plane formed by the wings and the fuselage. The pitch movement consists of turning the nose up or down, and it is generated by the movement of the elevators, located in the tail of the airplane. This is equivalent to a rotation in an axis along the wings. Finally, the roll movement is the rotation along the airplane longitudinal axis, which is the axis along the fuselage. This maneuver is provided by the ailerons, placed at the rear of the wings (TALAY, 1975). Figure 20 shows the yaw, pitch and roll airplane maneuvers, with their respective axes.

As seen in Figure 20, the axes are canonically attributed, respectively, to Z axis, for the yaw, Y axis for the pitch and X axis for the roll movements. Its positive rotations are given by the right hand law, with the thumb aligned with the arrow of the axis.

Figure 21 shows the Euler angles sequence for aerodynamics (CURTIS, 2013). Figure 21a shows the first rotation around the Z axis, by the yaw angle ψ, which carries the frame XYZ to the frame $X_1Y_1Z_1$, where the axis Z is congruent to Z_1; Figure 21b presents the second rotation, around the Y axis, by the pitch angle θ, moving the frame $X_1Y_1Z_1$ to the frame $X_2Y_2Z_2$, where the axis Y_1 is congruent to Y_2; Figure 21c shows the third and last rotation, in the X axis, by the roll angle ϕ, rotating from the $X_2Y_2Z_2$ reference frame to the final $X\prime Y\prime Z\prime$ frame. The notations ψ, θ and ϕ for the Euler angles yaw, pitch and roll, respectively, are canonical in the literature and, therefore, are employed in this work.

The rotation matrix for the Euler sequence yaw, pitch and roll angles, R_{YPR}, is

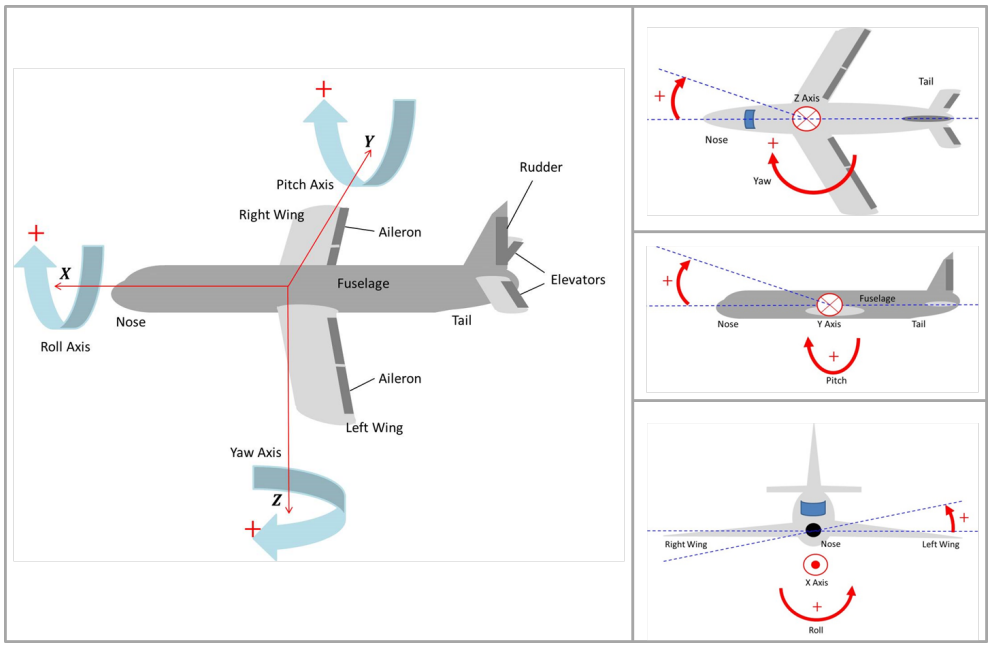


Figure 20 – Airplane yaw, pitch and roll maneuvers.

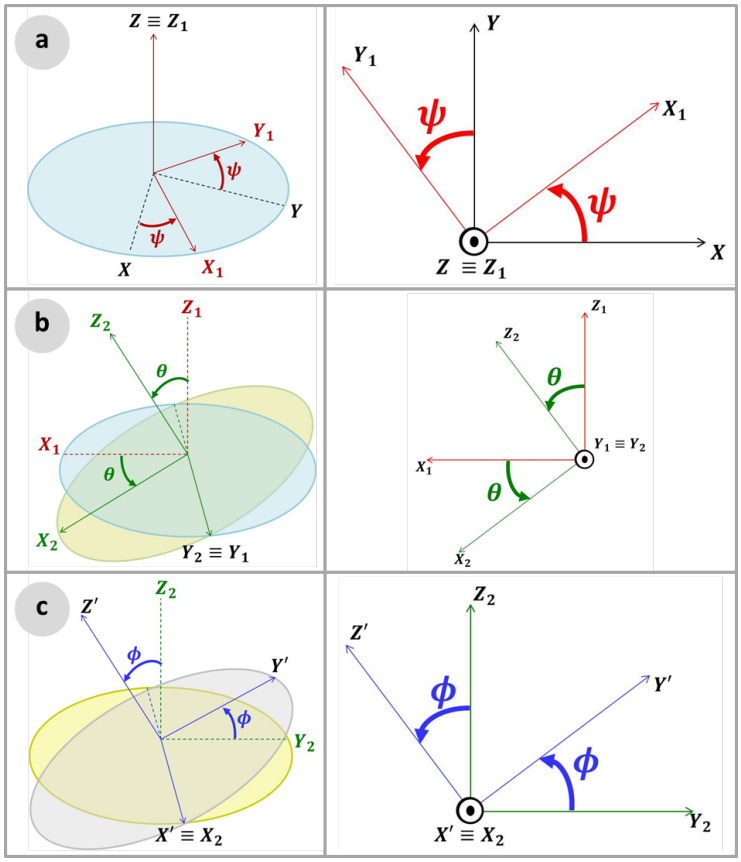


Figure 21 – Euler sequence: (a) Yaw; (b) Pitch; and (c) Roll.

given by (CURTIS, 2013):

$$R_{YPR} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \quad (39)$$

The Euler angles values are given by:

$$\tan\psi = \frac{\cos A_{XY'}}{\cos A_{XX'}} \quad (40)$$

$$\sin\theta = -\cos A_{XZ'} \quad (41)$$

$$\tan\phi = -\frac{\cos A_{YZ'}}{\cos A_{ZZ'}} \quad (42)$$

3.4- Direct georeferencing algorithm implementation

This section presents each step of the direct georeferencing process proposed in this thesis.

3.4.1 Image Frame to Camera Frame

Photographic cameras are devices that are able to record images from the three-dimensional world in bidimensional surfaces, or the image planes. Their construction is very simple, consisting of a dark box with a small aperture; the photons, reflected on the object surface in the three-dimensional world, will pass through the aperture and will form the image on the opposite surface of the dark box, where an optical sensor may be placed. Figure 22 shows this scheme.

The system can be mathematically modelled by introducing a coordinate reference in the camera aperture, called Camera Frame (C), where the object position is plotted in coordinates x_C, y_C, z_C, and another reference frame, the Image Frame (i), placed on the

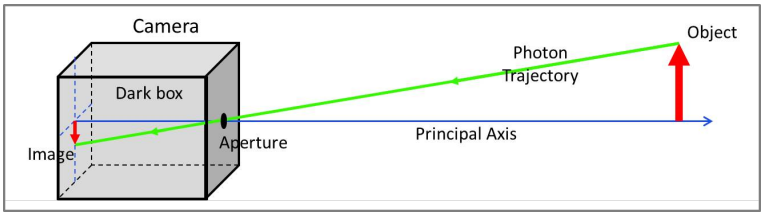


Figure 22 – Camera Model.

image plane, or focal plane, where the image position is plotted in the coordinates (x_i, y_i). Figure 23 shows the model.

This model is known as the Pinhole Camera Model. To make easier mathematical operations, a geometrically equivalent construction is obtained by mirroring the image plane, or the projection plane, to place it on the other side of the Camera Frame. This simple procedure allows to work only with positive values of the axes. Figure 24 demonstrates the scheme.

From Figure 24a the relation between the coordinates x_i, in the Image frame (i), and x_C, in the Camera frame (C), is obtained, where f is the focal distance.

$$\frac{x_i}{f} = \frac{x_C}{z_C} \therefore x_i = f\frac{x_C}{z_C}. \tag{43}$$

Furthermore, from Figure 24b:

$$\frac{y_i}{f} = \frac{y_C}{z_C} \therefore y_i = f\frac{y_C}{z_C}. \tag{44}$$

It is important to note that these equations are only valid for rectified images. If the image is not rectified, a rectifying process is needed prior to the use of these equations.

Figure 25 shows the same scheme in a more detailed perspective view. Here, the Image Frame is oriented in the two axis U and V:

$$u - c_x = x_i, \tag{45}$$

$$vs. - c_y = y_i, \tag{46}$$

where c_x and c_y are the coordinates of the principal point of the image.

Therefore, combining Equations (45) and (46) with Equations (43) and (44) leads to:

$$u = f\frac{x_C}{z_C} + c_x, \tag{47}$$

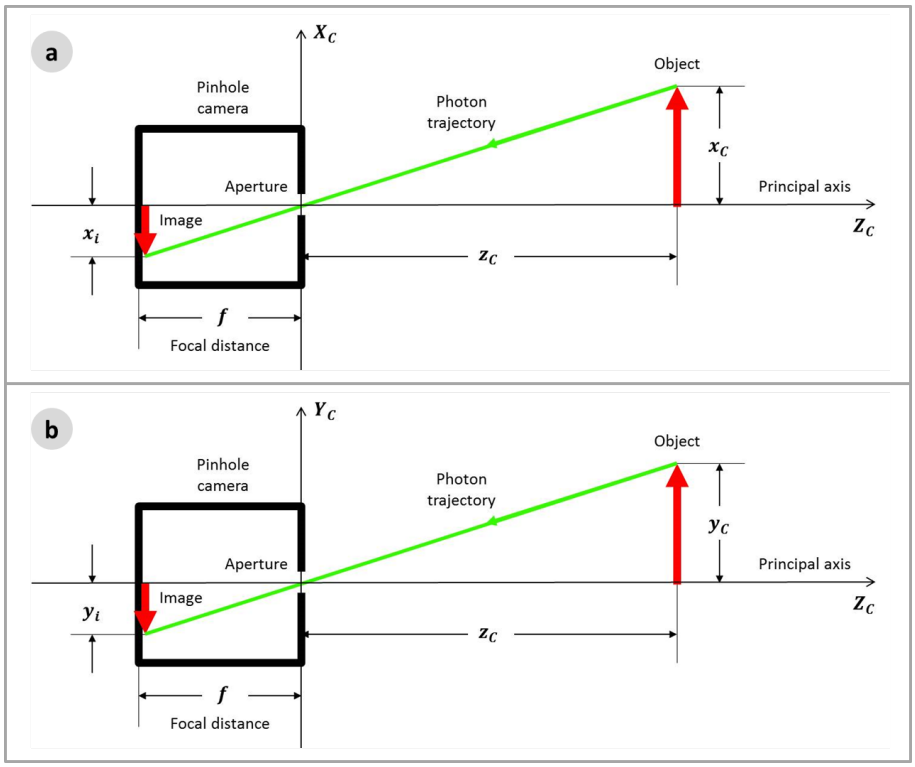


Figure 23 – Pinhole Camera Model: (a) X axis view; and (b) Y axis view.

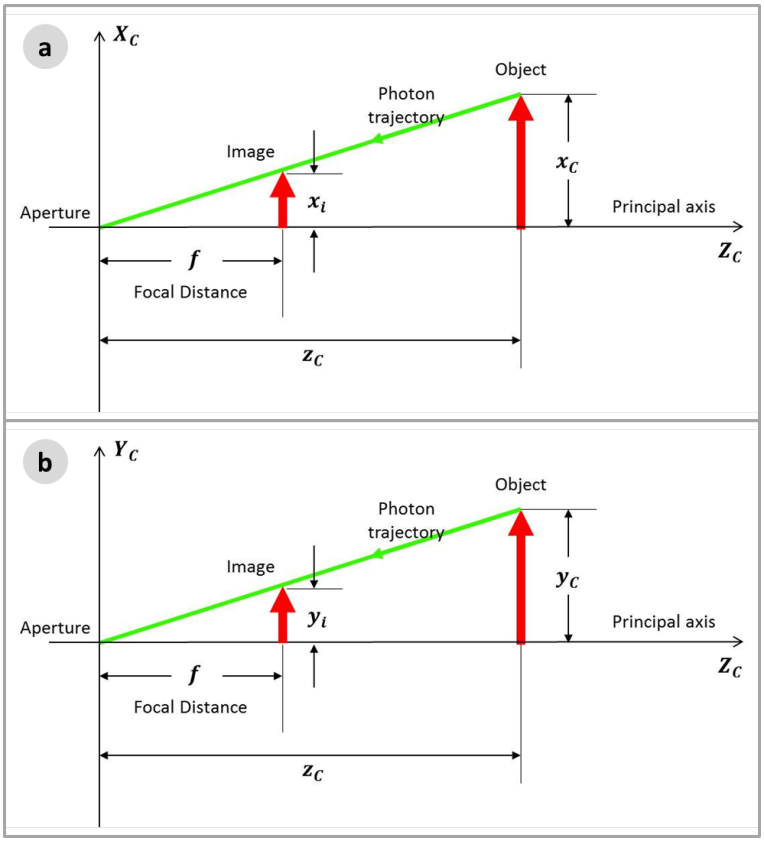


Figure 24 – Equivalent Camera Model: (a) X axis view; and (b) Y axis view.

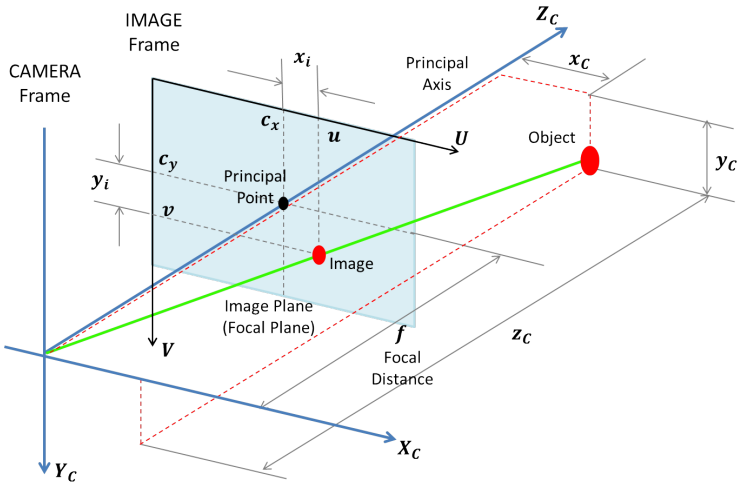


Figure 25 – Equivalent Camera Model, in a more detailed perspective view, with the Image Frame top left orientation in the two axis U and V.

$$vs. = f\frac{y_C}{z_C} + c_y. \tag{48}$$

In order to allow matrices operations, such as rotations and translations (the rigid transformations), which will be necessary in posteriors calculations, homogeneous coordinates will be employed. Therefore, Equations (47) and (48) take the following form:

$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}. \tag{49}$$

The coordinates u and v in the Image Frame i are conveniently used in pixels unit. The object coordinates in the Camera Frame (C) are usually given in meters. Therefore, naturally, the focal distance f and the coordinates of the principal point (c_x and c_y) should be given in pixels.

The focal distance (f) is usually given in millimeters. Therefore, to convert to pixels, multiply by the Image Size in pixels and divide by the Sensor Size in millimeters:

$$f[\text{px}] = f[\text{mm}]\frac{\text{Image Size[px]}}{\text{Sensor Size[mm]}} \tag{50}$$

In case of different pixel resolutions for x and for y dimensions, Equation (49) will

have the form:

$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}. \tag{51}$$

Therefore, f_x and f_y in pixels should be calculated based on the focal length (f in millimeters) image dimensions (ImageWidth and ImageHeight in pixels) and sensor dimensions (SensorWidth and SensorHeight in millimeters):

$$f_x = f\frac{\text{ImageWidth}}{\text{SensorWidth}}, \tag{52}$$

$$f_y = f\frac{\text{ImageHeight}}{\text{SensorHeight}}. \tag{53}$$

Equation (51) can also be written in the form:

$$z_C P_i = K P_C, \tag{54}$$

where $P_i = [u, v, 1]^\intercal$ is the vector of the object's coordinates in the Image Frame, $P_C = [x_C, y_C, z_C]^\intercal$ is the vector of the object's coordinates in the Camera Frame, and K is the intrinsic parameters camera matrix, also referred as the calibration matrix, as it can be obtained during the camera calibration process.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{55}$$

Equations (51) and (54) can be rewritten as:

$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}. \tag{56}$$

As the objective is to obtain the object's metric coordinates in the camera frame

from the object's pixel coordinates in the image frame, Equation (56) then becomes:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = K^{-1} z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{57}$$

Considering that z_C is unknown, in order to resolve the equation, it will be incorporated to P_C, generating the P'_C vector:

$$P'_C = \begin{bmatrix} \frac{1}{z_C} x_C \\ \frac{1}{z_C} y_C \\ \frac{1}{z_C} z_C \end{bmatrix} = \begin{bmatrix} x'_C \\ y'_C \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{58}$$

As will be further demonstrated, z_C will be easily determined at the end of the whole process, when the P'_C coordinates (x'_C and y'_C), will be determined (as well the P_C coordinates, x_C and y_C). Until there, however, the process will follow using P'_C.

3.4.2 Camera Frame to Gimbal Frame

The next step is the coordinates transformation from the Camera Frame (C) to the Gimbal Frame (G). The Gimbal Frame has its X axis, X_G, parallel to Z axis of the Camera Frame, Z_C; Y_G, the Y axis in Gimbal Frame, is parallel to the X axis of the Camera Frame, X_C; Z_G, the Z axis in the Gimbal Frame is parallel to the Y axis of the Camera Frame, Y_C. Figure 26 shows the Gimbal and the Camera frames.

The rotation matrix from the Camera Frame to the Gimbal Frame, $R_{C \to G}$, can be obtained throughout the projection of the Gimbal Frame in the Camera Frame, being given by:

$$R_{C \to G} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{59}$$

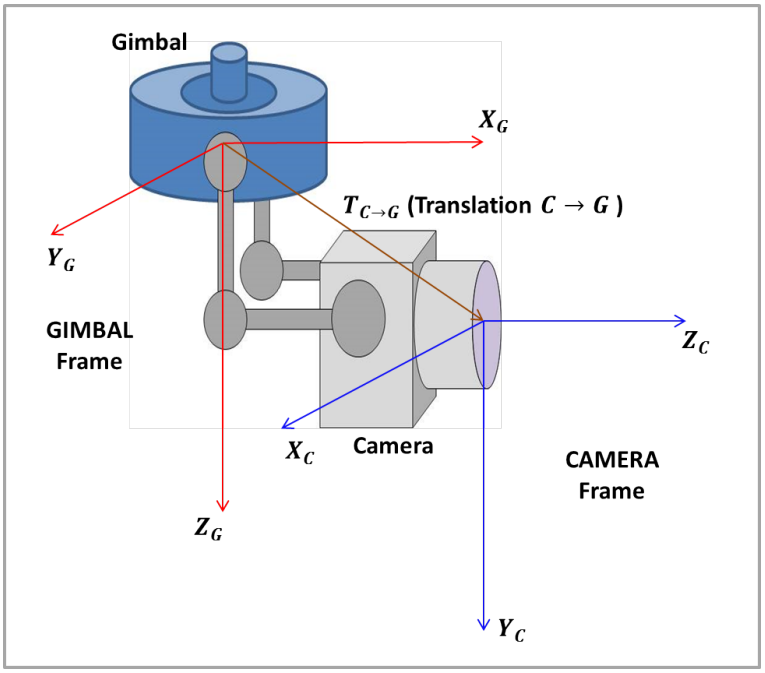


Figure 26 – Camera and Gimbal frames.

Therefore, the object's position in the Gimbal Frame prime, P'_G, is obtained by:

$$P'_G = R_{C\to G} P'_C, \tag{60}$$

or:

$$\begin{bmatrix} x'_G \\ y'_G \\ z'_G \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'_C \\ y'_C \\ z'_C \end{bmatrix}. \tag{61}$$

In cases where there is a difference in the position of the origin of the frames, as shown in Figure 26, a translation vector from the Camera Frame to the Gimbal Frame ($T_{C\to G}$) must be inserted. The object's position in the Gimbal Frame prime will be then given by:

$$P'_G = R_{C\to G} P'_C + T_{C\to G}. \tag{62}$$

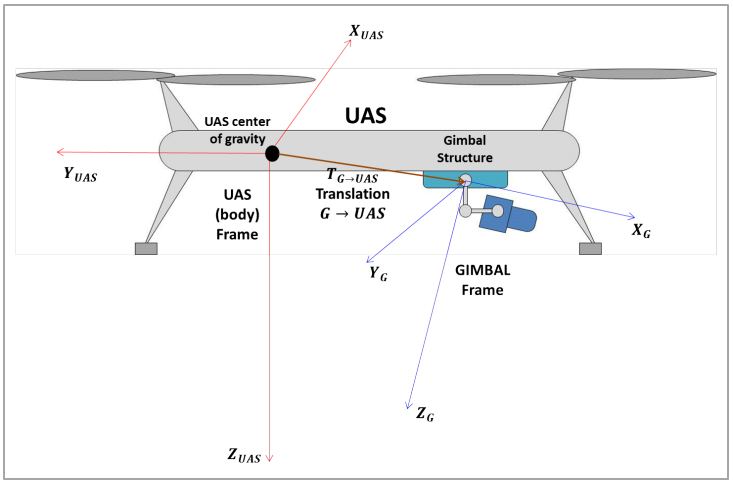


Figure 27 – UAS and Gimbal frames.

3.4.3 Gimbal Frame to UAS Frame

The next step is the object's coordinates transformation from Gimbal Frame to the UAS Frame, which is centered on the UAS center of gravity, as shown in Figure 27.

The transformation will require the Direction Cosine Matrix (DCM) from the Gimbal Frame to the UAS Frame, $R_{G\rightarrow UAS}$, given by the Euler angles yaw, pitch and roll. The operation is given by:

$$P'_{UAS} = R_{G\rightarrow UAS}P'_G + T_{G\rightarrow UAS}, \tag{63}$$

where $T_{G\rightarrow UAS}$ is the translation vector of the Gimbal Frame in the UAS Frame.

In a more complete form, Equation (63) may be written as:

$$\begin{aligned} P'_{UAS} &= R_{G\rightarrow UAS}P'_G + T_{G\rightarrow UAS} && (64)\\ &= R_{G\rightarrow UAS}(R_{C\rightarrow G}P'_C + T_{C\rightarrow G}) + T_{G\rightarrow UAS} && (65)\\ &= R_{G\rightarrow UAS}R_{C\rightarrow G}P'_C + R_{G\rightarrow UAS}T_{C\rightarrow G} + T_{G\rightarrow UAS} && (66) \end{aligned}$$

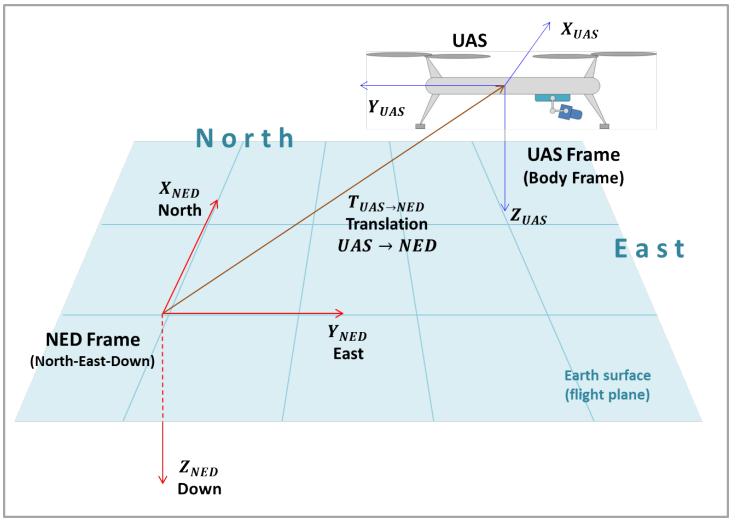


Figure 28 – NED and UAS frames.

3.4.4 UAS Frame to NED Frame

The following step is transform the object's coordinates from the UAS Frame to the NED Frame (North-East-Down Frame). Figure 28 presents the scheme, also considering a translation vector $T_{UAS \to NED}$, which is the position of the origin of the UAS frame in the NED frame, if there is a difference between the origins of the reference systems.

To perform the coordinates transformation, the DCM between the frames, $R_{UAS \to NED}$, must be employed, and so, the Euler angles yaw, pitch and roll must be known. The pixel position in the NED frame prime, P'_{NED}, is given by:

$$P'_{NED} = R_{UAS \to NED} P'_{UAS} + T_{UAS \to NED}. \tag{67}$$

Substituting Equation (64) in Equation (67), in order to obtain a more complete form:

$$\begin{aligned} P'_{NED} &= R_{UAS \to NED}(R_{G \to UAS} R_{C \to G} P'_C + R_{G \to UAS} T_{C \to G} + T_{G \to UAS}) + T_{UAS \to NED} \\ &= R_{UAS \to NED} R_{G \to UAS} R_{C \to G} P'_C + R_{UAS \to NED} R_{G \to UAS} T_{C \to G} \\ &\quad + R_{UAS \to NED} T_{G \to UAS} + T_{UAS \to NED} \end{aligned} \tag{68}$$

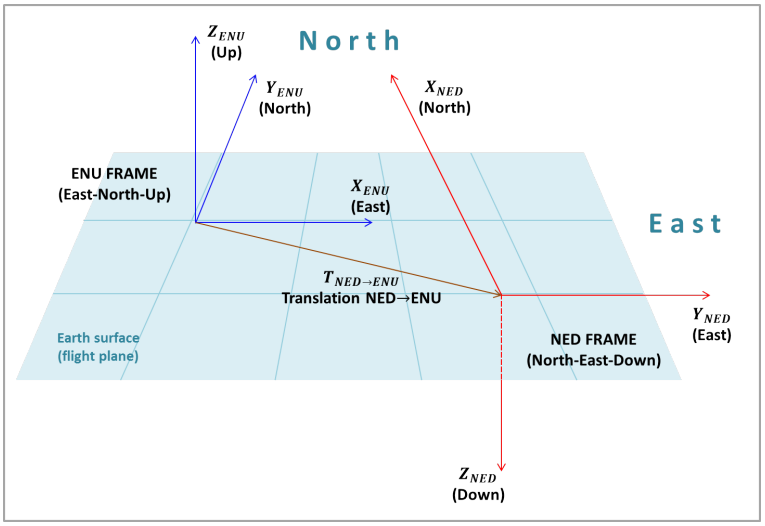


Figure 29 – ENU and NED frames.

3.4.5 NED Frame to ENU Frame

In many situations, it may be necessary to use the East-North-Up (ENU) reference system, as it is the reference frame employed in the Universal Transverse Mercator (UTM) coordinate system, which is directly associated with the World Geodetic System (WGS84) used by the Global Positioning Systems (GPS), with the well known latitude, longitude and height coordinates. Figure 29 presents the relation between the NED and the ENU frames.

From Figure 29, the rotation matrix from the NED frame to ENU frame, $R_{NED \to ENU}$, can be obtained:

$$R_{NED \to ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{69}$$

Thus, the primed position of the object in the ENU frame is given by:

$$P'_{ENU} = R_{NED \to ENU} P'_{NED} + T_{NED \to ENU}. \tag{70}$$

Considering the previous calculations, a general equation may be extracted, sub-

stituting Equation (68) in Equation (70):

$$\begin{aligned} P'_{ENU} &= R_{NED\to ENU}R_{UAS\to NED}R_{G\to UAS}R_{C\to G}P'_C \\ &+R_{NED\to ENU}R_{UAS\to NED}R_{G\to UAS}T_{C\to G} \\ &+R_{NED\to ENU}R_{UAS\to NED}T_{G\to UAS} \\ &+R_{NED\to ENU}T_{UAS\to NED} + T_{NED\to ENU} \end{aligned} \tag{71}$$

To simplify, it can be written as:

$$P'_{ENU} = R_{C\to ENU}P'_C + T_{C\to ENU}, \tag{72}$$

where the matrix $R_{C\to ENU}$ is the product of all rotations:

$$R_{C\to ENU} = R_{NED\to ENU}R_{UAS\to NED}R_{G\to UAS}R_{C\to G}, \tag{73}$$

and vector $T_{C\to ENU}$ is the sum of all translations until this stage, considering the rotations in each step:

$$\begin{aligned} T_{C\to ENU} &= R_{NED\to ENU}R_{UAS\to NED}R_{G\to UAS}T_{C\to G} \\ &+R_{NED\to ENU}R_{UAS\to NED}T_{G\to UAS} \\ &+R_{NED\to ENU}T_{UAS\to NED} \\ &+T_{NED\to ENU}. \end{aligned} \tag{74}$$

3.4.6 Determination of z_C

Equation (58) shows the relation between P_C and P'_C:

$$P'_C = \frac{1}{z_C}P_C. \tag{75}$$

Considering Equation (72), the object's position in the ENU frame is given by:

$$P_{ENU} = R_{C\to ENU}P_C + T_{C\to ENU}. \tag{76}$$

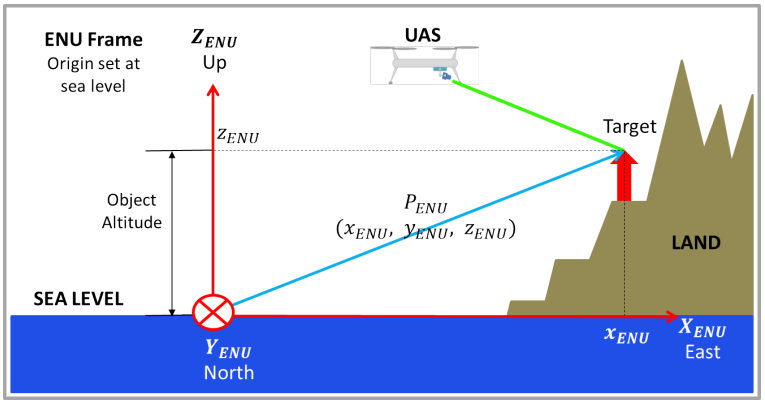


Figure 30 – Object's height in the ENU frame (z_{ENU} coordinate).

In order to determine z_C, it is necessary to solve the system formed by Equations (72) and (76):

$$\begin{cases} P'_{ENU} = R_{C\to ENU}P'_C + T_{C\to ENU} \\ P_{ENU} = R_{C\to ENU}P_C + T_{C\to ENU} \end{cases}. \tag{77}$$

The second equation of the system in Equation (77) can be rewritten in the form:

$$z_C P'_{ENU} - z_C T_{C\to ENU} = R_{C\to ENU}P_C. \tag{78}$$

The combination of Equations (78) and (77) leads to:

$$P_{ENU} = z_C P'_{ENU} - z_C T_{C\to ENU} + T_{C\to ENU}. \tag{79}$$

Equation (79) is a matrix equation, and has solution in the z coordinates, because the z_{ENU} coordinate, or the target object's z coordinate in the ENU Frame, is known: it is the object's altitude, considering that the origin of the ENU frame is set on sea level. In cases the ENU Frame is set on another level, the z coordinate will correspond to the vertical distance, as the height, from the origin of ENU frame to the object's level. Figure 30 shows the scheme.

The value of z_{ENU} must be known to perform the calculations, and it is very reasonable to assume this condition will be satisfied, because the region where the UAS is operating is known and so will be its altitude. In addition, digital elevation models can be used to estimate the altitude of the target.

Therefore, considering that z'_{ENU} and $z_{C\to ENU}$ had been determined in Equation

(72), z_C is finally given by:

$$z_C = \frac{z_{ENU} - z_{C \to ENU}}{z'_{ENU} - z_{C \to ENU}}. \tag{80}$$

Finally, the position of the object in the ENU frame can be obtained by using Equation (79)

$$P_{ENU} = z_C P'_{ENU} - z_C T_{C \to ENU} + T_{C \to ENU} \tag{81}$$

3.4.7 Camera Matrix

Very often, in the literature, all the calculations demonstrated before are summarized in an unique Equation (HELGESEN et al., 2021; HEMERLY, 2014; LEIRA et al., 2015; HELGESEN et al., 2019), in the form:

$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{ENU \to C} & T_{ENU \to C} \end{bmatrix} \begin{bmatrix} x_{ENU} \\ y_{ENU} \\ z_{ENU} \\ 1 \end{bmatrix} = M \begin{bmatrix} x_{ENU} \\ y_{ENU} \\ z_{ENU} \\ 1 \end{bmatrix}. \tag{82}$$

According to Hanning (HANNING, 2011), Equation (82) is called pinhole model equation, where M is the camera matrix, used to denote a projective mapping from world coordinates (x_{ENU}, y_{ENU}, z_{ENU}) to pixel coordinates (u, v).

Seeking the goal of this work, which is to bring a complete and detailed exposition about the entire direct georeferencing process, a brief explanation about the meaning of this expression must be taken here.

The main aspect in this equation is the use of homogeneous coordinates, from the projective geometry, in place of the Euclidean geometry coordinates. This representation is necessary to allow calculations in matrix form, using a single matrix (the camera matrix M) when mapping a three-dimensional image from the real world to a bi-dimensional camera image, and vice versa (HARTLEY; ZISSERMAN, 2004).

In the equation, K is the intrinsic parameter matrix (55), $R_{ENU \to C}$ is the 3 $\times$ 3 matrix representing the product of all rotations from the Camera frame to the ENU frame, and $T_{ENU \to C}$ is the 3 $\times$ 1 matrix, which is the sum of all rotated translations from the ENU

frame to the Camera frame.

According to Hanning (HANNING, 2011), in homogeneous coordinates, the extrinsic matrix ($\begin{bmatrix} R_{ENU \to C} & T_{ENU \to C} \end{bmatrix}$) is given by:

$$\begin{bmatrix} R_{ENU \to C} & T_{ENU \to C} \end{bmatrix} = \begin{bmatrix} R_{ENU \to C} & T_{ENU \to C} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (83)$$

The camera matrix M is given by:

$$M_{3\times4} = [K\ 0_{3\times1}] \begin{bmatrix} R_{ENU \to C} & T_{ENU \to C} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (84)$$

As it can be seen, the intrinsic matrix K has also to be used in homogeneous coordinates. The completion of the matrices with zeros and ones is done in a way that the result of the calculations is the same before and after the homogeneous coordinates transformation.

3.4.8 Lens Camera Distortion

In this research, the image used in the direct georeferencing process is assumed to be rectified. This means that no image distortion is present. Usually, manufacturers of UAS cameras provide the option to directly obtain the rectified image. However, as this subject may be an issue, it is briefly discussed in this section.

The geometry of the perspective or pinhole camera is simple since we assume the pinhole to be infinitely small. In reality, the light passes through a lens that complicates the camera model. The lens can distort the light rays projected onto the 2D sensor and

this geometric effect of image distortion is important to compensate for with regard to georeferencing. A camera with radial distortion is not well described by the pinhole model. Many wide-angle lenses have noticeable radial distortion which basically means that lines in the scene appear as curves in the image. There are two types of radial distortion: (1) barrel distortion and (2) pincushion distortion. There are also other forms of distortion effects, such as tangential distortion.

By only considering barrel distortion, the distorted coordinate on the sensor plane can be represented as (LI; LAVEST, 1996):

$$\begin{aligned} x_{distorted} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{distorted} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{85}$$

where $r = \sqrt{(x - c_x)^2 + (y - c_y)^2}$, c_x and c_y is the optical centre pixel on the sensor and k_1, k_2 and k_3 are the radial distortion coefficients, which can be obtained during the camera calibration process. Therefore, the original raw distorted image can be rectified.

3.5- Software Implementation

In this section, a pseudocode (Figure 31) to perform the direct georeferencing and the main functions used in the MATLAB and Python script are presented. The structures followed in each script are the same and follow the six steps discussed in Section 3.4. For each implementation script, the specific employed functions will be explained, as well as its special procedures.

3.5.1 MATLAB Implementation

The first step is the transformation from the Image to the Camera frame, and, in the beginning of all the calculations, the intrinsic parameters matrix data needs to be input. The intrinsic parameter matrix K is a 3-by-3 matrix, therefore in MATLAB has the following format: “K = [fx 0 cx; 0 fy cy; 0 0 1]”.

Algorithm 1 Direct Georeferencing

1: Input P_i
2: Define K
3: $P'_C \leftarrow inverse(K)P_i$
4: Define $R_{C\rightarrow G}$
5: Input $T_{C\rightarrow G}$
6: $P'_G \leftarrow R_{C\rightarrow G}P'_C + T_{C\rightarrow G}$
7: Input ψ_G, θ_G, ϕ_G
8: $R_{G\rightarrow UAS} \leftarrow transpose(DCM(\psi_G, \theta_G, \phi_G, "ZYX''))$
9: Input $T_{G\rightarrow UAS}$
10: $P'_{UAS} \leftarrow R_{G\rightarrow UAS}P'_G + T_{G\rightarrow UAS}$
11: Input $\psi_{UAS}, \theta_{UAS}, \phi_{UAS}$
12: $R_{G\rightarrow UAS} \leftarrow transpose(DCM(\psi_{UAS}, \theta_{UAS}, \phi_{UAS}, "ZYX''))$
13: Input $T_{UAS\rightarrow NED}$
14: $P'_{NED} \leftarrow R_{UAS\rightarrow NED}P'_{UAS} + T_{UAS\rightarrow NED}$
15: Define $R_{NED\rightarrow ENU}$
16: Input $T_{NED\rightarrow ENU}$
17: $P'_{ENU} \leftarrow R_{NED\rightarrow ENU}P'_{NED} + T_{NED\rightarrow ENU}$
18: $T_{C\rightarrow ENU} \leftarrow T_{NED\rightarrow ENU} + R_{NED\rightarrow ENU}T_{UAS\rightarrow NED} + R_{NED\rightarrow ENU}R_{UAS\rightarrow NED}T_{G\rightarrow UAS} + R_{NED\rightarrow ENU}R_{UAS\rightarrow NED}R_{G\rightarrow UAS}TC \rightarrow G$
19: $z_{C\rightarrow ENU} \leftarrow T_{C\rightarrow ENU}[2]$
20: $z'_{ENU} \leftarrow P'_{ENU}[2]$
21: Input z_{ENU}
22: $z_C = (z_{ENU} - z_{C\rightarrow ENU})/(z'_{ENU} - z_{C\rightarrow ENU})$
23: $P_{ENU} \leftarrow z_C P'_{ENU} - z_C T_{C\rightarrow ENU} + T_{C\rightarrow ENU}$
24: Output P_{ENU}

Figure 31 – Direct Georeferencing pseudocode

The pixel position coordinates in the Image Frame, u and v, are the next data required. The pixel position matrix in the Image Frame, P_i, is then determined. The MATLAB command "$K\backslash$" gives the product of the inverse matrix of the intrinsic matrix K with the following term.

The following operations are matrix multiplication for rotations and sum for translations, which in MATLAB is done in a straightforward way.

In the transformation from the Gimbal to the UAS frame, in order to obtain the DCM for the rotation, the Euler angles sequence, yaw, pitch and roll must be input. The MATLAB function "angle2dcm(angl1, angl2, angl3)" gives the DCM matrix, where the parameters "angl1", "angl2" and "angl3" are the aerodynamics Euler angles sequence, respectively, yaw, pitch and roll angles in this work. Here, it is important to note that the angles of the Euler sequence, yaw, pitch and roll, are commonly given in the most external reference frame, which in this case is the UAS Frame. Therefore, the MATLAB function will return the rotation from the UAS to the Gimbal frame. The rotation from the Gimbal to the UAS frame ($R_{G\rightarrow UAS}$) can be obtained by transposing the rotation matrix from the UAS to the Gimbal frame ($R_{UAS\rightarrow G}$), using the MATLAB function "transpose($R_{UAS\rightarrow G}$)". This same procedure needs to be applied when obtaining the rotation from the UAS to the NED frame.

The last step of the script consists of finding the target's position in the ENU frame. The coordinate z_{ENU}, which is equivalent to the distance from the target to the origin of the ENU frame in the z axis, needs to be input. It is also necessary to determine the total translation matrix, $T_{C \to ENU}$, which is given by the sum of all translations matrices, rotated with its related DCMs. Finally, to calculate z_C, z'_{ENU} and $z_{C \to ENU}$ are needed. In MATLAB, they are obtained by $P'_{ENU}(3)$ and $T_{C \to ENU}(3)$, respectively.

3.5.2 Python Implementation

In the Python implementation, NumPy (OLIPHANT, 2006) arrays are used for the matrices. As an example, to input a NumPy array for the intrinsic matrix, the following command is used: "K = numpy.array([[f_x, 0, c_x],[0, f_x, c_y],[0, 0, 1]])".

With the NumPy library, it is easy to get the inverse matrix of the intrinsic matrix by using the "numpy.linalg.inv(K)" command.

The matrix multiplication is done by using the operator "@" that was introduced in Python 3.5, for example: "C = A @ B".

The NavPy (NAVPY, n.d.) library is used to provide the DCM matrices by the function "navpy. angle2dcm" which inputs are the three angles of rotation (yaw, pitch and roll). The default rotation sequence is the "*ZYX*" sequence, which is the one used by this work. In addition, the documentation says that the DCM transforms a vector from the locally level coordinate frame (i.e., the NED frame) to the body frame. Therefore, in this work the transpose of the given DCM will be used, which transforms the vector from the body frame to the NED frame.

Finally, in the last step, where z_C is calculated, in order to access the third element of a NumPy array, the 2 index is used, for example: "z_T = T_T[2]".

3.6- Direct georeferencing applications

The algorithm developed in this thesis, and described above, was successfully applied in four use cases, of which two were included in a journal publication (CORREIA et al., 2022), and two were part of international conference publications: one in the AIRPHARO international conference (DE ALCANTARA ANDRADE et al., 2021) and one in the OCEANS conference (ANDRADE et al., 2022).

The applications included in the journal publication are detailed described in Chapter 4. The applications included in the two conference publications will be synthetically described next, and the direct georeferencing algorithm results in each case will be presented in Chapter 4. These applications confirm the accuracy of the method, as well as its versatility, showing that the algorithm may be used in many situations, with different characteristics, such as solar plants inspections and maritime traffic collision avoidance (the two cases particularly studied in these works), which demonstrates the importance of the contribution described in this chapter to the UAS scientific community.

3.6.1 Direct georeferencing of images of defective solar panels, taken by UAS cameras, in solar plants inspections

This research was partially funded by Equinor ASA, and produced with the partnership of the Norwegian Research Center (NORCE), as part of a bigger strategic project of research in solar-voltaic plants. The central idea is to use UASs in the inspection of solar panels, in solar-voltaic energy production plants, exploring the versatility, reliability and low costs of UASs.

In order to achieve the goals of that project, this research proposes an UAS inspection in a virtual reality solar plant. Figure 32 shows the basic scheme of the UAS path-planning control to fly in an inspection mission in a virtual reality solar plant. The solar plant was hosted in the Unreal Engine 4 (UE4), and the solar panels were modeled using Blender, an open-source 3D computer graphics software, with CAD Transform plugin, and Adobe Photoshop. The path-planning algorithm, for UAS flight controlling, was written

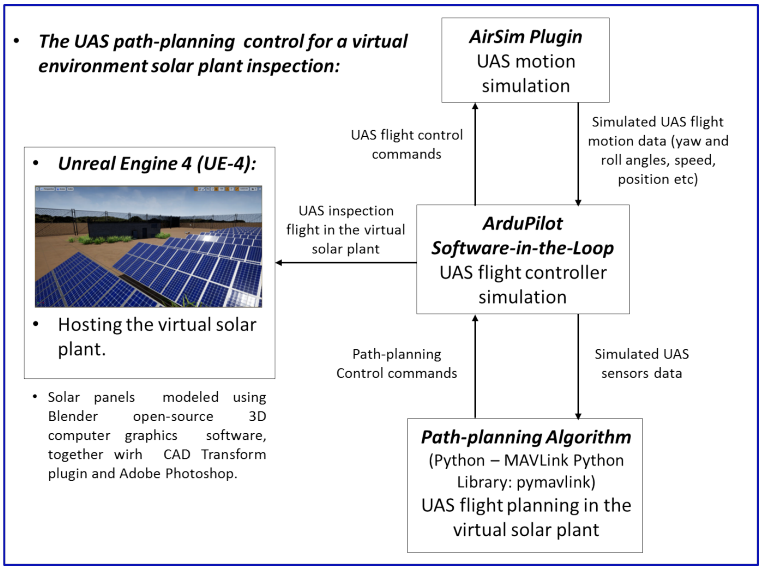


Figure 32 – The UAS path-planning autonomous mission control for a virtual reality solar plant inspection.

in Python, using the MAVLink Python Library (pymavlink), and hosted in the ArduPilot Software-in-the-Loop UAS flight controller, responsible for the UAS flight simulation control. The AirSim Plugin simulates the UAS motion, as well as velocity and flight attitude angles (yaw, pitch and roll).

Figure 33 shows the dynamics of images acquisition by the UAS RGB camera. Firstly, it is opportune to pose that the choice for a RGB device was due merely for the sake of the simplicity. In fact, any other optical imaging sensor, such as thermal cameras, can be employed with the algorithms used in this research. During the UAS autonomous mission in the unreal environment, AirSim's Python library is used to call the AirSim Image Application Programming Interface (API), to trigger the UAS on board RGB camera. An image is captured every time the UAS flies an user-defined distance. The API returns an 8-bit array, that is saved as a JPEG RGB image in a temporary folder, using the OpenCV library. Each image produced is analysed by a defect detection method, which was not the object of this research. In case of a defective panel is detected, the direct georeferencing algorithm is applied, in order to determine the geographic localization of the fault, making possible for the technical maintenance teams to work on it.

This explanation briefly exposes an overview of the research. The complete text is presented in the Appendix B of this thesis and, as pointed previously, the results of the direct georeferencing process involved in this research will be presented and discussed in Chapter 4.

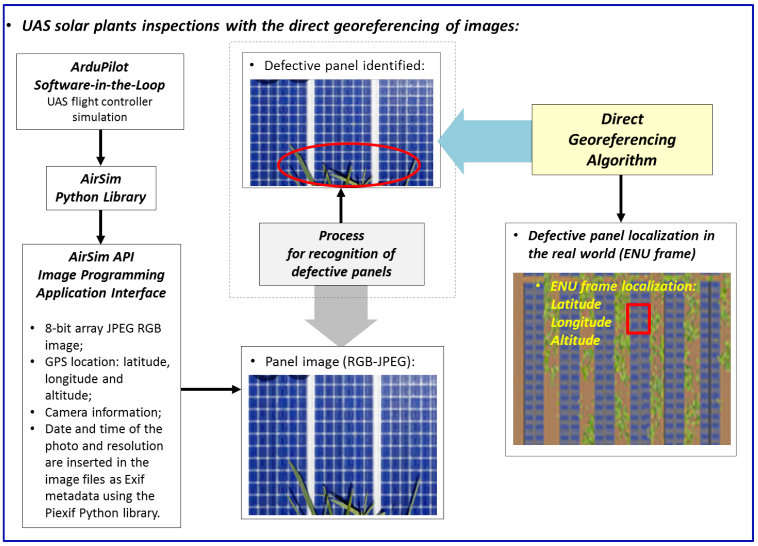


Figure 33 – The direct georeferencing of a defective solar panel image, taken from UAS RGB camera.

3.6.2 Direct georeferencing of images applied in a maritime collision avoidance system for small autonomous vessels

This research was produced with the partnership of the University of South-Eastern Norway (USN), in order to accomplish its undergraduate students participation in the annual competition between Norwegian universities, named AUTODRONE. Among many other challenges, the competitors teams must present solutions for maritime collision avoidance for autonomous vessels. This was the point that the present research fit in. The direct georeferencing algorithm developed and presented in this thesis, was successfully applied in the collision avoidance task proposed in the competition. This research generated an article, which was published in the OCEANS 2022 international conference (ANDRADE et al., 2022), whose full text is in the Appendix C.

For the collision avoidance task, it was employed the SeaDrone, an autonomous maritime vessel that belongs to the USN, presented in the Figure 34, which is equipped with many sensors, particularly, a RGB-D camera, magnetometers, GPS and IMU, whose data were used in the direct geofererencing algorithm, and are highlighted in the Figure 34.

The collision avoidance task in the AUTODRONE competition consisted in a mission where the USN SeaDrone must autonomously make the decision to change or

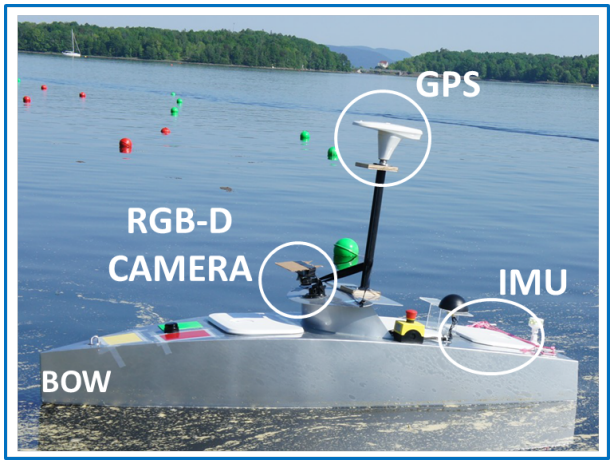


Figure 34 – The USN SeaDrone maritime autonomous vessel, showing the RGB-D camera, GPS and IMU on board sensors.

not its planned route, in order to avoid a possible collision risk with another vessel. Figure 35 schematically presents the collision avoidance task. SeaDrone must navigate in a straight line route in an area delimited between the red and green buoys, which red buoys must stand on the SeaDrone's port side, and the green buoys must be in its starboard side. The SeaDrone planned route is showed in the dotted line, marked with blue bold arrows. During the navigation, others vessels may cruise, perpendicularly to the SeaDrone planned route, which is showed in the dotted line with gray bold arrows, from the top to the bottom of the figure. The USN SeaDrone, then, must make the decision, in satisfactory timing, to give-way or to go on with the planned route.

Figure 36 presents a scheme of the algorithm used for collision avoidance autonomous decision making. Firstly, the on board RGB camera captures an image of the target, which is the vessel that may represent a possible threat to the USN SeaDrone navigation. A target detection algorithm, then, is used to identify the target. If the target is classified as "BOAT", it will be tracked. To calculate the target localization in the ENU frame, the direct georeferencing algorithm, developed in this thesis, is used, employing IMU, GPS, magnetometer and depth (from the RGB-D camera). In the next step, a Kalman filter is used to estimate the target position and velocity. Finally, the decision making algorithm takes the decision among two alternatives: "stand on", which means that the SeaDrone will not change the planned route; or "give away", implying in the alteration of the planned route, in order to avoid a possible risk of collision with the tracked target.

The algorithms showed to present satisfactory results in the AUTODRONE competition collision avoidance task. Specially, in what concerns to this thesis, the direct

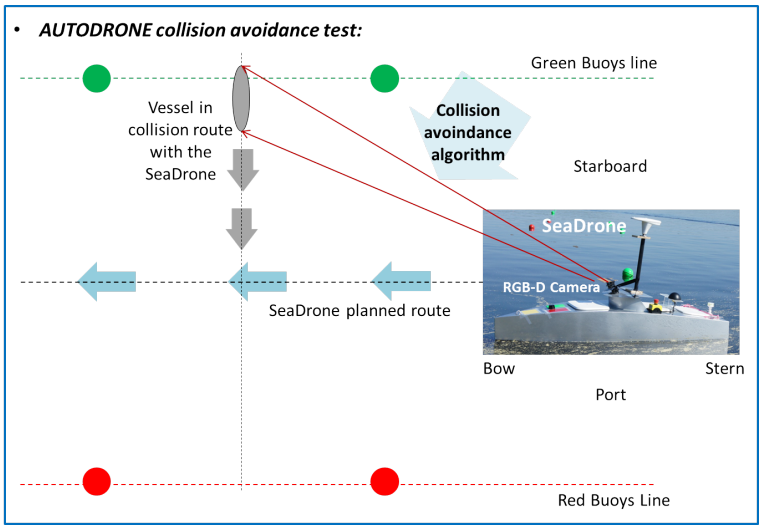


Figure 35 – The collision avoidance task in the AUTODRONE competition: the USN SeaDrone must make the decision between "stand on", which means to continue in the planned route, or "give away", changing the planned route.

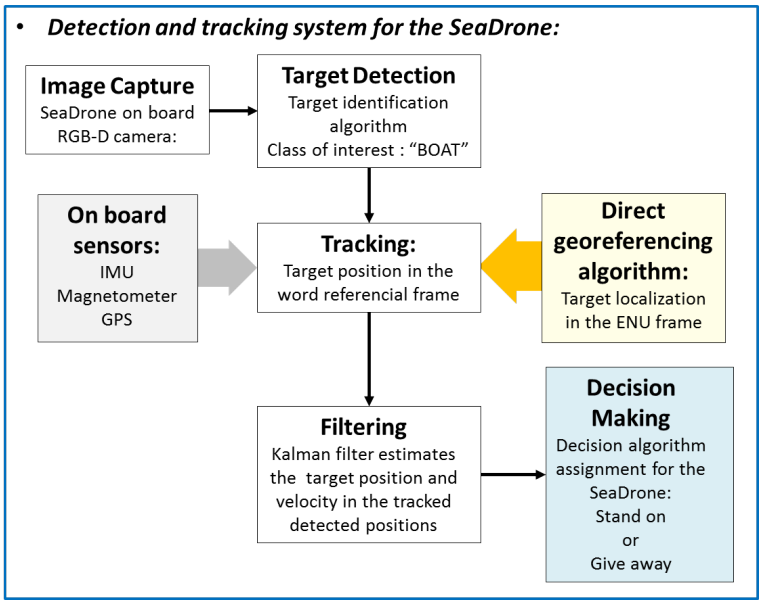


Figure 36 – The scheme of the collision avoidance algorithm, employed in the USN SeaDrone, in the AUTODRONE collision avoidance task.

georeferencing algorithm demonstrated accuracy, when applied to this situation. The results will be presented and commented in the Chapter 4.

4- Results

In this Chapter the results obtained in the researches performed in this thesis will be presented and discussed. Firstly, the results of the Search and Rescue algorithm, presented in Chapter 2 and published in the Journal Sensors article “Autonomous Unmanned Aerial Vehicles in Search and Rescue Missions Using Real-Time Cooperative Model Predictive Control” will be commented. Next, the results of the four direct georeferencing algorithm applications will be showed and discussed: in virtual reality and in UAS real flight test, published in the Journal Sensors article “Comprehensive Direct Georeferencing of Aerial Images for Unmanned Aerial Systems Applications”; direct georeferencing of defective solar panels, using images acquired from on board UAS RGB cameras, in inspection missions of a virtual reality solar-voltaic plant, published in the AIRPHARO international conference, in the article “Virtual Reality Simulation of Autonomous Solar Plants Inspections with Unmanned Aerial Systems”; and the direct georeferencing of vessels, in a collision avoidance system for small autonomous vessels, published in the OCEANS international conference, in the article “Detection and tracking of crossing vessels for small autonomous vessels equipped with stereo camera”.

4.1- Search and Rescue Results

In this section, the results of the search and rescue research, presented in Chapter 2, will be detailed commented.

4.1.1 Mission Scenario

Three operational profiles were evaluated for the mission scenario: employing only one UAS; employing two UASs; or employing three UASs. Five missions were executed

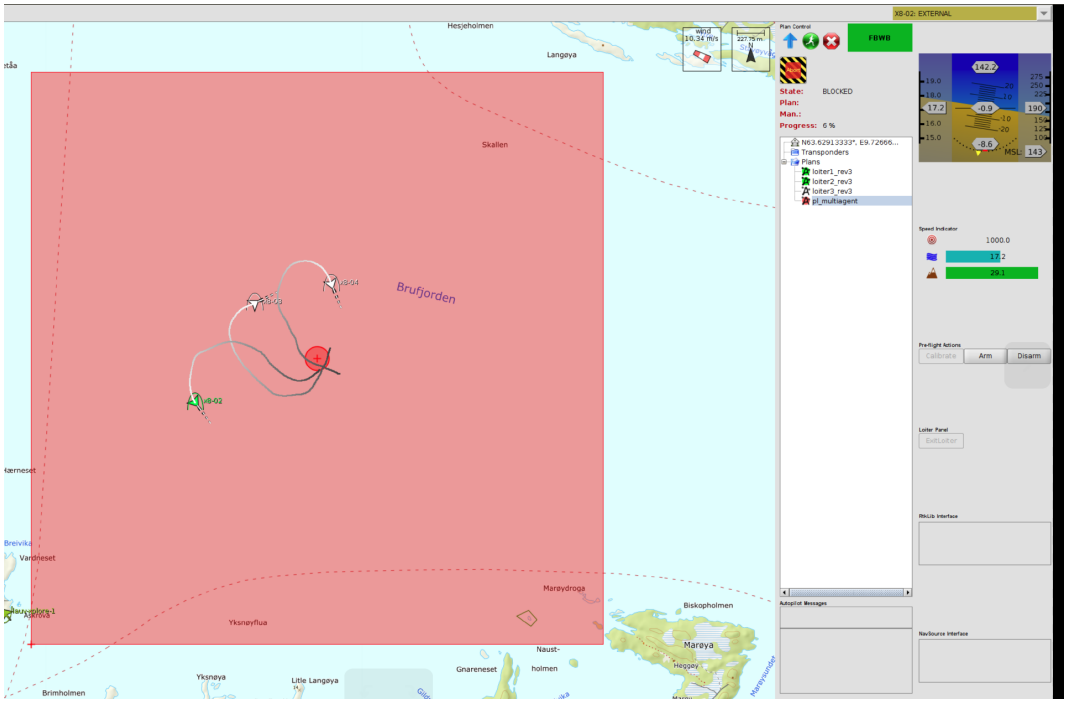


Figure 37 – Snapshot of a mission with three UASs being monitored with Neptus.

for each one of the profiles in order to obtain the average performance.

The reference search area was the Total Adjusted Search Area (A_t) for one UAS facility and Probability of Detection (POD) equal to 1.

The area was kept the same when employing two or three UASs in order to allow a fair performance comparison between the profiles. Figure 37 illustrates one mission with three UASs being monitored by the Command and Control software Neptus. The light red area is the search area and the dark red cross in the middle is the single point datum.

In all missions, the UASs departed from the same region (southeast of the search area as shown in Figure 38) where they were loitering and waiting for the mission start command. After receiving the command, the UASs departed to the search area and the mission time started to count from when the first UAS collected the first reward.

The IAMSAR manual describes the Probability of success (POS) as the probability of finding the search object with a particular search. For each sub-area searched, $POS = POD \times POC$. It is therefore the way to measure search effectiveness. As the Probability of detection (POD) is kept at 1, the POS is equal to the POC of the area, which in this work is the sum of all rewards collected by the group of UASs in the area.

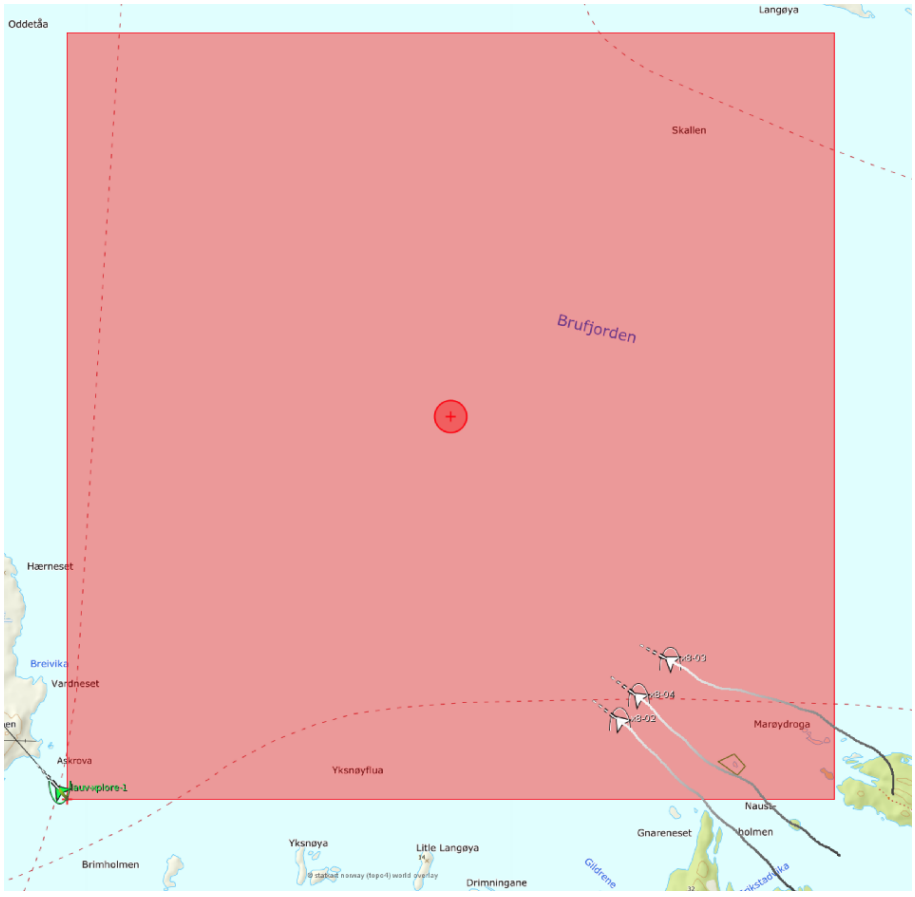


Figure 38 – Snapshot of the beginning of a mission.

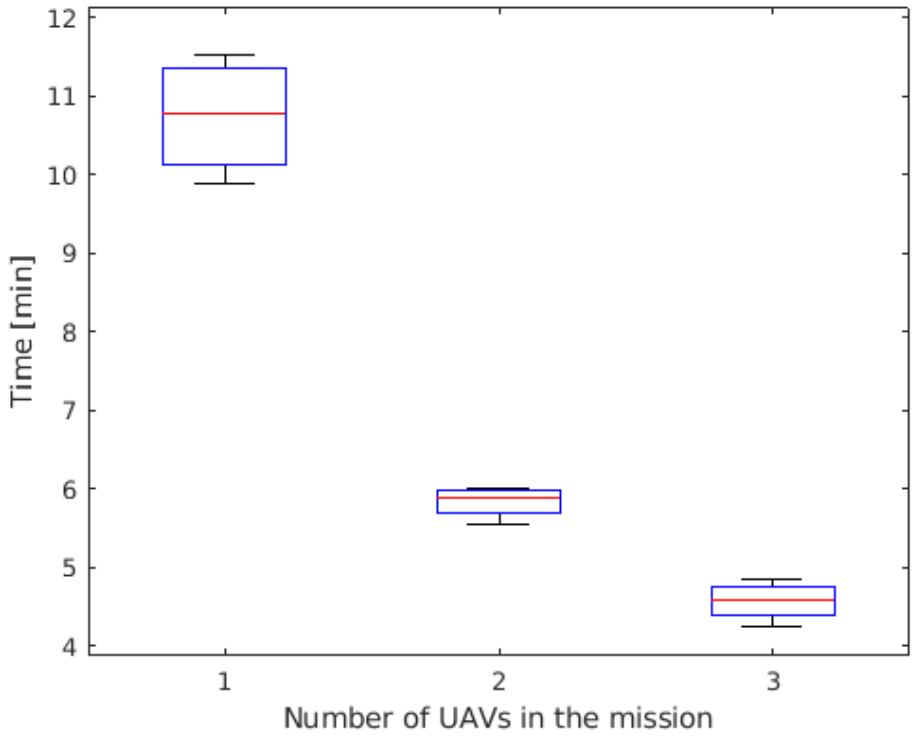


Figure 39 – Time to reach 50% of Probability of success (POS).

4.1.2 Results

The boxplot of the time to reach 50% of POS is shown in Figure 39 for the three operational profiles: employing one, two, or three UASs. It is possible to notice that the gain when a pair of UASs is used is very significant when compared to the single UAS profile, reaching 50% of POS 84% faster. When adding a third UAS, the gain was less significant: on average, the group reached 50% of POS 28% faster than when employing a pair of UASs. The decrease on the gain is probably due to the fact that the UASs are often flying over areas that have already been flown. A possible solution to avoid this situation is to reduce the width of the cells, increasing the resolution of the grid. Therefore, the UASs would better tune their maneuvers and still have the cells inside the UASs' sensor radius. However, this will increase the required computational power. This issue could be mitigated by optimizing the algorithm, for example.

Figure 40 shows the average POS during 20 min of mission. It is possible to notice that the results match the observed behavior when the missions were monitored. From

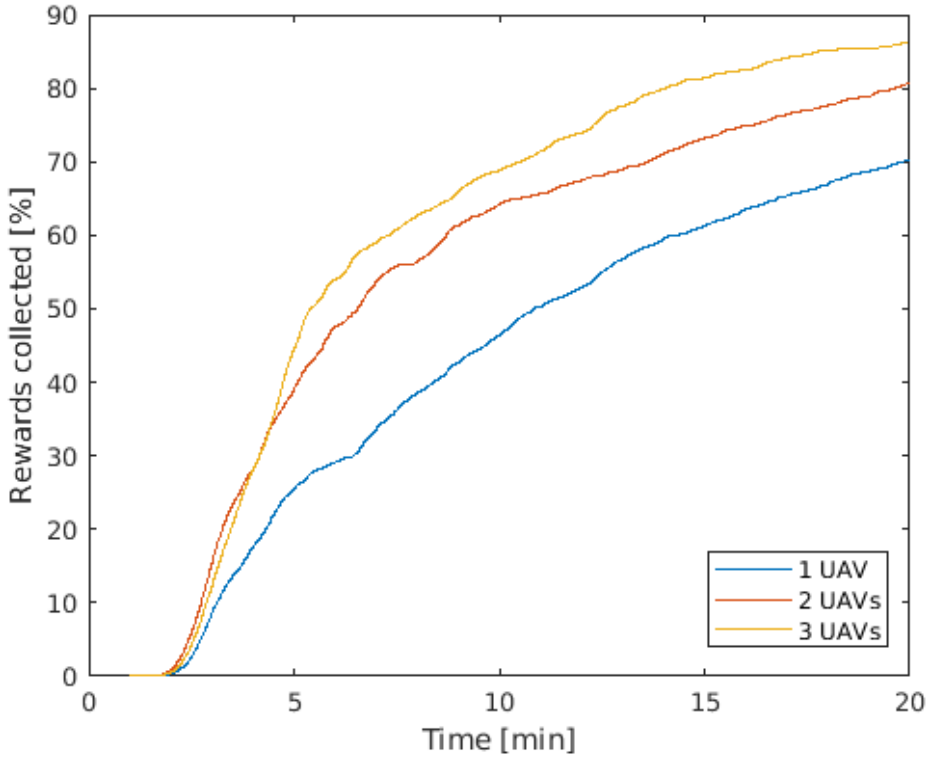


Figure 40 – Average Probability of Success in time.

the mission start to around 4 min, the UASs fly to the area close to the datum, where the reward (Probability of containment) is higher. When two UASs are employed, they fly parallel so that they cover more cells than when employing a single UAS. However, when three UASs are employed, even if they form a parallel path, they fly close to each other and, therefore, do not visit many more cells than the pair of UASs. This happens because, in case of the three UASs got far enough from each other to not visit the same cells, they would take a longer path to arrive at the central area (highest rewards), not being a cost beneficial solution.

After reaching the area close to the datum at around 4 min, the curve of rewards collection grows steadily and the difference between the three operational profiles is clear. When three UASs were employed, the group reached 50% of POS 2.35 times faster than the single UAS. The gain, however, reduces over time. For example, to reach 65% of POS, the group of three UASs did it 2.07 times faster than the single UAS. The reason for the decrease on the gain is that the more cells are already visited and rewards collected, the further the UASs have to fly to visit new cells and collect new rewards (that are also lower in value). Therefore, the closer it is to the end of the missions, the smaller is the difference

between the performance of the different operational profiles.

Figure 41 shows the boxplot of the POS after 20 min of mission for the three operational profiles. According to the IAMSAR Manual calculations, the single UAS is expected to reach 100% of POS in 60 min. It is possible to notice that the group of three UASs is able to reach close to 90% of POS in 20 min, showing that the improvement of adding extra UASs is approximately linear.

In order to evaluate the benefits of using the wind information in the kinematic model and test the system's robustness, extra simulations were performed for: a scenario where the wind was not considered in the MPC; and a scenario where the wind considered in the MPC was underestimated by 20%. Results are shown in Figure 42. It is possible to notice that the performance is superior when the wind information is used, collecting on average 4.4% more rewards in 20 min of mission time than when the wind is not considered in the model. The performance of the underestimated wind case was close to the ideal case, proving the system's robustness. It is important to notice that, despite the higher median of the underestimated wind case, the ideal case was able to reach higher values of rewards collected having both the upper quartile and maximum value higher than the underestimated wind case.

Finally, a pre-made path was created (Figure 43) in order to compare to the performance of the single UAS with the real-time MPC optimization. In this path, the UAS flies from the mission origin to the grid's midpoint and then flies a spiral path. This spiral path is close to the standard path suggested by the IAMSAR Manual.

In the spiral path, the lanes are equally spaced allowing the best coverage by the sensor's footprint. This would be the best possible simple standard path for the mission scenario being investigated. In addition, before starting the spiral path, the UAS is assumed to first fly to the center of the area.

Figure 44 shows that, for the same average airspeed of 15.5 m/s, the performance of the MPC path-planning was superior in the first 20 min of mission. In addition, in the spiral path, 50% of POS was reached in around 13 min, while it took less than 11 min when the MPC optimization was used.

In the spiral path, wind was not considered and the UAS keeps the ground speed constant, while, in the MPC path-planning, the UAS optimizes its speed to reach higher coverage, for example reducing the airspeed to achieve a steeper turn when needed. Another advantage, which is perhaps the most important, is that the MPC solution has

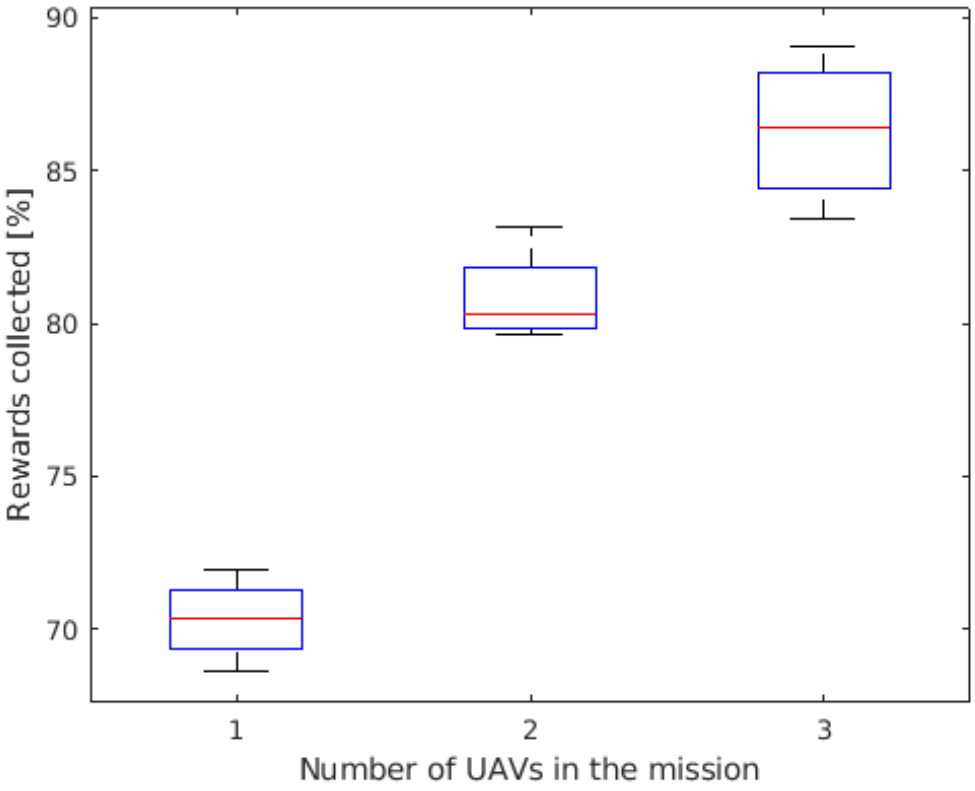


Figure 41 – Probability of success (POS) in 20 min of mission.

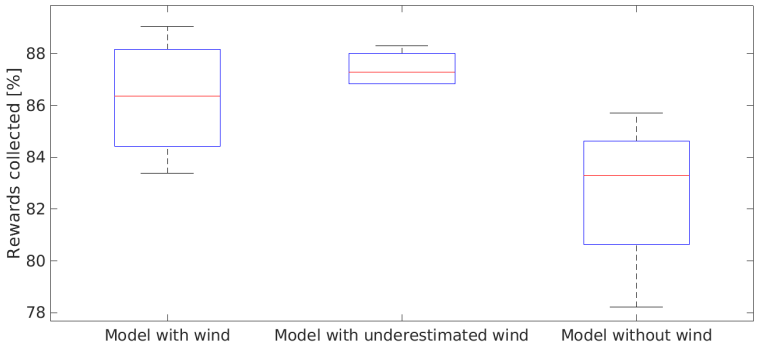


Figure 42 – Probability of success (POS) in 20 min of mission.

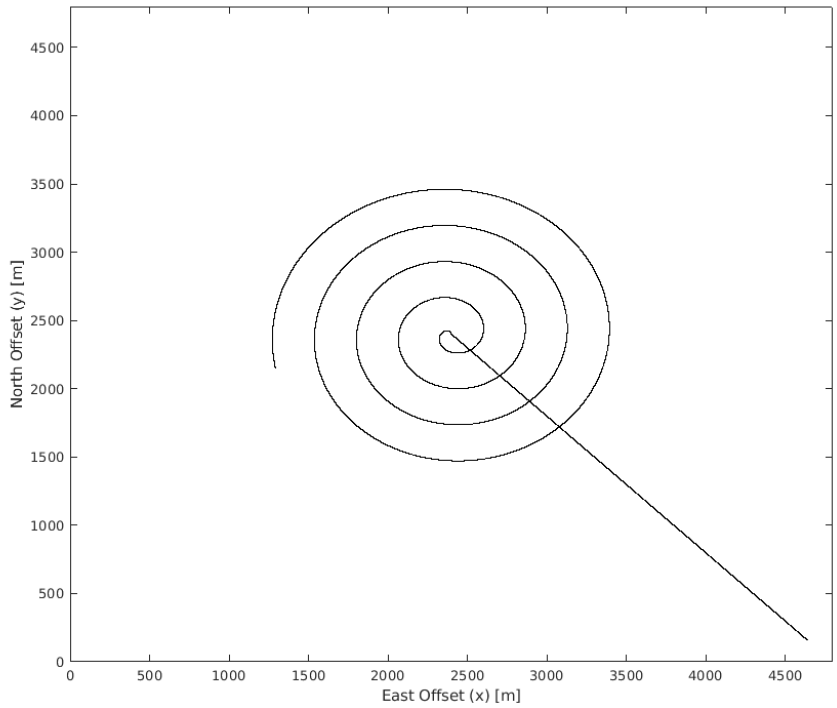


Figure 43 – Spiral path.

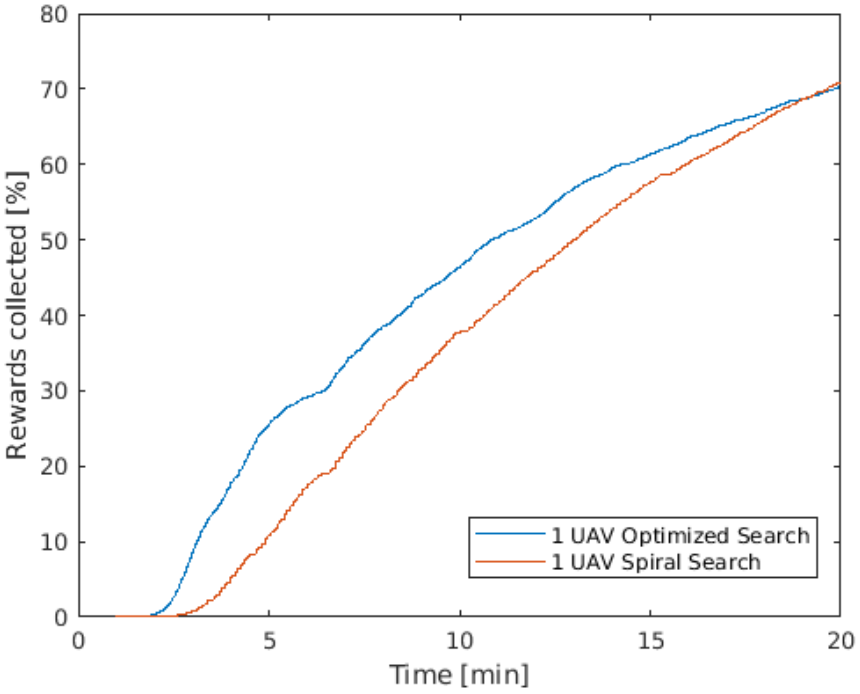


Figure 44 – Probability of Success.

the capability to deal with dynamic changes on the environment and mission parameters during the mission, as it is a real-time optimization. These changes can be wind variations, updated search and rescue reports or even the lost of one UAS in the middle of the mission due to technical problems or the addition of extra UASs that arrived later when the mission had already started.

4.2- Direct Georeferencing Results

As pointed before, the direct georeferencing method proposed in this thesis was applied in four applications, which had generated three articles. Just as an organization matter, results will be presented in the same order mentioned in the beginning of this Chapter, considering the articles where they were published.

4.2.1 A simulation environment example with UE4 and AirSim and a practical example with ROS in an UAS real flight simulation

Simulation environment example with UE4 and AirSim

In this case, a simulation environment was created in the Unreal Engine 4 (UE4). The AirSim plugin was used to simulate the UAS camera and flight dynamics. The UAS model in the simulation environment can be seen in Figure 45.

The virtual camera was placed 20 cm down to the drone and 30 cm to the front as illustrated with the example of Figure 46.

A target (white cross) was placed on the ground, in front of the door of a power house (Figure 47). To simulate the Universal Transverse Mercator (UTM) frame, a random location in the simulation environment was chosen to be the origin of the UTM frame. The target's position in virtual UTM coordinates was then 8.5 m to the east, -8.0 m to the north and 0 m of height, as it is positioned on the ground.



Figure 45 – UAS in simulation environment.

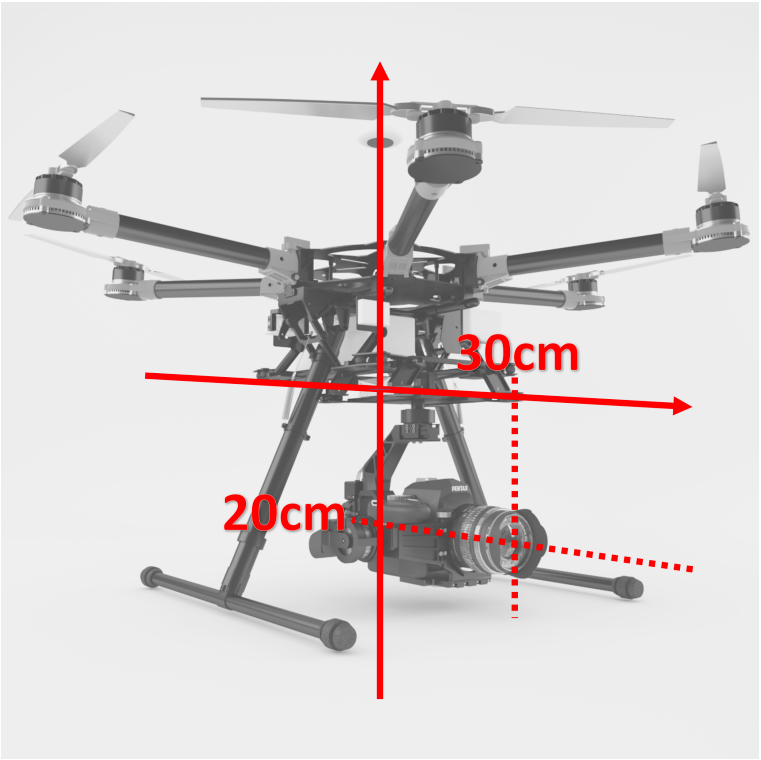


Figure 46 – Camera placement on UAS in the simulation application.

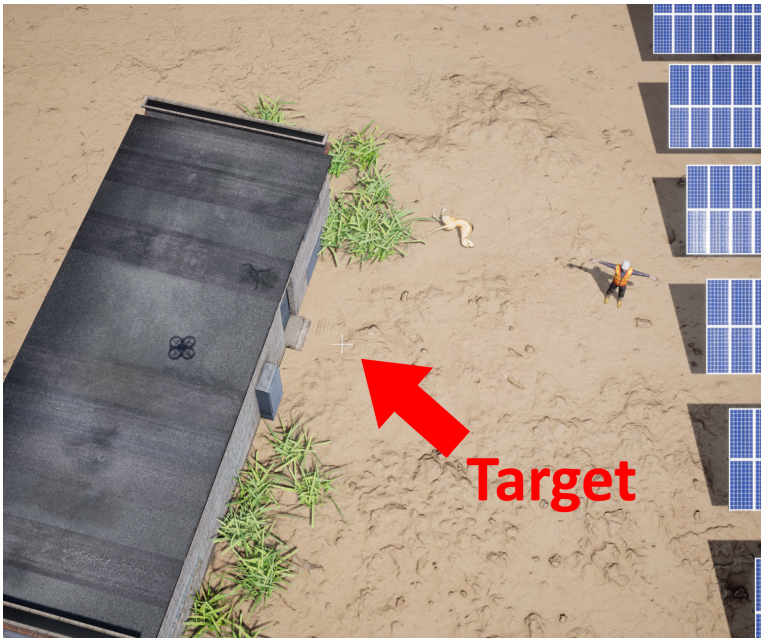


Figure 47 – Picture acquired in the simulation environment.

The gimbal's attitude was configured to have a yaw (ψ_G) of -90 degrees ($-\pi/2$ radians) and a pitch (θ_G) of -60 degrees ($-\pi/3$ radians) with respect to the world. The UAS was moved to a place where it could have a good view of the target and the surroundings to take the picture. The position of the drone ($T_{NED \to ENU}$ in meters) was given by the simulated sensors.

The camera was modeled based on the FLIR Chameleon 3 (FLIR, n.d.) camera sensor and a lens with 12.5 mm of focal distance. The camera has a resolution of 2448 $\times$ 2048 pixels and a sensor of 8.6 mm of width (FLIR, n.d.).

Additionally to the pixels resolution, the AirSim plugin models the camera by the horizontal Field of View in degrees. The following equation was used to calculate the horizontal field of view ($FOV_{\text{horizontal}}$ in degrees) from the horizontal sensor size (SensorWidth in millimeters) and the focal length (f in millimeters):

$$FOV_{\text{horizontal}} = 2 \arctan\left(\frac{\text{SensorWidth}}{2f}\right) \tag{86}$$

Therefore, for 8.6 mm of sensor width and 12.5 mm of focal length, the horizontal field of view is equal to 37.97 degrees.

The target (white cross) in Figure 47 is located on $[u, v] = [1095, 1099]$ pixels. The values of all parameters can be seen in Figure 48.

The calculated georeferenced position of the target in the ENU frame (P_{ENU}) was

Parameter	Value	Note
P_i	$[1095, 1099]$	
K	$\begin{bmatrix} 3558.1395 & 0 & 1224 \\ 0 & 3558.1395 & 1024 \\ 0 & 0 & 1 \end{bmatrix}$	
$T_{C \to G}$	$[0,0,0]$	The camera and the gimbal frames have the same origin.
$[\psi_G, \theta_g, \phi_G]$	$[-\pi/2, -\pi/3, 0]$	
$T_{G \to UAS}$	$[0.3, 0, 0.2]$	
$[\psi_{UAS}, \theta_{UAS}, \phi_{UAS}]$	$[0,0,0]$	As the gimbal's attitude is with respect to the world, the UAS attitude is not considered.
$T_{UAS \to NED}$	$[0,0,0]$	The UAS and NED frames have the same origin.
$T_{NED \to ENU}$	$[31.72212, -6.55099, 42.44889]$	
z_{ENU}	0	

Figure 48 – Parameters used for the direct georeferencing of the aerial imaged acquired in the simulation environment.

$[8.50283, -7.99841]$ m. This gives 100% of accuracy if rounded to the second decimal digit, proving that the georeferencing method is correct.

Practical example with ROS, in an UAS real flight test

For the practical experiment, a ZED 2 camera (LABS, n.d.) was mounted on the UAS pointing straight down without a gimbal. The Robot Operating System (ROS) was used to capture and save the data.

The camera ROS wrapper provides the topics of "image_rect_color", where it is possible to obtain the rectified image; and the "camera_info", where the camera intrinsic matrix K for the rectified image can be obtained.

The camera system is equipped with an IMU and a Magnetometer which are used in this research to calculate the UAS' attitude. These sensors are mounted close to the camera sensor. The distance between the attitude sensors and the camera ($T_{G \to UAS}$) and the orientation between the camera and the attitude sensors ($[\psi_G, \theta_G, \phi_G]$) are given by

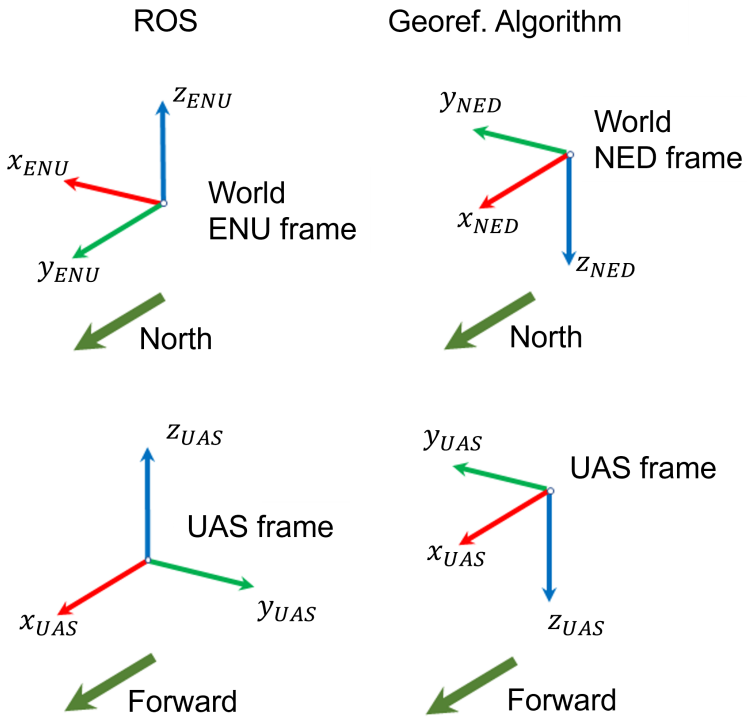


Figure 49 – ROS and georeferencing algorithm frames.

the manufacturer in the ROS wrapper through the topic “left_cam_imu_transform”.

The attitude is given in quaternions format in the topic “imu/data”. As Euler angles are used in this research, a conversion from quaternions to angles is needed. However, in order to proceed with the georeferencing calculations, additional conversions are required as the frame in ROS have different orientation than the UAS frame used by this research (Figure 49), which is the same used by flight control units. Therefore, as the y and z axes in ROS’ frame are pointing to the opposite direction of the axes y and z in this work’s UAS frame, the pitch angle obtained from the “imu/data” ROS topic needs to be inverted; and the yaw angle needs to be inverted and added 90 degrees ($\pi/2$ radians).

The test setup was to take a picture (Figure 50) of the target object, which is a table of 80 $\times$ 60 $\times$ 85 cm.

The parameters considered for the direct georeferencing can be seen in Figure 51. The translation from camera to gimbal frames ($T_{C\rightarrow G}$) is zero because there is no gimbal installed in this setup. To keep the same implementation steps that were described in the previous sections, as no gimbal was installed, the gimbal angles ($[\psi_G, \theta_G, \phi_G]$) considered in the calculations are actually the angles of the camera attitude with respect to the IMU (UAS frame). The same is valid for the translation ($T_{G\rightarrow UAS}$). The attitude of the IMU



Figure 50 – Aerial image taken by camera mounted on UAS.

Parameter	Value	Note
K	$\begin{bmatrix} 1055.33422 & 0 & 1055.33423 \\ 0 & 990.06824 & 544.24640 \\ 0 & 0 & 1 \end{bmatrix}$	
$T_{C \to G}$	$[0, 0, 0]$	
$[\psi_G, \theta_G, \phi_G]$	$[0.00176, 0.00116, 0.00138]$	
$T_{G \to UAS}$	$[-0.002, 0.023, 0.002]$	
$[\psi_{UAS}, \theta_{UAS}, \phi_{UAS}]$	$[6.04630, -1.46643, -0.10614]$	
$T_{UAS \to NED}$	$[0, 0, 0]$	The UAS and NED frames have the same origin.
$T_{NED \to ENU}$	$[0, 0, 8.88]$	

Figure 51 – Parameters used for the direct georeferencing of the aerial imaged acquired in a real flight.

(UAS frame) is obtained from converting the quaternions from the ROS topic “imu/data” to Euler angles and making the required conversions from the frame used in ROS to the frame used in this work. The NED frame’s origin is considered at the same place as the IMU (UAS frame) and the ENU horizontal origin is placed on the UAS. Finally, the vertical ENU origin was placed on the ground, therefore, the height of the UAS ($z_{NED \to ENU}$) was obtained from its relative height (height from take-off).

The strategy used to evaluate the method was to georeference the four corners of the table’s top shelf. As it can be seen in Figure 52, the blue squares have the corners’ pixel coordinates and the green squares have the calculated georeferenced coordinates in the ENU frame. It is important to notice that in this case, as the corners of the top shelf are being georeferenced, the height of the object (z_{ENU}) is of 0.85 m.

The calculated lengths from corner to corner can be seen in Table 1. This was

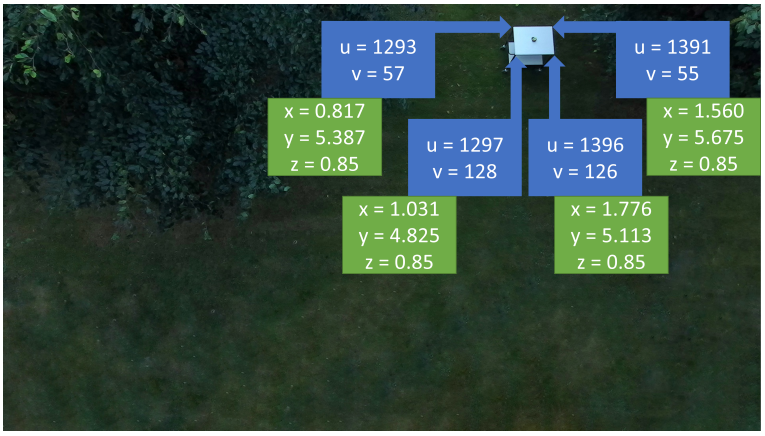


Figure 52 – Pixel and georeferenced positions of table's top shelf's four corners.

Table 1 – Calculated length between shelf corners.

	Length [m]
top left to top right	0.80
top left to bottom left	0.60
top right to bottom right	0.60
bottom left to bottom right	0.80

the strategy chosen to evaluate the algorithm as there was no way to precisely measure the position of the table during the experiments. Therefore, the georeferenced position presented in Figure 52 is relative to the UAS, as it can be observed in Figure 51 from the $T_{NED \to ENU}$ translation vector. The result was very satisfactory as the calculated values perfectly match the dimensions of the shelf.

Finally, to evaluate the proposed solution's performance when georeferencing targets at different heights, the distance between corners that are placed on top of each other was calculated and compared with the measured distance.

The bottom right corner of the top shelf is assumed at 0.85 m of height (z_{ENU}) and the bottom right corner of the bottom shelf is assumed at 0.35 m of height.

The georeferenced coordinates of both corners were calculated (Figure 53) and used to calculate the distance between them. The calculated distance was of 0.52 m, against 0.50 m of measured distance. Therefore, there was an error of 2 cm, which is small and could have happened because of the shelf's thickness or deviations when manually choosing the pixels in the image.

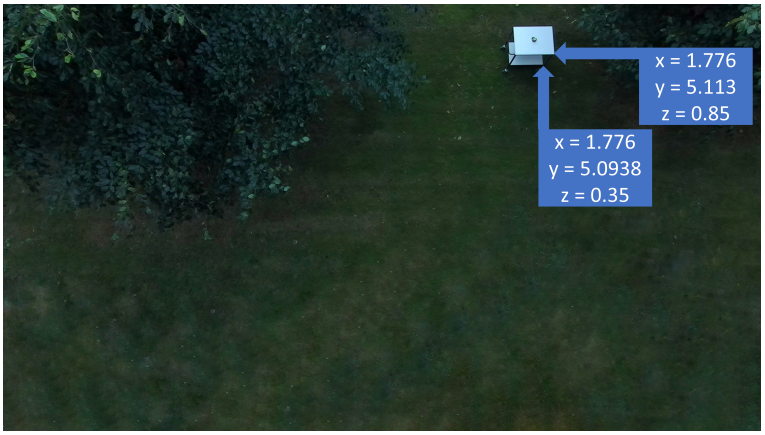


Figure 53 – Georeferenced positions of bottom right corner of the top shelf and bottom right corner of the bottom shelf.

Discussions

There are many different orientation frames in the process of direct georeferencing aerial images acquired by cameras mounted on UAS. First, the image frame, then the camera, gimbal, UAS (or IMU) and the geographic/world (NED and ENU) frames. In this work, in order to follow the literature and most common applications, two world frames were considered: the NED (North-East-Down); and ENU (East-North-Up) frames.

Despite camera, gimbal and UAS, NED and ENU frames were presented in this research, not all applications will require the use all these frames. Therefore, it is important to apply the method with the needed changes in each case.

As seen in the practical case study, it is important to understand the formats of the attitude data given by the flight control units or ROS, to be able to convert to the formats used in the direct georeferencing algorithm presented in this work. This is one of the reasons why this research is considered relevant as it explains the implementation steps so that the readers can understand the reasons behind the equations, in order to adapt the solution to their use.

In addition, it is important to note that accurate measurements are important for an accurate direct georeferencing. Therefore, GNSS RTK solutions should be used, in addition to highly accurate Inertial Measurement Units (IMU), magnetometers and gimbals. Furthermore, platform mounting measurements between the origins of the frames of reference (lever arms) should be included in the calculations as translation vectors. This

is often neglected in the UAS community, since the lever arms are short, in the range of centimeters. However, for high resolution sensors in the order of millimeters, or larger UAS or manned aircraft, the lever arms are necessary to account for.

Another important aspect to consider is that depending on the IMU orientation, there may be problems when computing the yaw angles when the camera is pointing down. A possible solution for this issue is to change the orientation of the z axis so that it is perpendicular to the world when the UAS is horizontally positioned. A way to eliminate this issue is to use quaternions in all steps of the direct georeferencing implementation. This could be interesting future work.

4.2.2 Direct georeferencing of RGB camera images in solar panels inspections in virtual environment

After finding defects in an image, such as Figure 54, where the vegetation is covering part of the panels, the pixels of the four vertexes of the defect's area were used by the georeferencing algorithm to calculate the UTM coordinates of the defect.

The georeferencing algorithm also uses the information that is recorded in the image file as metadata, such as GPS position and attitude of the UAS and gimbal when the picture was taken. In addition, the following intrinsic parameters of the sensor are considered, which were calculated for the sensor used as a reference for this research:

$$\boldsymbol{K} = \begin{bmatrix} \mathbf{3558.14} & \mathbf{0} & \mathbf{1224} \\ \mathbf{0} & \mathbf{3878.79} & \mathbf{1024} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{87}$$

The UAS flies with a fixed heading of zero degrees north and has the gimbal pointing straight to the panels, which are 30 degrees tilted to the east from the horizontal position. In the AirSim simulator, the gimbal orientation must be chosen. In the simulations performed in this research, the gimbal is assumed perfect. Therefore, the rotation matrix is calculated based on the gimbal angles, which are $pitch = -60deg$; $roll = 0deg$; and $yaw = -90deg$.

After calculating the UTM coordinates, a polygon was drawn on the raster layout



Figure 54 – Inspection product with pixel coordinates of defective area

plant, giving the location of the defective panels in the plant. Figure 55 shows the calculated polygon of the area with defective panels. The error between the expected and calculated midpoints of the polygons was 0.34 meters on the North axis and 0.26 meters on the East axis. This error is due to the tilt of the panels, as the georeferencing algorithm assumes a flat ground; and because the distance between the center of mass of the gimbal and the center of mass of the UAS was assumed nonexistent. As the standard solar panel has dimensions of 2.0 x 1.0 m, these results show that it is possible to determine the defective panels accurately.

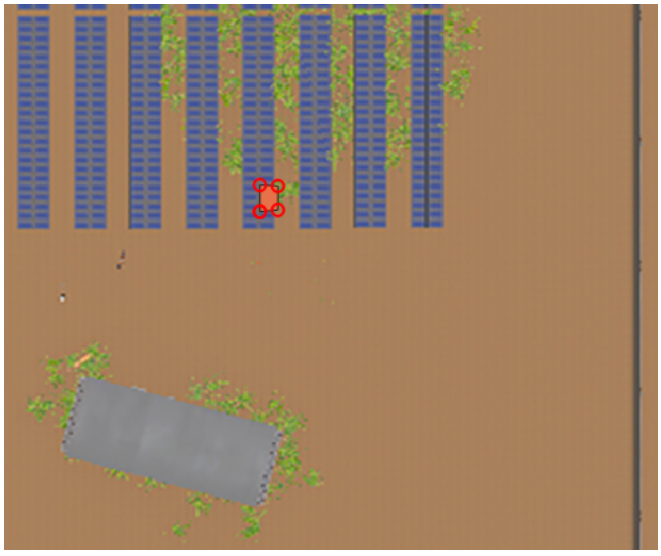


Figure 55 – Location of the defective panels in the solar plant layout

4.2.3 Results of the direct georeferencing of RGB-D images in collision avoidance systems for small autonomous vessels

In this research the direct georeferencing algorithm results can be evaluated through the tracking and positioning graphics of the targets, whose information were obtained through the USN SeaDrone on-board sensors. The success of positioning and tracking process of these target vessels was made possible because of the accuracy of the direct georeferencing algorithm results, and made possible for the SeaDrone to make the right decision to avoid the collision risk.

Real tests were performed to evaluate the proposed solution, the collision avoidance algorithm. First, a stationary boat was detected and tracked when the SeaDrone was moving on the direction of the target boat. As it can be seen in Figure 56, the detected stationary boat's position (gray) has a deviation of around 1.5 meters in the x axis and 1 meter in the y axis. The reasons for this deviation may be the fact that the detection returns the center pixel of the bounding box of the target boat and as the SeaDrone is moving, this center pixel may vary if the target boat is being spotted from different angles. Other reasons may be sensor reading errors, for example, the depth, IMU or even the GNSS, as no RTK solution is used. For the purposes of what concerns to this thesis research, which is the direct georeferencing algorithm evaluation, the good results of the tracking and positioning process, despite the little scale relative error, demonstrates the satisfactory performance of the direct georeferencing algorithm. In addition, it must be considered that due to the magnitude of the dimensions involved in a maritime navigation problem, such as distances between the vessels and their sizes, the deviations (that are not due to the direct georeferencing algorithm) are very acceptable and even insignificant, and would not compromise the operations.

The objective of the second test was to investigate how the system performed when rotating around the steady boat (Figure 57).

It is possible to notice that the there is a sudden jump for the position of the tracked boat. Again, this is due to the fact that the boat is big, so when the SeaDrone started the tracking, it was spotting one side of the boat and, after a while, it began to spot the other side of the boat. Again, even this unexpected result reinforces the demonstration that the tracking and positioning process performed well, and so, the direct georeferencing

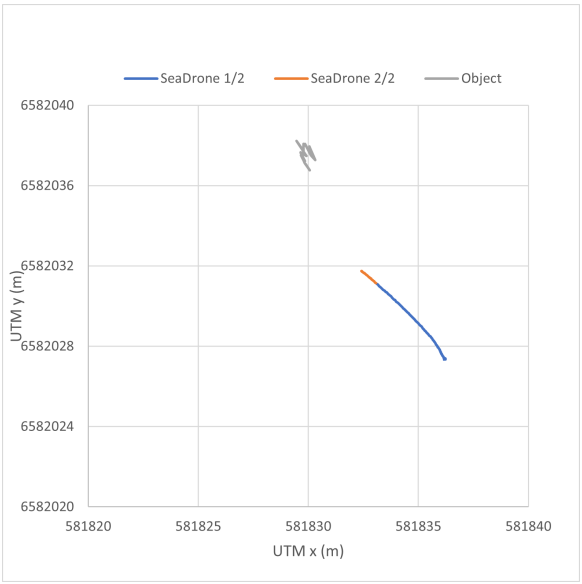


Figure 56 – SeaDrone moving straight tracking a steady boat



Figure 57 – SeaDrone rotating around the tracked boat

algorithm proved to be efficient.

Finally, a last test was performed, when the SeaDrone was moving and tracking multiple targets that were also moving (Figure 58). It is possible to see that the the objects 5 (red) and 6 (dark gray) are most likely the same target and that the tracking module assigned a new number tag to it in the tracking process. The same can be said about the object 1 (gray) and 2 (yellow). Probably the SeaDrone started tracking the object and because it started turning, it lost the object from the camera view and then assigned a new number tag when it was detected again.

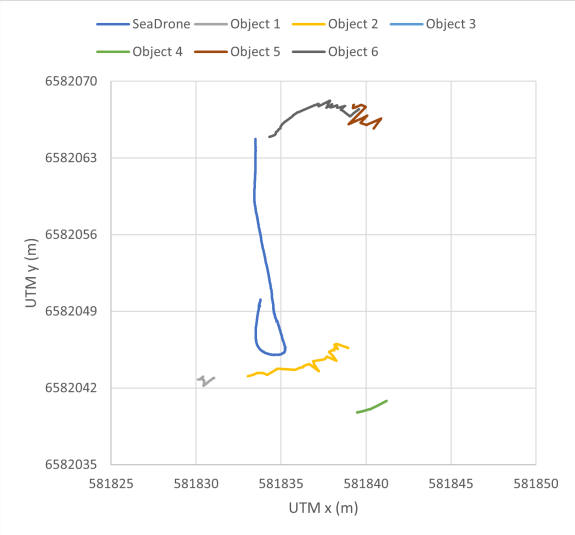


Figure 58 – SeaDrone moving when tracking multiple moving objects

All the three real navigation test show that the tracking and positioning algorithm, which depends on the direct georeferencing algorithm, performed in satisfactory terms, demonstrating the accuracy of its results, otherwise, all the collision avoidance process would be compromised and the decision making could not be correctly accomplished.

5- Conclusions

In this chapter the conclusions obtained in the two parts of this doctoral thesis will be presented.

5.1- Search and Rescue conclusions

In this research, a real-time path-planning for search and rescue with Model Predictive Control solved by Particle Swarm Optimization was proposed. The solution was implemented to be embedded in the UAVs' on-board computer and tested in a Software-In-The-Loop environment with flight dynamics simulations. The search area was defined using the International Aeronautical and Maritime Search and Rescue (IAMSAR) directives. In addition, the area was divided into a grid of cells, where each cell had a correspondent reward, referred to the IAMSAR's Probability of containment. Results were analyzed for missions where one, two or three Unmanned Aerial Vehicles (UAVs) were employed. To reach 50% of Probability of success, the performance of the group of three UAVs was on average 2.35 times faster than the single UAV search. The effects of the inclusion of wind information were also evaluated. When wind information was used, even if the wind speed was underestimated by 20%, the team of three UAVs achieved on average 4.4% higher Probability of success in 20 min of mission, when compared to the scenario where no wind information was used. The performance of the single UAV was also compared to a standard search pattern based on the IAMSAR's suggested pattern. The search using the proposed solution outperformed the standard search pattern in the first 20 min, with the additional advantage of being a real-time method that can deal with environmental dynamic changes and new mission directives.

5.2- Direct Georeferencing method conclusions

In this section the conclusions of the four applications of the direct georeferencing method will be commented.

This thesis presented a comprehensive method for direct georeferencing of aerial images taken by monocular cameras mounted on UAS. To perform the calculations, the method uses data about the camera sensor, relative position and orientation between camera, gimbal, UAS and world, and position and attitude sensor data.

Four case studies were performed to show the method's employment: georeferencing of a target in an image taken in a simulation environment; georeferencing an object from an image taken in a real flight; georeferencing a defective solar panel; and georeferencing boats in a collision avoidance task.

For an image taken in a simulation environment, the georeferenced target position was precisely calculated. In the practical experiment, where the corners of a table were georeferenced, there were small inaccuracies, which were expected, as in real life setups there are small terrain deviations, sensor noise, among other problems. Therefore, the main goal of this research, which was to present a comprehensive direct georeferencing method and its implementation, was accomplished. This included the mathematical fundamentals and the assessment of practical challenges explored in the case studies.

The direct georeferencing method was also used to localize a defective panel in a virtual solar plant built in Unreal Engine 4. The AirSim plugin was used to capture the images of the solar panels. The Ardupilot SITL flight controller was responsible for the sensor data that is used by the georeferencing algorithm. The solution was tested in a case scenario, where the georeferencing algorithm was used to locate the defective panel in the real world, which demonstrated to perform satisfactory, localizing precisely the defective panel.

Finally, a solution for georeferencing tracked boats for collision avoidance was presented. The system was equipped with a depth camera, IMU, magnetometer and GNSS sensors. The solution was developed in the Robot Operating System (ROS) framework. Results show that the direct georeferencing algorithm is able to correctly calculate the position of the targets in the real world, allowing the tracking of the boats. This evidences the accuracy of the direct georeferencing method proposed in this thesis.

5.3- Proposals of future works

A didactic and practical exposition of MPC

Throughout this research, while working in the development of the search and rescue solution, it became evident that there was a lack in the scientific literature, such as papers, articles and books, that presented the MPC algorithm in a didactic and practical form, that make easy for researches to understand the method. In order to match this goal, and then, supply the fault of such literature, solving the great difficult researches may face, it is intended the future publishing of a paper with an easy understanding exposition of the MPC, showing in detail each step of the implementation of the method, following the same structure of the work in direct georeferencing method, presented in Chapter 3. The main idea is to use the theoretical concepts, demonstrated in Chapter 2, and apply them in an simple control problem, as an example, in order to provide the reader a complete and comprehensive overview of the MPC algorithm implementation.

Hardware-in-the-loop simulations for UASs in SAR autonomous missions

In future work, Hardware-In-The-Loop simulations will be conducted in order to prepare the system for flight tests, for UASs in SAR autonomous missions.

Direct Georeferencing method using quaternions

As pointed in Section 4.2.3, in Chapter 4, about the discussions over the direct georeferencing method in this thesis, the use of quaternions in all steps of the direct georeferencing implementation, is a new trend to be explored in the theme, and a very interesting possibility for future work and publication.

Appendix A - Search and Rescue Cost Function and Simulation Environment

This appendix is an extract of the paper published in the Sensors journal (ANDRADE, 2019)

Optimization Technique

In this application, the Particle Swarm Optimization (PSO) (EBERHART; KENNEDY, 1995) technique is used to find an optimal set of airspeed (u_v) and roll angle (u_ϕ) that minimizes the cost function. PSO is a meta-heuristic optimization method where the particles (solutions) are updated every iteration based on the best global and local solutions. In this application, a standard PSO algorithm was implemented using CUDA C programming language in order to benefit from the parallelism of the NVIDIA® Graphics Processing Unit that is assumed to be used in the UAV on-board computer.

The algorithm was set to run a fixed number of iterations on every loop. In addition, the number of particles must be defined. These two parameters affect the processing time and need to be fine-tuned according to the requirements.

The initial solutions are initiated with random values following the uniform distribution, where the minimum and maximum values are the defined boundaries of the airspeed and roll angle control inputs (Equations (12) and (13)).

SAR Directives Applied to UAVs Equipped with Remote Sensing

The Search and Rescue (SAR) consists, according to the Department of Defense (DoD) of the United States of America, in “the use of aircraft, surface craft, submarines, and specialized rescue teams and equipment to search for and rescue distressed persons on land or at sea in a permissive environment” (DOD, 2007). This work focuses on the sea cases; therefore, the following description emphasizes sea SAR missions. In addition,

as only Unmanned Aerial Vehicles (UAVs) are used in this work, only the directives for aircraft facilities are studied.

Search Area

According to the International Aeronautical and Maritime Search and Rescue (IAMSAR) Manual (IAMSAR, 2007), the Total Adjusted Search Area (A_t), which is the mission's actual search area, is calculated based on the Total Available Search Effort (Z_{ta}), the Optimal Search Area (A_o) and the targeted Probability of Detection (POD). The first is a measure of the total area that a set of search facilities can effectively search within limits of search speed, endurance, and sweep width. The second is the search area which will produce the highest probability of success when searched uniformly with the available search effort and is essentially calculated based on the leeway and the Datum probable position error. Leeway is the the movement of a search object through water caused by winds blowing against exposed surfaces and Datum is a geographic point, line, or area used as a reference in search planning, such as the "Last Known Position" or the "Estimated Incident Position".

If the Total Available Search Effort (Z_{ta}) is smaller than the Optimal Search Area (A_o), a strategy must be chosen to balance the Probability of Detection (POD) and the Total Adjusted Search Area (A_t). Usually, the chosen strategy is to fly on higher altitudes, increasing the sensor's footprint or the crew's field of view while decreasing the POD. However, in this work, as UAVs equipped with automated remote sensing are assumed to be used, resolution requirements usually can not be relaxed. Therefore, no trade-off between the POD and the search area is made and the POD is set to its maximum value of one, which makes the Total Adjusted Search Area (A_t) equal to the Total Available Search Effort (Z_{ta}).

In order to calculate the Total Available Search Effort (Z_{ta}), the sweep width (W) must be defined. When employing UAVs equipped with automated remote sensing in such missions, the sensor being used has a direct influence on this parameter. Altitude, view angle and image quality may affect the capability of identifying a survivor or an object on the sea. This is especially important to be taken into account because, if the image does

not contain the object of interest properly recorded, the computer vision algorithm will not identify it, independently of the ability that the algorithm has on identifying an important occurrence on an image. This can occur due to low image quality or too long sensing distances, making the object of interest imperceptible.

In the IAMSAR manual, the sweep width is calculated based on the altitude of the aircraft, the visibility and the sensor system specifications. In sea SAR missions with aircraft facilities and visual search, the Corrected Sweep Width (W) is adjusted regarding the weather, velocity and crew fatigue correction factors. However, these factors can be excluded from the equation when automated remote sensing systems are used and the system's velocity constraints are respected. Therefore, in this work, the Corrected Sweep Width (W) is considered equal to the original Uncorrected Sweep Width (W_u).

Finally, the Search Effort (Z_a), which represents the area which can be covered by a specific facility, is calculated by:

$$Z_a = V \times T \times W, \tag{88}$$

where V is the Search Facility Speed (average speed) in [m/s], T is the Search Endurance in [s] and W is the Sweep Width in [m].

Note that the Search Endurance is the time available for the facility to fly looking for the survivors. The IAMSAR manual considers this time as 85% of the lower value between the Daylight Hours Remaining and the On-Scene Endurance. This is due to the fact that human crew is often only able to search with visible light. In contrast, UAVs are often capable to equip sensors that are not affected by that, such as infrared cameras, which allows the task to be done even along the night. This is a considerable advantage of using UAVs equipped with remote sensing systems.

By summing the Search Effort of all facilities, the Total Available Effort (Z_{ta} in [m^2]) can be found:

$$Z_{ta} = \sum_{f=1}^{F} Z_{af}, \tag{89}$$

where F is the number of facilities.

As described above, in this work, the Total Adjusted Search Area (A_t) is equal to the Total Available Effort (Z_{ta}). Therefore, for Single Point Datum, the Length and the Width of the search area are given by the square root of the Total Available Effort (Z_{ta}) as defined by the IAMSAR manual.

0%	0%	5%	0%	0%
0%	6%	8%	6%	0%
5%	8%	24%	8%	5%
0%	6%	8%	6%	0%
0%	0%	5%	0%	0%

Figure 59 – Initial probability table example.

Probability Map

The Probability of containment (POC) distribution in the search area is very important to guarantee efficient employment of the SAR facilities. When the initial indications do not provide enough information about the area, a standard distribution is assumed. The two most used types of standard distributions are the standard normal distribution and the uniform distribution, according the nature of the datum. For datum point and lines, the standard normal distribution is used. For datum areas, the uniform distribution is the most used. In this work, only the single point datum case is studied. Single point datum occurs, for example, when there is no significant leeway (e.g., when the target is a person in water (BREIVIK et al., 2011)).

The probability map is a set of grid cells where each cell is labelled with the probability of containing (POC) the search object in that cell. As the the probability map follows a probability distribution function, the total sum of all cells should be equal to 100%. An example of probability table for single point datum with 5×5 cells is shown in Figure 59.

Cost Function

An exploration cost function was developed based on (TRODDEN; RICHARDS, 2008) to search a given area.

The region of interest is divided into $M \times N$ square cells of a width (r_e in [m]), whose value must be chosen to be smaller than the optical imaging sensor's footprint radius (R_e in [m]) times the square root of 2. The sensor radius is equal to the radius of the circle inscribed in the sensor's footprint. Figure 60 shows an example of a 4×4 grid with 100 m of cell width (r_e) and a UAV at position **Cx** equipped with a sensor with 100 m of radius (R_e).

The matrix $\mathbf{B}^i \in R^{M \times N}$ is used to identify if a cell was visited by the ith UAV. The matrix $\mathbf{b}^i \in R^{M \times N}$ is used to identify if a cell is planned to be visited by the ith UAV in the MPC horizon. In every $M \times N$ matrix used to identify if the cells are visited, each element has an associated value of 1 if the referring cell is visited or 0 if it is unvisited. Each cell has also an associated reward, given by $\phi \in R^{M \times N}$.

The function $J(\bar{\mathbf{x}}_k)$ is the sum of all cells associated value (1 or 0) in the step k times the correspondent reward:

$$J(\bar{\mathbf{x}}_k) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \phi_{mn} y_{mn_k}(\bar{\mathbf{x}}_k), \tag{90}$$

with

$$y_{mn_k}(\bar{\mathbf{x}}_k) = (\|\mathbf{Cx}_k - r^1_{mn}\| < R_e \wedge \|\mathbf{Cx}_k - r^2_{mn}\| < R_e \wedge \|\mathbf{Cx}_k - r^3_{mn}\| < R_e \wedge \|\mathbf{Cx}_k - r^4_{mn}\| < R_e) \vee y_{mn_{k-1}}, \tag{91}$$

where r^1_{mn}, r^2_{mn}, r^3_{mn} and r^4_{mn} are the four vertices of the cell (Figure 60) and $y_{mn_{k-1}}$ is the associated value of the cell in the previous horizon step.

The starting value of y_{mn_0} is given by the logical sum of the matrices of already visited cells of all UAVs and the matrices of cells planned to be visited by other UAVs:

$$y_{mn_0} = \mathbf{B}^i_{mn} \vee \mathbf{B}^j_{mn} \vee \mathbf{b}^j_{mn}, \forall j \in \{0, ..., I-1\} \backslash \{i\}. \tag{92}$$

Finally, $F(\mathbf{x}_K)$ is the terminal cost. This function is necessary for the algorithm to consider the search beyond the prediction horizon by having a *cost-to-go* term. It is given

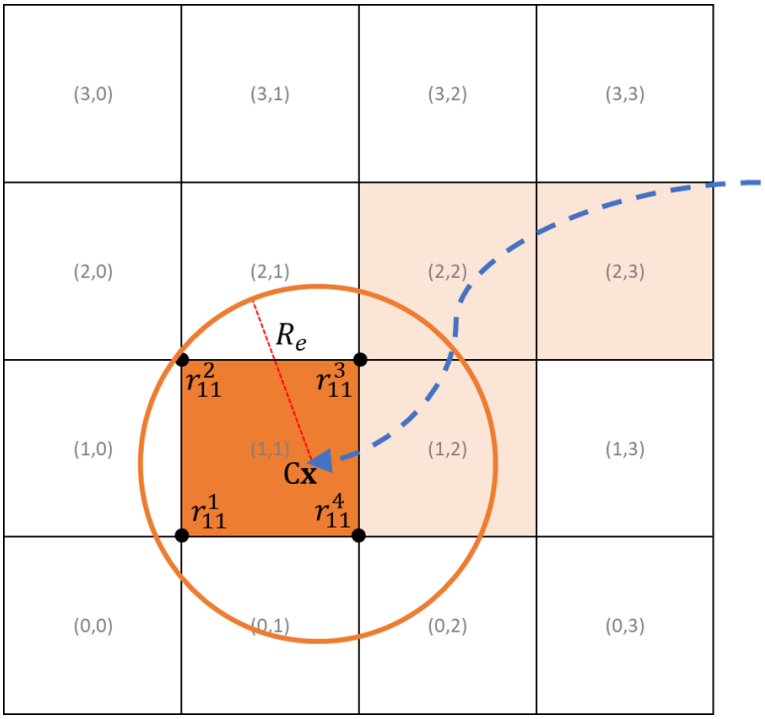


Figure 60 – Cells grid example.

as the minimum Euclidean distance from the latest state of the UAV in the horizon, to the center of the closest unvisited cell, weighted by the correspondent reward, in the end of the horizon:

$$F(\mathbf{Cx}_K) = \min_{\forall m \in O, \forall n \in P} \frac{\|\mathbf{Cx}_K - \mathbf{r}_{mn_K}\|}{\phi_{mn}}, \tag{93}$$

where $O \subseteq M$ and $P \subseteq N$ are subsets of all unvisited cells and $\mathbf{r} = [x, y]$ are the north and east positions of the cell's center.

Simulation Platform

The optimization algorithm was written in CUDA C programming language in order to benefit from the parallelism, with the goal to embedded it on a NVIDIA® Jetson™ board in the future for Harware-In-The-Loop simulations and field tests.

The simulations were run in a laptop with the NVIDIA 940MX graphics card, which has 384 CUDA cores and 8 GB 128-bit memory. With the parameters presented in this study, each UAV is able to run one optimization in around 400 ms when three UAVs

perform the optimization at the same time. When only one UAV is used, each optimization loop takes around 250 ms. Therefore, it is expected that the NVIDIA® Jetson™ board, which has 256 CUDA cores and 4 GB 64-bit memory will be able to run one optimization in less than 400 ms. If this performance is not achieved, the parameters can be fine-tuned to achieve shorter processing time. Another possibility is to implement an optimization stopping feature that will run as many iterations as possible within a given time, instead of a fixed number of iterations.

It is also relevant to mention that the optimization time of each step was adjusted to 400 ms also when using only one or two UAVs. This was done by inserting a delay, so each UAV's optimization time is the same for simulations when one, two or three UAVs are used. Therefore, this gives a fair comparison of the results.

Appendix B - Solar Plants Inspection Simulation Environment

This appendix is an extract of the paper published in the AIRPHARO international conference (DE ALCANTARA ANDRADE et al., 2021) with the description of the complete system, that used the direct georeferencing method presented in this thesis.

Abstract

This research presents the development of a virtual reality simulation environment for Unmanned Aerial Systems (UAS) solar plant inspection. The objective of this work is to provide a tool to test autonomous inspection and computer vision algorithms and generate realistic synthetic data for deep learning. These techniques demand realistic synthetic data, which can be made available by high-quality graphics engines, such as the ones used for game development. In this work, Unreal Engine 4 is used to host the virtual solar plant. The solar panels were modeled using Blender and Photoshop. Microsoft's AirSim plugin is used to simulate the UAS motion, together with the ArduPilot Software-In-The-Loop flight controller. The environment was evaluated through a virtual autonomous inspection of a plant with 9200 panels, where a georeferencing algorithm was used to locate the defective solar panel in a raster plant layout, based on the pixel position of the defects in the aerial images. The virtual inspection resulted on more than 1000 images and the localization of the defective panels in the layout plant using the georeferencing algorithm had an error of 0.34 meters on the North axis and 0.26 meters on the East axis, which is acceptable for large solar plants with sparse modules' arrangement.

Introduction

The increasing demand for clean and renewable energy sources is driving the growth of solar plants worldwide, as well as its extension and complexity. Such solar plants

have a considerable number of photovoltaic panels, and the inspection of the many panels for maintenance is human-intensive work, claiming for autonomous and reliable solutions.

Unmanned Aerial Systems (UAS), popularly known as drones, are being increasingly used for inspection of solar plants. For example, an autonomous flight planning algorithm for solar plants inspections with UAS was presented in (HENRY et al., 2020). In that work, the UAS was equipped with a thermal camera and an RGB camera. The system was able to automatically detect and estimate the location of the faulty solar panel in the plant. Such algorithms are normally exhaustively tested in simulation environments, as the one presented in this work. Among other reasons, this has to be done before real flights in order to facilitate the development, as it prevents risk to people and to the equipment in case of problems.

In such inspections, after the fault detection phase, it is necessary to locate the defective panel in the real world for maintenance or replacement purposes. In the case study presented in this work, a georeferencing algorithm is used to locate the defective panel in the solar plant. In the literature, there are georeferencing solutions that are able to address this challenge, such as (HEMERLY, 2014), where the author presents an algorithm for automatic georeferencing images obtained from UAS cameras, dismissing ground control points. In addition, in (LEIRA et al., 2015), the authors present a georeferencing algorithm for thermal images from a lightweight thermal camera coupled in small fixed-wing UAS.

As mentioned, the development of autonomous and reliable inspection tools demand exhaustive tests before operationalization. Simulation tools, such as the one developed in this work, are important means for testing such solutions. Recently, with the advancements of computational power and software complexity, physically and visually realistic simulation tools were made available. In (BONDI et al., 2018), the authors present an autonomous system for UAS used to combat poaching, based on thermal camera images obtained from the UAS. The solution employs state-of-the-art artificial intelligence techniques, such as the Faster R-CNN detection technique. Simulations with AirSim were used for a live demonstration of the system. In (ÇETIN; BARRADO; PASTOR, 2020), the authors propose a deep reinforcement learning architecture for counter-drone measures by using artificial intelligence. The simulation environment reproduces stationary obstacles, such as trees and buildings, along with moving obstacles, such as other non-hostile UAS. In (AMER et al., 2019), the authors propose a deep convolutional neural network-based

system for UAS aerial autonomous navigation for predetermined and known trajectories, using only images from an onboard camera. The system was thought for autonomous UAS navigation in regular trips, through the same locations, dismissing GNSS. Tests were taken in a virtual reality environment, using the Unreal Engine and AirSim.

In addition, realistic simulation tools are being used to generate synthetic data for machine learning algorithms, such as in (CHRISTIE et al., 2018), where the authors explored training an object detector using only synthetic data to detect rooftop stacks for UAS sampling tasks. Synthetic data was also used in (MARCU et al., 2018), where 3D reconstructions of real places were used for learning depth from in-flight images and further segmentation into UAS safe-landing and obstacle regions.

A similar approach to what is presented in this work can be found in (WANG et al., 2017), where a real-time virtual reality Unreal Engine 4 and AirSim based platform for UAS deep learning training is presented. The platform images reach a good level of realism and sophistication, reproducing natural environment features in detail, such as geographic accidents, weather changes due to characteristics of each season, etc. Among its many benefits, it allows the capture of images for deep learning for autonomous flight and obstacle avoidance. The main difference from this work is that here the virtual reality environment was developed specifically for solar plant inspection with UAS, including the modeling of the solar panels and the evaluation of a case study.

Other relevant research related to the AirSim plugin should be cited, such as (HENTATI et al., 2018), where the authors evaluate the most suitable flight simulators for UAS, highlighting their goals, weakness and strengths. X-Plane, FlightGear, Gazebo, JMavSim, AirSim and UE4Sim flight simulators are compared and evaluated in the research. AirSim was highlighted as having great visuals with the downside of demanding powerful GPUs. In (REDULLA; SINGH, 2018), the authors developed and compared two simulators (MatLab and AirSim) to investigate optimal strategy. The Pedestrian Tag game and the Homicidal Chauffeur game were performed in both simulators, as well as other differential games, in order to compare with other strategies from different methods. In (HO et al., 2018), the authors make an experiment in which chosen maneuvers and tasks are accomplished in a virtual environment using AirSim; parameters are collected and performance is improved using machine learning. The optimum parameters are then applied and tested in real flights. In (YAO et al., 2018), the authors present a virtual reality, VR-based, test platform for autonomous-driving vehicles, combined with AirSim system

and the UE4; the controller-in-the-loop simulation method was used in the tests in a virtual environment under different conditions of traffic and driving, and the results were used to optimize the control system.

The use of drones for inspections has many potential advantages. The main challenge to deploy autonomous UAS for inspections is the balance between safety and operational performance. In other words, the failure rate of the drone should be low enough to not cause any safety threat to humans or infrastructure. For this reason, a full and comprehensive virtual environment allows the development and evaluation of different scenarios. Therefore, providing an appropriate risk estimate before field deployment.

With the advancements of computer science and the quality of computer hardware, it became possible to build platforms that use Virtual Reality to create environments that are not only able to simulate realistic flight dynamics, but are also visually close to reality. This is a factor of extreme relevance for the development of computer vision and deep learning algorithms.

In this work, a virtual reality solar plant environment for autonomous drone inspection simulation was built using the AirSim plugin and the Unreal Engine 4 (UE4). To evaluate the environment, a virtual inspection was performed with an autonomous inspection algorithm using the ArduPilot Software-In-The-Loop (SITL). Finally, a georeferencing algorithm was created and used for the localization of defective panels in the solar plant.

This solution can be used for generating synthetic data and for optimizing and testing inspection systems without the need of real flights, which are costly, demand special operational measures and have a risk of damage to the drone and the plant in case of a crash. Until the submission of this work, no other work about solar plant inspection using UE4 and AirSim was found by the authors. Therefore, the main contributions are listed as follows:

1. The development of a 3D virtual reality solar plant environment for drone inspection simulation with AirSim and ArduPilot SITL flight controller;

2. The development of an algorithm for mission planning based on solar panels inspection characteristics, such as pixel resolution requirements and solar piles tilt; and

3. The development of a virtual autonomous inspection algorithm, including a georeferencing algorithm for localization of the defective solar panels in the plant

Virtual Reality Environment

In this section, the development of the virtual reality environment is presented. To build the 3D models of the solar panels, the Blender 3D computer graphics software (COMMUNITY, 2018) and the Substance Painter texture software (ADOBE INC., n.d.[b]) were used. Unreal Engine 4 (UE4) was used to simulate the solar plant. Finally, the AirSim simulator was used for the drone physics simulation, acting as a bridge between the ArduPilot Software-In-The-Loop simulator and the UE4.

Solar Panels 3D modelling

To create a virtual solar plant, the main characteristics considered in this work are: the organization of the solar plant; the appearance of the solar panels; the number of panels that are arranged on each support; and the distribution throughout the plant.

The appearance of the solar panels was the first issue addressed as the stage of proper modeling and texturing of the panels is of tremendous importance to ensure the best similarity between the virtual and the real environment.

A 3D model can be divided between mesh and texture. The mesh is the description of the geometric shape modeled using polygons. The texture is the file that describes which image must be represented in each polygon, so that the final result refers to a realistic model. To model the mesh (Figure 61), the Blender free and open-source 3D computer graphics software together with the open-source CAD Transform plugin was used. The texturing (Figure 62) was done using the Substance Painter software and fine-tuned using the Adobe Photoshop (ADOBE INC., n.d.[a]) software for better contrast and saturation. In the texturing stage, it is vital to keep in mind the characteristics of real solar panels for the representation to be realistic. In the case of this work, prefabricated textures available in the Substance Painter software were used to represent the metal frame and the characteristic appearance of solar cells on panels. Characteristics such as reflectivity, roughness and occlusion shadows were considered.

The final step to create the 3D model was to apply the texture to the mesh. This

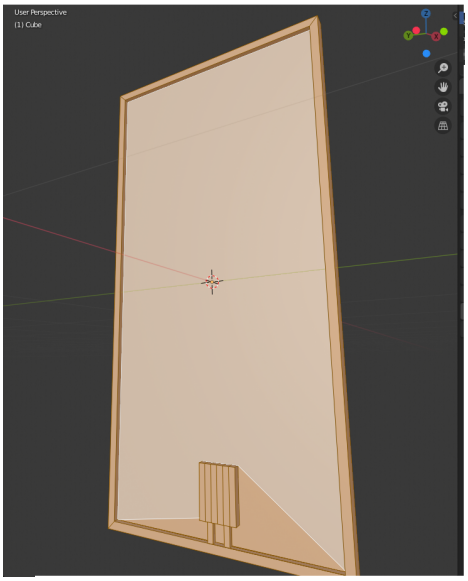


Figure 61 – Mesh of the virtual solar panel

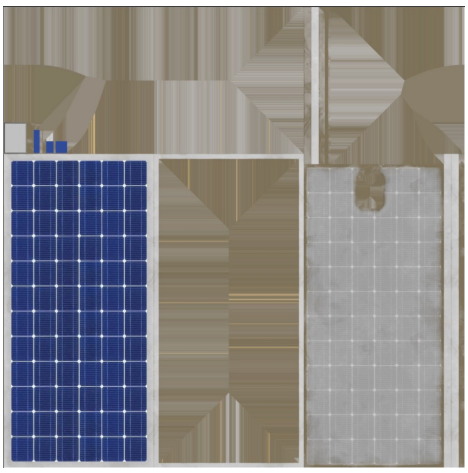


Figure 62 – Testure of the virtual solar panel

step starts with the UV unwrapping and consists of representing all the polygons that make up a 3D mesh in a 2D image, known as UV Map, so that the images that form the texture are easily applied on top of this representation and recognized appropriately by software, which in the case of this work is the Unreal Engine 4.

For using the Blueprints system in UE4, a function was created to control the angle of the panels in relation to a support base so that they could be easily rotated depending on the sun's position. This function was created following these steps:

1. Import the 3D models into UE4, separating the fixed part from the rotating part into independent meshes and convert them into Static Meshes;

2. Create a Blueprint Class file inside a preferred folder using the Content Browser;

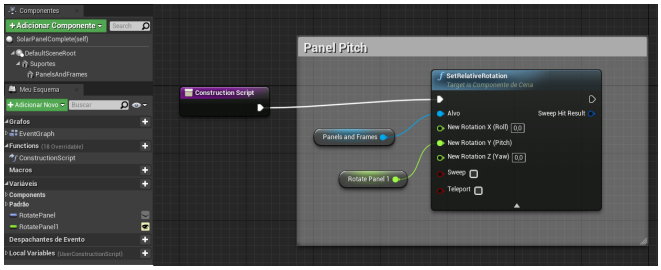


Figure 63 – Blueprint setup for the rotation function

3. In the Blueprint file, import the static meshes into the DefaultSceneRoot 63 (it is important to organize the hierarchy with the rotating part inside the fixed one);

4. Create a float variable inside the Blueprint, marking the Instance Editable box so it is possible to edit this variable in real-time inside the environment.

5. Use the slider range to set the angle range for the rotation. In this solar panel's case, the range $[-66deg; 66deg]$ was utilized; and

6. Inside the ConstructionScript, connect the items as shown in Figure 63. The function SetRelativeRotation was used, and the variables created were connected at the shown ports.

It is important to note that some changes in this process may be necessary depending on the application.

3D Environment

The 3D models were imported into the UE4 and a terrain was created of dimensions similar to an existing solar plant, in order to be as close as possible to a realistic one. The panels were arranged on the map using the Instance Tool plugin, to achieve higher precision in the positioning and to make the process more optimized, given the large number of panels to be placed. To increase the realism of the scene, models of a building, a worker, vegetation around the panels and mountains on the horizon were also included. The final aspect of the environment can be seen in Figure 64.

An important remark is that the PlayerStart instance was placed at position (0,0,0) of the UE4 environment for compatibility with the AirSim plugin and the ArduPilot SITL



Figure 64 – The unreal environment

flight controller. This also reduces the work needed to convert the coordinates between the virtual reality world and the real world.

AirSim

One of the main reasons for choosing Unreal Engine 4 (UE4) for this work is the fact that, as pointed out in other studies (WANG et al., 2017) (REDULLA; SINGH, 2018) (YAO et al., 2018), it has a cutting edge technology, not only by itself but also improved by AirSim (SHAH et al., 2017), which has a central role in the application reported in this work. AirSim is an abbreviation for Aerial Informatics and Robotics Simulation, a UE4 plugin created by Microsoft. The simulator promises to be relevant in the step of building algorithms for autonomous vehicles, e.g. cars, UAS, and rovers. Because it is open-source, AirSim has a very active community, with researchers from institutions all around the world. AirSim's contribution to this research is made by delivering the simulation of the physics, and, therefore, sensors and actuators' behavior to the flight controllers used to control the UAS. This allows developers to test their solutions inside a virtual reality 3D environment (Figure 65).

AirSim has a Python library, which facilitates transferring the information gathered in missions, as it presents methods and tools for extracting data generated within the simulation. A good example is the function to acquire pictures taken by the camera sensor. The plugin also provides a ”REC” button inside the UE4 to save the current camera sensor image, without the need to build any code for it. These are extremely important tools for

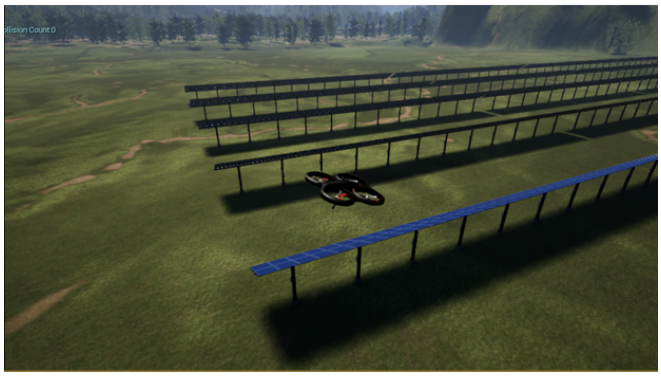


Figure 65 – Drone flying in a virtual environment

data collection and processing for machine learning and computer vision solutions, given that the data extracted from the environment is a key input for these.

After building AirSim, it is possible to test the plugin's features in a preliminary environment called "Blocks". However, creating personalized environments is normally needed. AirSim's website provides several tutorials on this matter.

In this work, the UAS guidance and control software should be integrated into AirSim by the Ardupilot's SITL flight controller. Therefore, the drone can be easily controlled using MAVlink commands. This is exemplified in section 5.3. AirSim is also compatible with ROS (Robot Operating System) (STANFORD ARTIFICIAL INTELLIGENCE LABORATORY ET AL., 2018).

Virtual autonomous inspection

The virtual reality solar plant simulation environment was designed to be used for testing inspection algorithms and generating synthetic data. Therefore, in this section, the development of the virtual autonomous inspection solution is presented. Its flowchart can be seen in Figure 66.

First, the mission parameters are defined. This includes the list of latitude, longitude and height of the solar panels' piles to be inspected, as well as the tilt of the solar trackers. Based on the mission requirements, such as pixel resolution, the distance from the drone to the panels is defined and used by the mission planning algorithm to calculate the mission waypoints. In addition, before the mission starts, the camera resolution and field

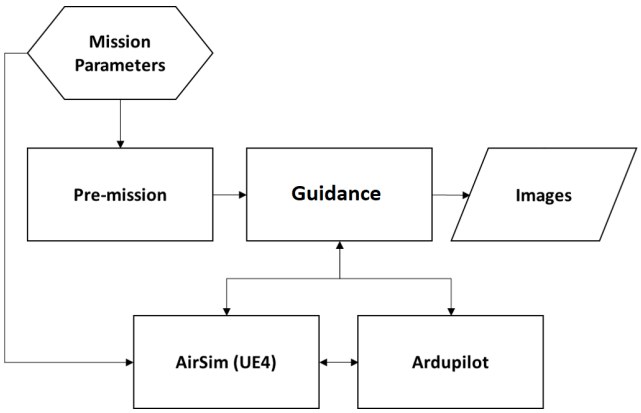


Figure 66 – System flowchart

of view (FOV) need to be set at the AirSim plugin configuration file.

The list of waypoints is used by the guidance Python script to configure the mission in the ArduPilot Software-In-The-Loop (SITL) flight controller. The guidance script is also responsible for sending the mission parameters, such as fixed heading value and desired speed, as well as commanding the drone to arm, takeoff and perform the mission.

The flight dynamics are simulated by AirSim physics engine; and the ArduPilot is responsible for the drone controls based on mission configuration and sensor data gotten from the simulated flight dynamics. The Unreal Engine 4 exhibits the flight as a 3D real-time virtual reality animation.

During the mission execution, the guidance script also gets the camera images from AirSim by using AirSim's Python library; includes the Exif (Exchangeable image file format) metadata, such as sensor information from ArduPilot; and saves the images in a local folder. Therefore, the images are ready to be analyzed by a defects detection solution, which is not in the scope of this work.

Finally, after the defects are detected in the images, the georeferencing algorithm is able to calculate the location of the defective solar panels in the solar plant based on the location of these defects in the image and the image metadata. This procedure is described in the next section.

Mission Planning

In the mission planning phase, the solar panels' piles coordinates and tilt must be provided. Universal Transverse Mercator or World Geodetic System coordinates are accepted as world coordinates. As the inspection site is a virtual reality solar plant, the coordinates must be converted from the Unreal Engine 4 (UE4) coordinate system to world coordinates. For this conversion, the world coordinates of the starting position of the UAS must be defined. These coordinates correspond to the "PlayerStart" position in the UE4 environment, which must be in the position $[x, y, z] = [0, 0, 0]$ (Figure 67).

The list of coordinates, together with the tilt information are used by the mission planning algorithm to calculate the actual flight path that the UAS should follow based on the desired distance from the vehicle to the solar panels, as seen in Figure 68. In the figure, the gray line represents the solar panels' pile, the yellow line represents the actual flight path and the red line represents the normal between them, which length is the required distance between the drone and the solar panels for complying with the mission requirements, such as pixel resolution and imaging geometry. For high fidelity thermal inspection of solar panels, a ground resolution of 3 cm is required. The images should be captured as perpendicular to the panel surface as possible, while omitting sun glare and shadows (FLIR, 2020).

Guidance

ArduPilot(ARDUPILOT. . . , 2019) is an open-source unmanned vehicle Autopilot Software Suite with an extended number of functionalities. It is one of the main firmware used in UAS' flight controllers. It uses the Micro Air Vehicle Link (MAVLink) protocol (MAVLINK. . . , 2019), which has several built-in commands that can be used to send commands and receive data from the vehicle.

In this work, a Python script was developed to be embedded on the UAS' onboard computer. This script uses mainly MAVLink's Python library (pymavlink).

The commands must be sent in the correct order for the drone to perform the

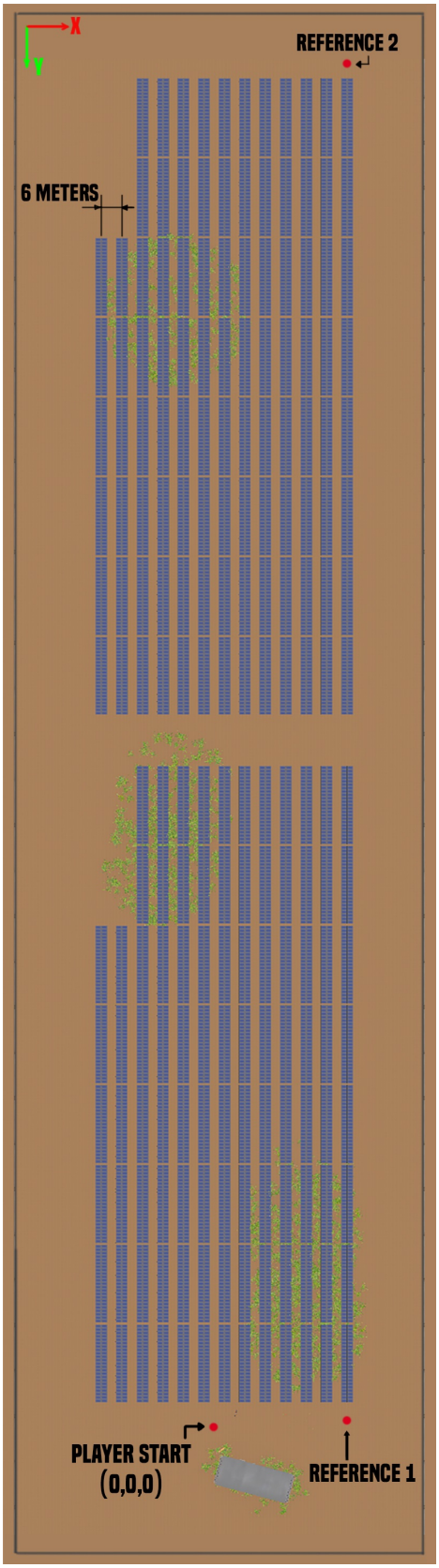


Figure 67 – Virtual Solar Plant

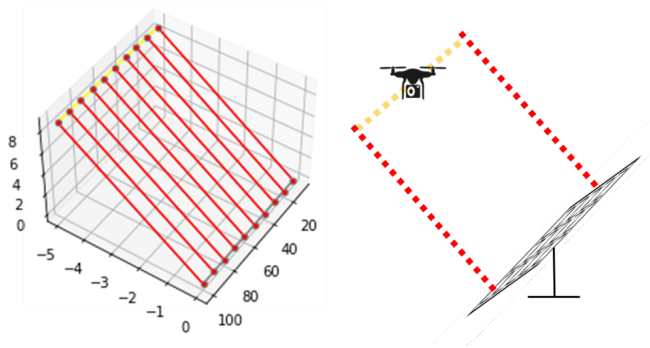


Figure 68 – Mission Planning. Gray line represents the solar panels pile; yellow line represents the calculated flight path; and red line is the normal between the drone flight path and the solar panels pile.

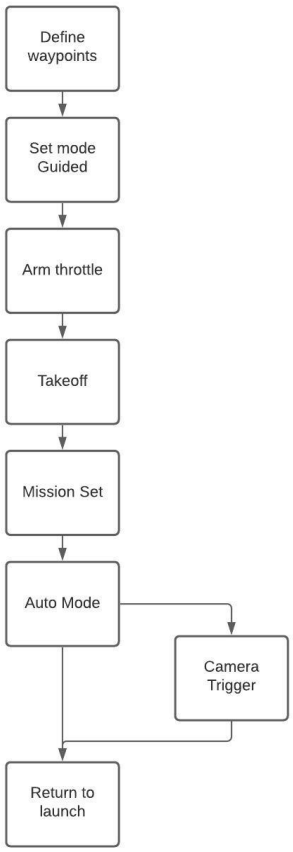


Figure 69 – Guidance algorithm flowchart

desired mission. The flowchart of the commands can be seen in Figure 69.

First, the communication between the Python instance and the flight controller is set, where the target system and component parameters are obtained. Next, the waypoint's coordinates are sent to the flight controller. In this case, a text file format (TXT) document with the latitude, longitude and altitude of each waypoint is loaded by the

Python script, which sends them to the flight controller using MAVLink messages of type MISSION_ITEM_INT.

After that, the flight mode is changed to GUIDED mode and a message to arm throttle is sent to start the motors, followed by the takeoff message. After taking off, the mission is set and finally started when the flight mode is changed from GUIDED to AUTO. From this point, the UAS autonomously follows the waypoints and returns to the launch coordinates in the end, disarming the motors and reestablishing the initial state being ready for the next mission.

Camera trigger

An image is captured every time the UAS flies a user-defined distance. To capture the images, AirSim's Python library is used to call the AirSim Image Application Programming Interface (API). The API returns an 8-bit array, that is saved as a JPEG RGB image in a temporary folder using the OpenCV library.

For simplicity, in the virtual reality environment used in this project, only RGB images were captured. However, the concept is the same for any type of imaging sensor. Therefore, the algorithms used in the simulated inspection can be easily adapted to real-world inspections, where other sensors may be triggered by a similar procedure.

Before the camera triggers, the simulator is paused and then resumed after saving the images. This must be done because, depending on the resolution, it may take a while to render the images, causing the simulator to misbehave.

Images metadata

The Exif metadata is a form of data present in image files. The data can vary from camera brand to the attitude of the platform, such as UAS' Euler angles.

In this work, for each picture taken by the UAS, GPS location (latitude, longitude and altitude); camera information; date and time of the photo; and resolution are inserted

in the image files as Exif metadata using the Piexif Python library. These data can later be used for gathering information about the conditions when the picture was taken. This information can be used to ascertain the position of the solar panel present in the image, determine the solar incidence with respect to the time that the picture was taken, how it can affect the inspection quality, among other uses.

Localization and Mapping

In this section, the procedure to create a georeferenced raster layout plant is presented, as well as the mathematical formulation of the aerial georeferencing to locate the defective panels in the raster, based on the pixels of the defects in the images acquired by the UAS.

Solar Plant Raster

The orthophoto produced for this paper was taken inside the Unreal Environment. This can be done by selecting the TOP orthographic view inside the Viewport at the upper left corner. Then, marking the Lit view mode it is possible to take the best aerial shot of the environment for the GeoTiff creation.

The Georeferencing add-on in QGIS 3.10 LTS (QGIS DEVELOPMENT TEAM, 2009) allows the georeferencing of raster images to a geographic or projected (metric) coordinate system, by creating a new GeoTiff image. The simplest way to georeference a raster is to locate points in the file, whose coordinates were measured as accurately as possible. At least 4 points are needed, spread across the 4 corners of the image, and the more points (coordinates) are informed, the better is the outcome of the process. In this work, the four vertexes of the fences are the points used to generate the GeoTiff.

Aerial Georeferencing

Georeferencing involves basic mathematical fundamentals, employing simple linear algebra operations, but demands very accurate references of the objects' positions in order to produce correct results. In this section, the georeferencing of a pixel position from the image referential, or image frame, to the navigation referential, or ENU (East-North-Up) frame, will be described. In the equations, the distances from the camera to the gimbal, and from the gimbal to the drone are assumed negligible.

The first step is to obtain the pixel position in the camera referential, or the camera frame (P_c), from its coordinates in the image frame (P_i), which is done as follows:

$$z_c \boldsymbol{P_i} = \boldsymbol{K P_c} \;\therefore\; \frac{\mathbf{1}}{\boldsymbol{z_c}} \begin{bmatrix} \boldsymbol{x_c} \\ \boldsymbol{y_c} \\ \boldsymbol{z_c} \end{bmatrix} = \boldsymbol{K^{-1}} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \mathbf{1} \end{bmatrix}, \tag{94}$$

where u and v are the pixel coordinates in the image frame, which origin (1,1,1) is the top-left pixel of the image; x_c, y_c and z_c are the coordinates in the camera frame; and K is the camera intrinsic parameters matrix, with is calculated based on camera specifications and its construction features:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{f_x} & \boldsymbol{\gamma} & \boldsymbol{c_x} \\ \mathbf{0} & \boldsymbol{f_y} & \boldsymbol{c_y} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{95}$$

where f_x, f_y are the focal lengths of each axis in pixels unit, γ is the skew parameter, which is not used in this work and has value 0, c_x and c_y are the central image pixel coordinates (principal point offsets or optical center).

Figure 70 (1) shows the Camera and the Gimbal frames, as well as the rotation matrix for coordinates transformation of the pixel position from the Camera frame, which was calculated in the previous step, to the Gimbal frame ($R_{C \to G}$).

Figure 70 (2) shows the Gimbal frame and the NED frame, which origin is centered in the UAS. The rotation of the pixel position from the Gimbal frame to the NED frame is given by the direction cosine matrix between these frames ($R_{G \to NED}$), calculated with the known Euler angles of the Gimbal (yaw, pitch and roll).

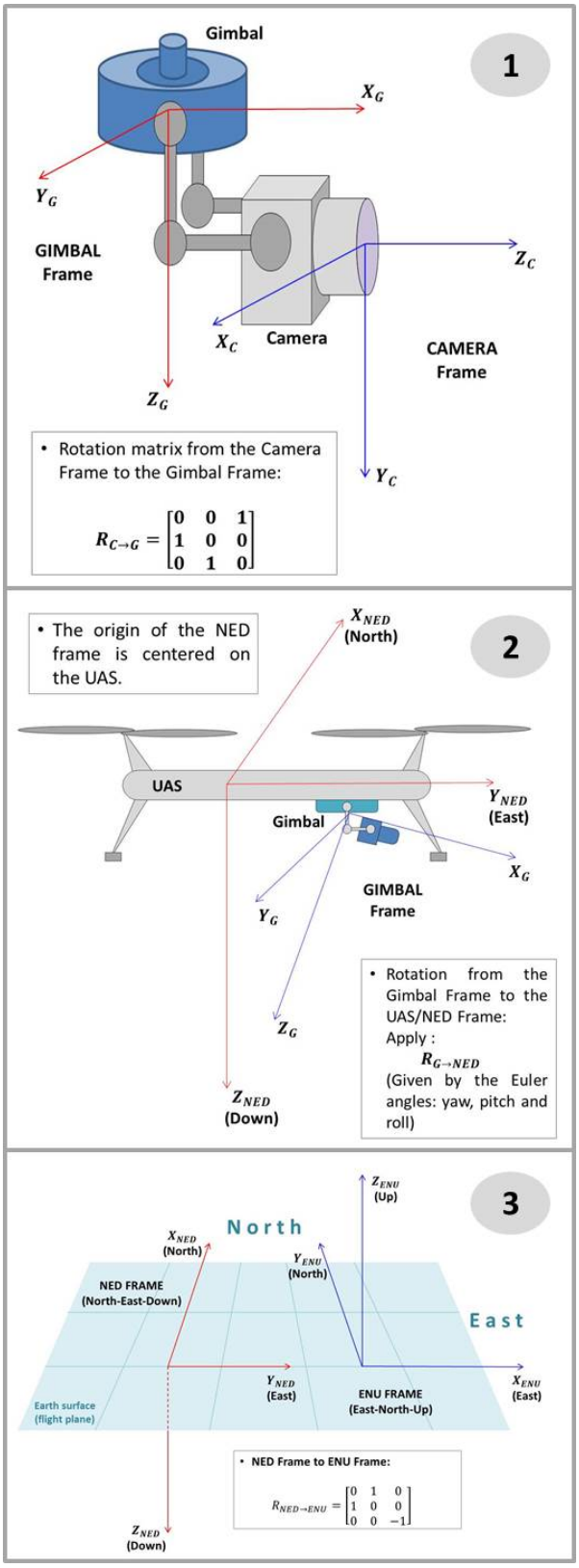


Figure 70 – Georeferencing in three steps

Finally, Figure 70 (3) shows the NED and ENU frames, and the rotation matrix between them ($R_{NED \rightarrow ENU}$).

Therefore, by employing the steps described above, it is possible to obtain the rotation matrix from the camera frame to the ENU frame ($R_{C \rightarrow ENU}$) as follows:

$$R_{C \rightarrow ENU} = R_{NED \rightarrow ENU} R_{G \rightarrow NED} R_{C \rightarrow G}. \tag{96}$$

Now it is possible to calculate the position in the world frame (P_w):

$$\boldsymbol{P_w = R_{C \rightarrow ENU} P_c + T}, \tag{97}$$

where T is the position of the camera in the world frame, given by:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \end{bmatrix}, \tag{98}$$

where X, Y and Z are the east, north and up positions of the camera in the world frame, that in this work is the Universal Transverse Mercator coordinate system.

It is important to notice that z_c is determined at the end of the calculations, by using the third element of P_w, which is the known altitude of the pixel in the world frame.

Results

Case scenario

The case scenario is a solar plant with 9200 panels displaced in an area of 55000 squared meters (Figure 67). The plant was created as a rectangular fenced area with two groups of panel piles, each one formed by eleven rows with eight piles and two rows with six piles, where each pile has two rows of 23 panels. In addition, a small building, some vegetation, a dog and a worker were included. Mountains were also placed close to the

horizon. The sky is almost clear with few clouds and the sun has 60 deg of elevation and 80 deg of azimuth. Therefore, the panels were tilted by 30 deg to the east.

The onboard camera was modeled based on the FLIR Chameleon 3 camera sensor and a lens with 12.5 mm focal length. The camera sensor size is 8.6 x 6.6 mm with a resolution of 2448 x 2024 pixels.

Mission Planning

The Field-Of-View (FOV) was calculated based on the required sensor footprint. To acquire images with a high spatial resolution, the distance from the UAS to the panels was calculated in order to have the pile width covering a major amount of the image. For the sensor used as a reference for this work, this gives a distance of 10 meters and, consequently, a horizontal FOV of 38 degrees.

After calculating the UTM coordinates of each row and determining the desired inspection path, the mission planning algorithm was used to calculate the actual mission waypoints based on the solar panels' tilt of 30 degrees and the required distance of 10 meters from the UAS to the panels.

Mission Execution

For a real case scenario equivalent to the simulated one, a complete inspection is expected to last for approximately 30 minutes. This means that the UAS takes pictures every 5 meters to cover the 9200 panels, at 3 m/s of ground speed to avoid motion blur. More than 1000 images were captured through the virtual inspection. However, the virtual inspection lasted for 45 minutes, approximately, when using a computer with an Intel i9-9900K 3600 MHz processor with a NVIDIA GeForce RTX 2080Ti graphics card and 48GB RAM.

This longer duration is due to the procedure of rendering the many high definition images, which makes the simulation very heavy and therefore slower, compelling the

simulator to pause in order to wait for the image processing. This time should be reduced if a better GPU is used or lower image resolution is set.

Concluding Remarks and Future Work

In this work, the development of a virtual reality environment for simulating inspections of solar plants with unmanned aerial systems was presented. The Unreal Engine 4 was used as the 3D graphics platform, together with the AirSim plugin for simulating the UAS physics and communication interfaces with the ArduPilot's SITL flight controller. A personalized 3D virtual solar plant was created, including the development of the solar panel models. The solution was tested in a case scenario, including the development of an autonomous guidance algorithm to capture images of the entire plant and a georeferencing algorithm to locate the defects in the real world.

As future work, more defects, such as cracks, dust, snail trails and bird drops will be modeled and integrated into the plant. In addition, the mission planner and the guidance algorithms will be improved. The mission planner needs to include sun glint and own shadow avoidance, as well as geofencing and endurance planning. This means that safe zones can be established to prevent the UAS from planning a mission that enters such zones; and that if the planned mission is longer than the endurance of the UAS, the mission can be split into shorter ones. Regarding the guidance algorithm, a Model Predictive Control solution using computer vision will be developed to use visual descriptors to adjust the position of the UAS and gimbal in real-time to acquire higher quality images and also reduce the inspection time.

Appendix C - Detection and tracking of crossing vessels for small autonomous vessels equipped with stereo camera

This appendix is an extract of the paper published in the OCEANS international conference (ANDRADE et al., 2022) with the description of the complete system, that used the direct georeferencing method presented in this thesis.

Abstract

When Small Autonomous Vessels perform a mission in a non-segregated area, that is, an area where other vessels (manned or not) will also be present, they need to be able to comply with navigation rules, such as collision avoidance. In order to decide to give-way or stand-on, the autonomous vessel must keep track of other vessels. In this work, a vessels detection and tracking solution is presented. The system uses a depth camera, IMU, magnetometer and GNSS as main sensors for this task. The boat detection is performed by the SSD-MobileNet V2 Neural Network pre-trained with the COCO dataset. The tracking is performed by a simple and fast algorithm, and the samples are later filtered using a Kalman Filter. The solution is implemented in the Robot Operating System (ROS) framework. Practical tests were performed, where stationary and moving targets were tracked. Results show that the method performs well and is able to achieve even better results after some fine tuning.

Introduction

When navigating at sea, vessels need to follow the COLREG (International Regulations for Preventing Collisions at Sea) regulations (ORGANIZATION, 2003). The rules for encountering other vessels states that one vessel must stand-on while the other vessel must give-way, depending on the position and direction of the vessels. As these rules must

be followed by any vessel, small autonomous vessels are also included.

In order to leverage the development of small autonomous vessels, Norwegian universities organize an annual competition called AUTODRONE (SOUTH-EAST NORWAY, n.d.), which, among many challenges, has a Collision Avoidance task. In this task, the vessel needs to navigate through a small canal with green and red buoys. The objective is to get through to the other side, without crashing into intersecting traffic (Figure 71).

Therefore, this project aims to implement a collision avoidance module on the university's small autonomous vessel (SeaDrone) for the competition.

As similar works, it is possible to highlight the works in (KAMSVÅG, 2018) and (HELGESEN, 2019), where the authors investigated the use of Deep Learning, sensor fusion and tracking strategies for boat detection.

Additionally, in (KUFOALOR et al., 2020), the authors expose and discuss the results of an exercise with maritime autonomous vessels for collision avoidance, in compliance with the COLREG instructions.

In (TENGESDAL; JOHANSEN; BREKKE, 2020), probabilistic data of a vessel movement was calculated in order to infer its trajectory in a generalized framework for obstacle intentions predictions. These predictions are used in a collision avoidance (COLAV) system. The Probabilistic Scenario-Based Model Predictive Control (MPC) was employed and a finite set of obstacle maneuvers were considered and its respective probability of collision were estimated.

In (HAGEN et al., 2018), a new approach in the collision avoidance problem was proposed, using a prediction model of the simulation-based MPC to model the unmanned surface vehicle (ASV) collision avoidance system as an extension of its guidance system, where the decisions of the guidance system are employed as references for the MPC COLAV method.

In (KUFOALOR; BREKKE; JOHANSEN, 2018), the authors propose a proactive, instead of a reactive, collision avoidance method for ASVs, for the Velocity Obstacles Framework, which employs relative motion arguments in order to proceed collision avoidance operations.

In (JOHANSEN; PEREZ; CRISTOFARO, 2016) MPC is used to predict a finite and low number set of obstacle maneuver scenarios for control of an autonomous COLAV system. The number of control variables are also reduced, and the model works only with the guidance course angle, commanded to the autopilot, and with propulsion command.

Despite restricting the degrees of freedom of the control, this choice achieves more effectiveness in the control of COLAV system performance, reducing its complexity and computing demand.

The novelty of this project lies in coupling a Deep Learning object detection, direct georeferencing using the depth, GNSS and IMS (Inertial Measurement System) measurements, tracking, and filtering. The solution is developed using the Robot Operating System (ROS). In addition, field tests were performed to evaluate the proposed solution.

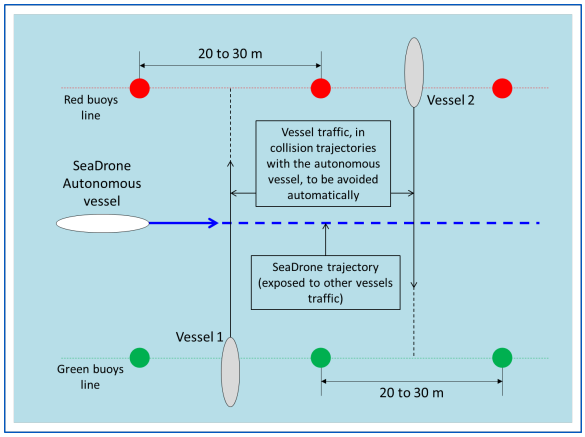


Figure 71 – Collision Avoidance Mission layout.

Development

The block diagram that summarizes the system is presented in Figure 72. First, the image is captured and the Deep Learning model identifies potential boats in the image. These detected objects are tracked by a tracking algorithm. Then, a Kalman Filter is updated for each tracked object with the new measurements. Finally, based on the filtered position and estimated velocity of the tracked objects, the decision making algorithm evaluates if the SeaDrone should give-way or stand-on.

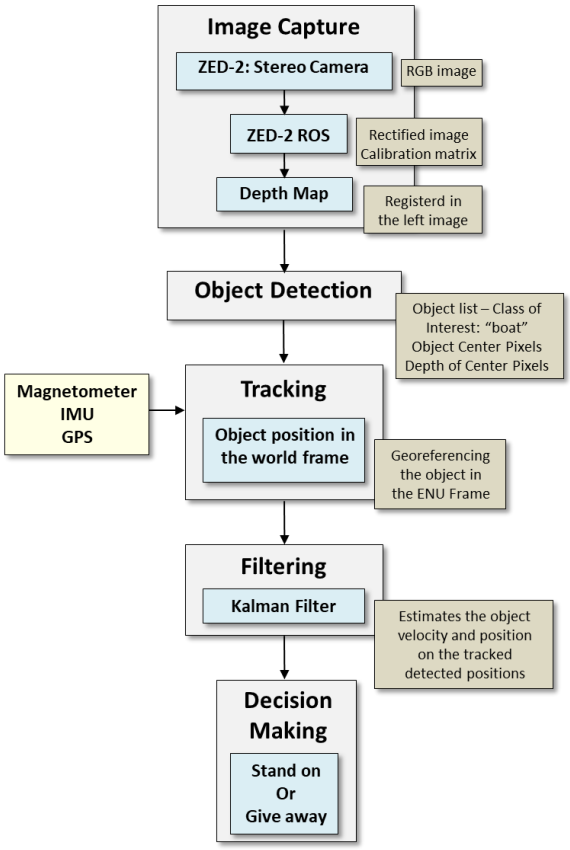


Figure 72 – System's block diagram.

Hardware

ZED 2 Stereo Camera

The Stero Lab's ZED2 camera (LABS, n.d.) is a stereo camera with neural depth perception, built-in IMU, barometer and magnetometer, it also has a full 120 °field of view. The stereo camera is absolutely essential for the SeaDrone's ability to interpret the surroundings and also for extracting depth data of crossing vessels.

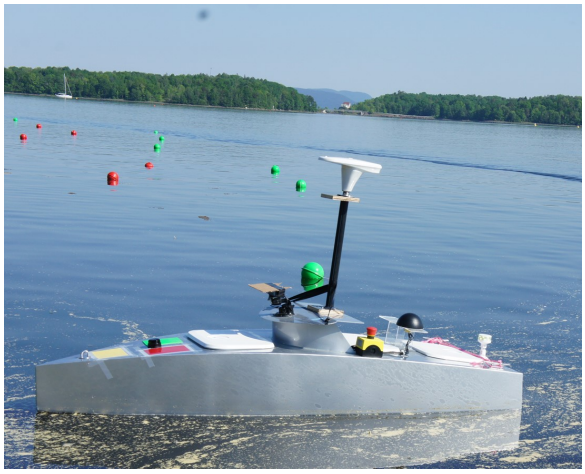


Figure 73 – SeaDrone

NVIDIA Jetson Xavier NX

The NVIDIA Jetson Xavier NX (NVIDIA, n.d.) is an advanced microcontroller that has exceptionally high performance in relation to size and power. Xavier is specifically designed to perform demanding graphical and arithmetic functions, such as the Deep Learning models used in this work. The Xavier is connected to the ZED 2 Stereo Camera through USB 3.0 and to the GNSS receiver and Raspberry Pi through Ethernet. In addition, it runs all the algorithms developed in this work.

Raspberry Pi 3 Model B+

Raspberry pi 3b+ (PI, n.d.) is a microcontroller that works as the main control unit for the navio2. The 3b+ model is equipped with 64 bits 4 core processor, dual-band 2,4GHz, bluetooth 4,2 and 5GHz wireless LAN.

Emlid Navio2 HAT

The Navio2 HAT (Hardware Attached On Top) (EMLID, n.d.) is an autopilot add-on card for Raspberry Pi with ArduPilot and ROS support. The card is equipped with GNSS receiver, IMU, Barometer and and RC I/O co-processor.

At the same time the card is equipped with GNSS receiver, RC I/O coprocessor, IMU, Barometer and has support for SBUS. [17] The Navio2 card is mounted together with the Raspberry Pi via the GPIO pins, which makes it possible to retrieve data from the GPS and compass sensors. Navio2 is also equipped with a number of PWM outputs which are used to control the subsea motors and servomotors in the Propulsion and Control System.

SIMRAD HS60 GPS Compass

The HS60 GPS Compass (SIMRAD, n.d.) provides accurate course and position data. It is ideal for applications where accurate course data is important, such as autopilot systems. It determines the course by using two separate GPS antennas, a gyro and a tilt sensor. The two last sensors are also used to provide position updates during temporary loss of GPS signal. The communication with ROS is done via a NMEA driver package.

Image Capture

The image is made available by the ZED2 stereo camera. The RGB image is taken by the left camera and rectified. ZED2 ROS Wrapper provides the rectified image as well as the camera calibration matrix for the rectified image. The depth map is registered on left image.

Object Detection

For the boat detection, the pre-trained SSD-Mobilenet-v2 object detection network was used. This network was developed by TensorFlow and was trained using the COCO 2017 dataset (LIN et al., 2014). It can detect 91 different classes, including the class ”boat”, which is the object of interest in this work. The object detection network was implemented using the NVIDIA inference toolkit.

The detection threshold was set to 50% and the center pixels of the bounding boxes of only the objects classified as ”boats” are sent to the tracking module.

Tracking

The boat detection module gives the pixel position of the center of the detected boats bounding boxes. Therefore, together with the data from the GPS, IMU, magnetometer and depth, it will be possible to calculate the position of the boat in the world frame.

In addition, more than one boat may be detected and the order of the detected boats may vary from frame to frame. Therefore, it is necessary to use a tracking strategy that is able to assign the new position to an already detected boat.

The position of the boat in the world frame in metric units is calculated from its position in pixels units in the image frame through the following rotations and translations (Figure 74), which are described in the following equations.

First, the position of the boat in the camera frame ($[x_C, y_C, z_C]^T$) is calculated based on the pixel position $[u, v, 1]^T$ (equation 99). Here, the X_C axis points to the right, the Y_C points down and the Z_C axis points straight. u and v are the horizontal and vertical pixel coordinates, where the pixel $[1, 1, 1]^T$ is in the top-left of the image. As the Z_C axis points straight, z_C is the distance between the camera and the detected boat. This information is the depth provided by the ZED2 stereo camera though the ROS Topic /zed/depth/depth_registered, which is already mapped on the left image.

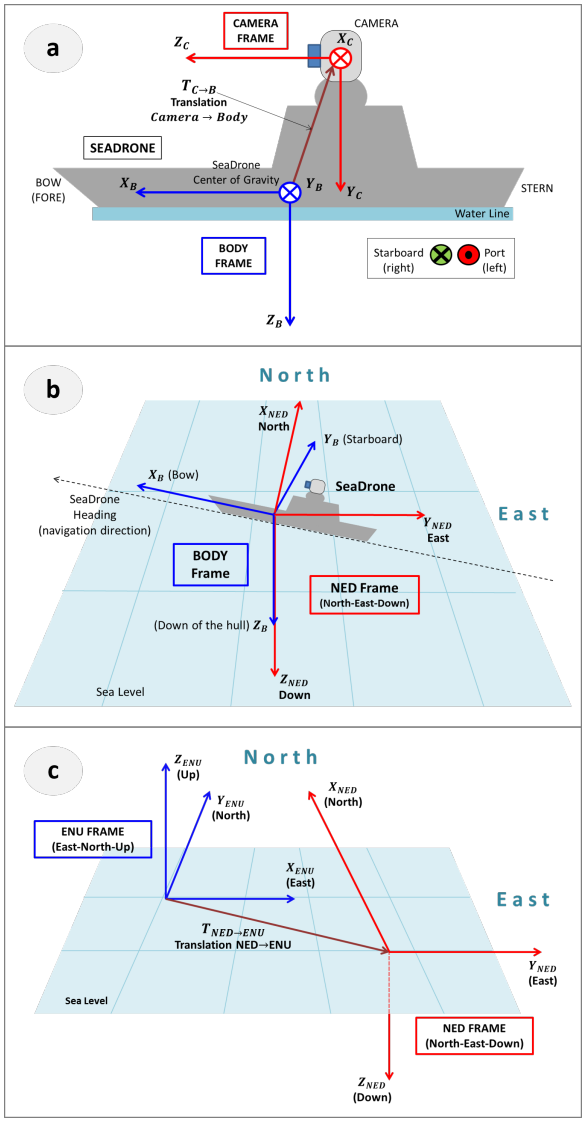


Figure 74 – Direct georeferencing steps.

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \mathbf{K}^{-1} z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{99}$$

where the intrinsic camera matrix $\mathbf{K}$ is given by:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{100}$$

The matrix parameters are made available by the ZED2 Camera ROS Wrapper

and obtained by subscribing to the ROS Topic /zed/left/camera_info.

Now, to calculate the detected boat position in the body frame, a simple rotation is needed followed by a translation:

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \mathbf{R_{C \to B}} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} + \mathbf{T_{C \to B}}, \tag{101}$$

where $\mathbf{T_{C \to B}}$ is the position of the camera in the body frame and the rotation from the camera to the body frame ($\mathbf{R_{C \to B}}$) is given by:

$$\mathbf{R_{C \to B}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{102}$$

The next step is to calculate the position in the world frame. First, a rotation to the NED frame ($\mathbf{R_{B \to NED}}$) will be done using the Euler angles: yaw (ψ), pitch (θ) and roll (ϕ). Then, the position is calculated in the ENU frame, using the same origin as the Universal Transverse Mercator (UTM) zone:

$$\begin{bmatrix} x_{ENU} \\ y_{ENU} \\ z_{ENU} \end{bmatrix} = \mathbf{R_{NED \to ENU}} \mathbf{R_{B \to NED}} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} + \mathbf{T_{NED \to ENU}}, \tag{103}$$

where $\mathbf{R_{B \to NED}}$ is the transpose of the Direction Cosine Matrix (DCM) with rotation order ZYX (yaw-pitch-roll).

In this work, a very simple tracking strategy was implemented. The process is described in the Algorithm 1 pseudo-code. Only two parameters need to be set: a maximum possible distance in meters between subsequent detected positions; and a maximum timeout in frames.

Given a list of tracked points and a list of detections, the algorithm calculates the distance from all detections to all tracked points. If the distance is shorter than the maximum distance, a detection is assigned to the closest tracked point and the timeout of the tracked point is zeroed. If a tracked point ends without any detection, one is added to the timeout. If the timeout reaches the maximum timeout the tracked point is deleted. If there is a rest of detections (e.g it is not assigned to any tracked point), a new tracked

Algorithm 1 Tracking

```
1: Define timeout_max
2: Define distance_max
3: Input detections
4: Input points
5: for point in points do
6:     for detection in detections do
7:         Calculate distance between detection and point
8:         if distance > distance_max then
9:             distances ← distance
10:        end if
11:    end for
12: end for
13: points_temp ← points
14: while distances do
15:    distance_temp ← minimum distance in distances
16:    point(distance_temp) ← detection(distance_temp)
17:    timeout(point(distance_temp)) ← 0
18:    Delete detection(distance_temp)
19:    Delete points_temp(distance_temp)
20:    Delete distance in distances
21: end while
22: for point in points_temp do
23:    Add 1 to timeout(point)
24:    if timeout(point) > timeout_max then
25:        Delete point in points
26:    end if
27: end for
28: for detection in detections do
29:    Create a new point
30:    point ← detection
31: end for
```

Figure 75 – Pseudo-code

point is created for that detection.

Filtering

A Kalman Filter was used to estimate the positions and velocities of the detected boats based on the tracked detected positions. A new Kalman Filter is initialized and assigned to each new tracked point created.

As the boats are on sea level, only the horizontal coordinates are used. Therefore,

the measurement vector $\mathbf{z}$ is equal to $[x, y]$ and the state vector $\mathbf{x}$ is equal to $[x, y, v_x, v_y]$, where x and y are the East and North positions and v_x and v_y are the East and North velocities. The observation matrix (H) is, therefore, given as:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{104}$$

and the state transition matrix (A) is given as:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \tag{105}$$

where Δt is calculated every loop based on the time between the current and the previous detection. No control input data was used in the prediction step.

Decision Making

The decision making takes place only if the distance between both vessels is shorter than a defined distance. This is to avoid making unnecessary decisions and maneuvers if the encountering does not represent any risk.

If the vessels are within the defined distance, the process goes as follows. First, to decide if the SeaDrone should stand-on or give-way, the bearing between the SeaDrone and the detected boat is compared with the heading of the SeaDrone. Therefore, if the detected boat is on the left side, the SeaDrone should stand-on, except if the distance between both vessels are shorter than a defined safe distance. In this case, the SeaDrone stops regardless.

If the detected boat is on the left side of the SeaDrone, the heading of both vessels are compared. Therefore, if the detected boat is moving away, the SeaDrone stands-on. If the vessel's heading is such as it will likely cross the SeaDrone course, the SeaDrone gives-way.

Results

Field tests were performed to evaluate the proposed solution. First, a stationary boat was detected and tracked when the SeaDrone was moving on the direction of the target boat. As it can be seen in Figure 76, the detected stationary boat's position (gray) has a deviation of around 1.5 meters in the x axis and 1 meter in the y axis. The reasons for this deviation may be the fact that the detection returns the center pixel of the bounding box of the target boat and as the SeaDrone is moving, this center pixel may vary if the target boat is being spotted from different angles. Other reasons may be sensor reading errors, for example, the depth, IMU or even the GNSS, as no RTK solution is used.

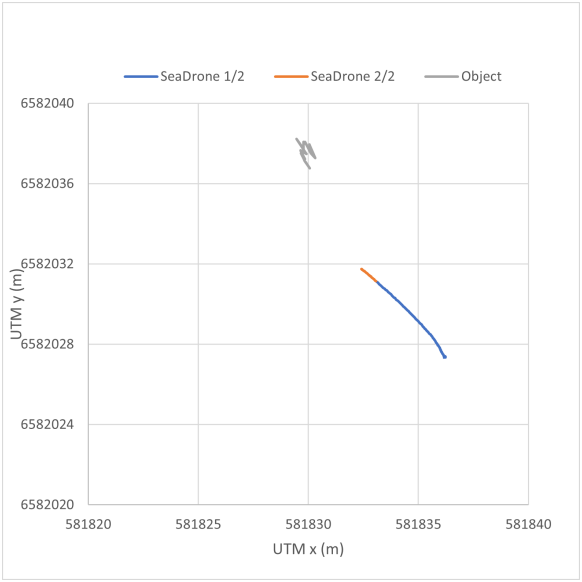


Figure 76 – SeaDrone moving straight tracking a steady boat

In Figure 76, the trajectory of the SeaDrone was divided into two colors, the blue trajectory is before the target boat was detected, and the orange trajectory is during the detection and tracking. In Figure 77, the heading of the SeaDrone is presented, only from the moment where the target boat is detected, until the end of the test.

The objective of the second test was to investigate how the system performed when rotation around the steady boat (Figure 78). Therefore, a variation of 150 degrees of the heading was observed (Figure 79 during the test.

It is possible to notice that the there is a sudden jump for the position of the tracked boat. Again, this is due to the fact that the boat is big, so when the SeaDrone started the

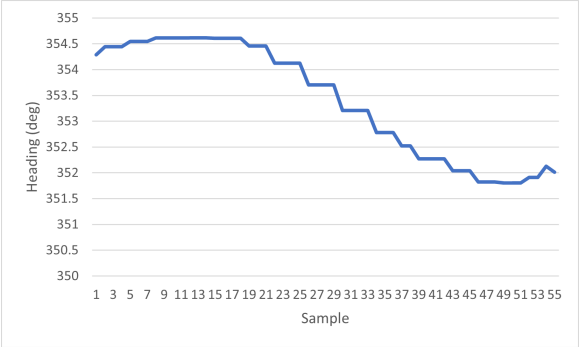


Figure 77 – SeaDrone heading during the tracking

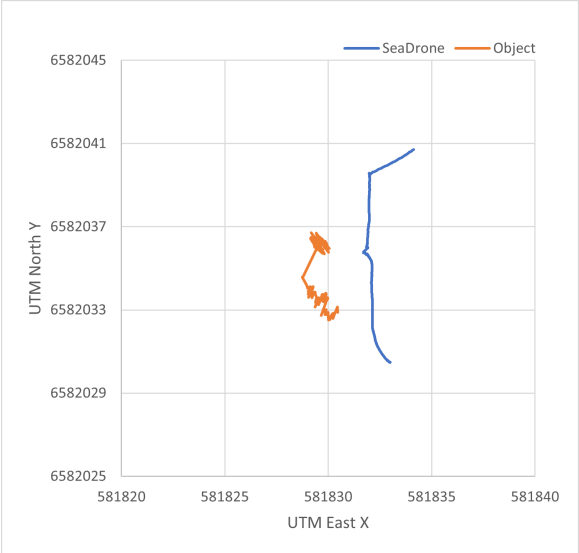


Figure 78 – SeaDrone rotating around the tracked boat

tracking, it was spotting one side of the boat and, after a while, it began to spot the other side of the boat. An easy solution to avoid problems on the collision avoidance is to always have a safe collision avoidance distance that is bigger than the maximum expected length of a crossing vessel. If very large vessels are expected, the SeaDrone should be able to re-evaluate the obstacles during the waypoint navigation.

Finally, a last test was performed, when the SeaDrone was moving and tracking multiple targets that were also moving (Figure 80). It is possible to see that the the objects 5 (red) and 6 (dark gray) are most likely the same target and that the tracking module assigned a new number tag to it in the tracking process. The same can be said about the object 1 (gray) and 2 (yellow). Probably the SeaDrone started tracking the object and because it started turning, it lost the object from the camera view and then assigned a new number tag when it was detected again.

Regarding the heading (Figure 81), it is possible to notice that the SeaDrone

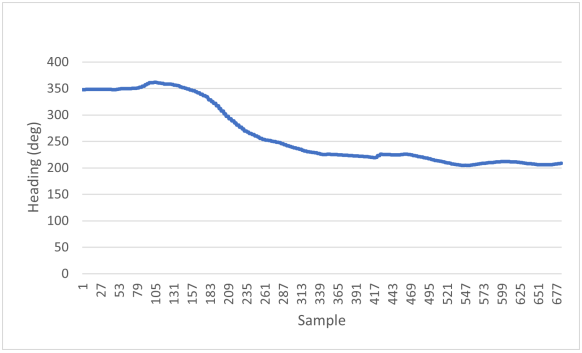


Figure 79 – SeaDrone heading when rotating around the tracked boat

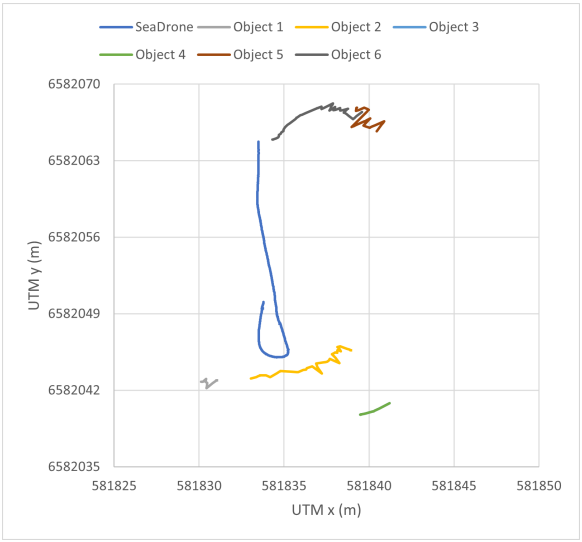


Figure 80 – SeaDrone moving when tracking multiple moving humans

starts moving south, with around 170 degrees of heading, then turns to north direction. The heading does not reach zero degrees (north) probably because the SeaDrone has the course of zero degrees but the heading is around 40 degrees. This means that the SeaDrone is moving sideways.

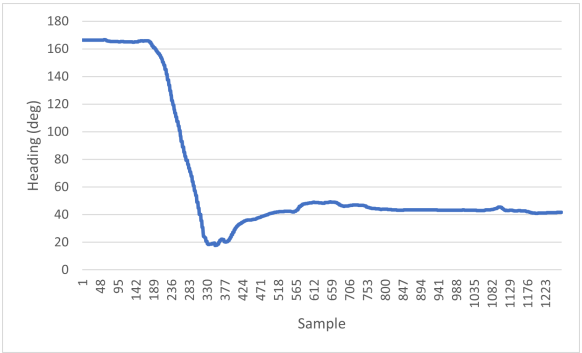


Figure 81 – SeaDrone heading when tracking multiple moving humans

Conclusion

In this work, a solution for detection and tracking of boats for collision avoidance was presented. The system is equipped with a depth camera, IMU, magnetometer and GNSS sensors. The solution was developed in the Robot Operating System (ROS) framework. Results show that the system is able to detect and track the targets. The tracking and filtering work as expected. As future work, further testing and tuning will be performed to achieve better results. In addition, the solution will be completed using Model Predictive Control to follow a trajectory that avoids the intersecting traffic.

References

ADOBE INC. **Adobe Photoshop**. [S.l.: s.n.].

_____. **Substance Painter — Substance 3D**. [S.l.: s.n.].

ALCANTARA ANDRADE, Fabio Augusto de et al. Path Planning of Multi-UAS Communication Relay by Decentralized MPC. In: IEEE. 2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO). [S.l.: s.n.], 2018. p. 1–8.

ALOTAIBI, Ebtehal Turki; ALQEFARI, Shahad Saleh; KOUBAA, Anis. LSAR: Multi-UAV Collaboration for Search and Rescue Missions. **IEEE Access**, IEEE, v. 7, p. 55817–55832, 2019.

AMER, Karim et al. Deep convolutional neural network-based autonomous drone navigation. **arXiv preprint arXiv:1905.01657**, 2019.

ANDERSON, John David. **Aircraft performance & design**. [S.l.]: McGraw-Hill Science Engineering, 1999.

ANDRADE, Fabio AA et al. Detection and tracking of crossing vessels for small autonomous vessels equipped with stereo camera. In: IEEE. OCEANS 2022-Chennai. [S.l.: s.n.], 2022. p. 1–6.

ANDRADE, Fabio Augusto de Alcantara. Real-time and offline path planning of Unmanned Aerial Vehicles for maritime and coastal applications. NTNU, 2019.

ARDUPILOT. [S.l.: s.n.], 2019. `ardupilot.org`. [Online; accessed 10-July-2019].

BARBER, D. B. et al. Vision-based Target Geo-location using a Fixed-wing Miniature Air Vehicle. **Journal of Intelligent and Robotic Systems**, v. 47, p. 361–382, 2006.

BEARD, Randal W; MCLAIN, Timothy W. **Small unmanned aircraft: Theory and practice**. [S.l.]: Princeton university press, 2012.

BERNDT, Jon. JSBSim: An open source flight dynamics model in C++. In: AIAA Modeling and Simulation Technologies Conference and Exhibit. [S.l.: s.n.], 2004. p. 4923.

BONDI, Elizabeth et al. Near Real-Time Detection of Poachers from Drones in AirSim. In: IJCAI. [S.l.: s.n.], 2018. p. 5814–5816.

BREIVIK, Oyvind et al. Wind-induced drift of objects at sea: The leeway field method. **Applied Ocean Research**, Elsevier, v. 33, n. 2, p. 100–109, 2011.

CAMACHO, Eduardo F et al. Model predictive control techniques for hybrid systems. **Annual reviews in control**, Elsevier, v. 34, n. 1, p. 21–31, 2010.

ÇETIN, Ender; BARRADO, Cristina; PASTOR, Enric. Counter a Drone in a Complex Neighborhood Area by Deep Reinforcement Learning. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 8, p. 2320, 2020.

CHAO, Zhou et al. UAV formation flight based on nonlinear model predictive control. **Mathematical Problems in Engineering**, Hindawi, v. 2012, 2012.

CHRISTIE, Gordon et al. Training object detectors with synthetic data for autonomous uav sampling applications. In: IEEE. 2018 International Conference on Unmanned Aircraft Systems (ICUAS). [S.l.: s.n.], 2018. p. 352–357.

COMMUNITY, Blender Online. **Blender - a 3D modelling and rendering package**. Stichting Blender Foundation, Amsterdam, 2018.

CORREIA, Carlos AM et al. Comprehensive Direct Georeferencing of Aerial Images for Unmanned Aerial Systems Applications. **Sensors**, MDPI, v. 22, n. 2, p. 604, 2022.

CURTIS, Howard D. **Orbital mechanics for engineering students**. [S.l.]: Butterworth-Heinemann, 2013.

DE ALCANTARA ANDRADE, Fabio Augusto et al. Virtual Reality Simulation of Autonomous Solar Plants Inspections with Unmanned Aerial Systems. In: 2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO). [S.l.: s.n.], 2021. p. 1–8. DOI: 10.1109/AIRPHARO52252.2021.9571060.

DI, Bin; ZHOU, Rui; DUAN, Haibin. Potential field based receding horizon motion planning for centrality-aware multiple UAV cooperative surveillance. **Aerospace Science and Technology**, Elsevier, v. 46, p. 386–397, 2015.

DOD, US. Department of Defense dictionary of military and associated terms. **Joint Publication**, p. 1–02, 2007.

EBERHART, Russell; KENNEDY, James. A new optimizer using particle swarm theory. In: IEEE. MICRO Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. [S.l.: s.n.], 1995. p. 39–43.

EMLID. **NAVIO2**. Available from: ¡https : / / navio2 . emlid . com/¿. (accessed: 19.07.2021).

FARROKHSIAR, Morteza; PAVLIK, Graham; NAJJARAN, Homayoun. An integrated robust probing motion planning and control scheme: A tube-based MPC approach. **Robotics and Autonomous Systems**, Elsevier, v. 61, n. 12, p. 1379–1391, 2013.

FLIR. **Chameleon3 USB3 — Teledyne FLIR**. [S.l.]. Available at https://www.flir.com.br/products/chameleon3-usb3/ (accessed: 23.12.2021).

FLIR. **A guide to inspecting solar fields with thermal imaging drones**. [S.l.: s.n.], 2020. flir.eu. [Online; accessed 26-May-2021].

FORSMO, Erik J et al. Optimal search mission with unmanned aerial vehicles using mixed integer linear programming. In: IEEE. 2013 International conference on unmanned aircraft systems (ICUAS). [S.l.: s.n.], 2013. p. 253–259.

GABRIELY, Yoav; RIMON, Elon. Spanning-tree based coverage of continuous areas by a mobile robot. **Annals of mathematics and artificial intelligence**, Springer, v. 31, n. 1-4, p. 77–98, 2001.

GRYTE, Kristofer et al. Aerodynamic modeling of the skywalker x8 fixed-wing unmanned aerial vehicle. In: IEEE. 2018 International Conference on Unmanned Aircraft Systems (ICUAS). [S.l.: s.n.], 2018. p. 826–835.

HAGEN, Inger Berge et al. MPC-based collision avoidance strategy for existing marine vessel guidance systems. In: IEEE. 2018 IEEE International Conference on Robotics and Automation (ICRA). [S.l.: s.n.], 2018. p. 7618–7623.

HALS, Martin; SKJONHAUG, Martin. **Optimization of Coordinated Path Planning for Autonomous Vehicles in Ice Management**. 2017. MA thesis – NTNU.

HANNING, Tobias. **High precision camera calibration**. [S.l.]: Springer, 2011.

HARTLEY, Richard; ZISSERMAN, Andrew. **Multiple View Geometry in Computer Vision**. [S.l.]: Cambridge University Press, 2004.

HELGESEN, Håkon Hagen et al. Camera-Based Tracking of Floating Objects using Fixed-wing UAVs. **Journal of Intelligent & Robotic Systems**, Springer, v. 102, n. 4, p. 1–16, 2021.

HELGESEN, Håkon Hagen et al. Real-time georeferencing of thermal images using small fixed-wing UAVs in maritime environments. **ISPRS Journal of Photogrammetry and Remote Sensing**, Elsevier, v. 154, p. 84–97, 2019.

HELGESEN, Øystein Kaarstad. Sensor fusion for detection and tracking of maritime vessels. **mastersthesis, NTNU, Trondheim**, 2019.

HEMERLY, Elder M. Automatic georeferencing of images acquired by UAV's. **International Journal of Automation and Computing**, Springer, v. 11, n. 4, p. 347–352, 2014.

HENRY, Chris et al. Automatic Detection System of Deteriorated PV Modules Using Drone with Thermal Camera. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 10, n. 11, p. 3802, 2020.

HENTATI, Aicha Idriss et al. Simulation tools, environments and frameworks for UAV systems performance analysis. In: IEEE. 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). [S.l.: s.n.], 2018. p. 1495–1500.

HO, Chia Yu et al. A parameter sharing method for reinforcement learning model between AirSim and UAVs. In: IEEE. 2018 1st International Cognitive Cities Conference (IC3). [S.l.: s.n.], 2018. p. 20–23.

HONKAVAARA, Eija et al. Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight UAV spectral camera for precision agriculture. **Remote Sensing**, Multidisciplinary Digital Publishing Institute, v. 5, n. 10, p. 5006–5039, 2013.

HOY, Michael; MATVEEV, Alexey S; SAVKIN, Andrey V. Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments. **Robotics and Autonomous Systems**, Elsevier, v. 60, n. 10, p. 1253–1266, 2012.

IAMSAR, II. International aeronautical and maritime search and rescue manual. **Mission coordination**, v. 2, 2007.

JAKOB, Michal et al. Occlusion-aware multi-uav surveillance of multiple urban areas. In: CITESEER. 6TH Workshop on Agents in Traffic and Transportation (ATT 2010). [S.l.: s.n.], 2010. p. 59–66.

JOHANSEN, Tor Arne; PEREZ, Tristan; CRISTOFARO, Andrea. Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment. **IEEE transactions on intelligent transportation systems**, IEEE, v. 17, n. 12, p. 3407–3422, 2016.

KAMSVÅG, Vegard. Fusion between camera and lidar for autonomous surface vehicles. **mastersthesis, NTNU**, 2018.

KOBAYASHI, Audrey. **International Encyclopedia of Human Geography**. [S.l.]: Elsevier, 2019.

KUFOALOR, D Kwame Minde; BREKKE, Edmund Førland; JOHANSEN, Tor Arne. Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method. In: IEEE. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S.l.: s.n.], 2018. p. 2402–2409.

KUFOALOR, D Kwame Minde et al. Autonomous maritime collision avoidance: Field verification of autonomous surface vehicle behavior in challenging scenarios. **Journal of Field Robotics**, Wiley Online Library, v. 37, n. 3, p. 387–403, 2020.

LABS, STEREO. **ZED 2**. [S.l.]. Available at `https://www.stereolabs.com/zed-2/` (accessed: 19.07.2021). (accessed: 19.07.2021).

LEIRA, Frederik S et al. A ligth-weight thermal camera payload with georeferencing capabilities for small fixed-wing UAVs. In: IEEE. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). [S.l.: s.n.], 2015. p. 485–494.

LI, Mengxiang; LAVEST, J-M. Some aspects of zoom lens camera calibration. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 18, n. 11, p. 1105–1110, 1996.

LIN, Tsung-Yi et al. Microsoft coco: Common objects in context. In: SPRINGER. EUROPEAN conference on computer vision. [S.l.: s.n.], 2014. p. 740–755.

LUNA, Taryn. New technology making drones easier, more affordable. **The Boston Globe**, 8 Dec. 2015.

MARCU, Alina et al. SafeUAV: Learning to estimate depth and safe landing areas for UAVs from synthetic data. In: PROCEEDINGS of the European Conference on Computer Vision (ECCV) Workshops. [S.l.: s.n.], 2018. p. 0–0.

MAVLINK Micro Air Vehicle Protocol. [S.l.: s.n.], 2019. `mavlink.io/en`. [Online; accessed 10-July-2019].

NAVPY. **Welcome to NavPy's documentation! — NavPy 0.1 documentation**. [S.l.]. Available at `http://navpy.readthedocs.io` (accessed: 06.09.2021).

NIGAM, Nikhil. The multiple unmanned air vehicle persistent surveillance problem: A review. **Machines**, Multidisciplinary Digital Publishing Institute, v. 2, n. 1, p. 13–72, 2014.

NOVAK, Matt. U.S. Border Patrol Flew More Drone Missions Last Year Than Ever Before. **GIZMODO**, 26 Sept. 2018.

NVIDIA. **JETSON XAVIER NX**. Available from: ¡https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/¿. (accessed: 19.07.2021).

OLIPHANT, Travis E. **A guide to NumPy**. [S.l.]: Trelgol Publishing USA, 2006. v. 1.

ORGANIZATION, International Maritime. **COLREG: Convention on the International Regulations for Preventing Collisions at Sea, 1972**. [S.l.]: IMO, 2003.

PI, Raspberry. **Raspberry Pi 3 Model B+**. Available from: ¡https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/¿. (accessed: 19.07.2021).

PINTO, Jose et al. The LSTS toolchain for networked vehicle systems. In: IEEE. 2013 MTS/IEEE OCEANS-Bergen. [S.l.: s.n.], 2013. p. 1–9.

QGIS DEVELOPMENT TEAM. **QGIS Geographic Information System**. [S.l.], 2009.

REDULLA, Anne; SINGH, Surya PN. Simulating differential games with improved fidelity to better inform cooperative & adversarial two vehicle UAV flight. In: IEEE. 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR). [S.l.: s.n.], 2018. p. 130–136.

REES, Eric van. A Quick Introduction to Drone Photogrammetry. **medium.com**, 20 Mar. 2019.

RODIN, Christopher Dahlin et al. Object Classification in Thermal Images using Convolutional Neural Networks for Search and Rescue Missions with Unmanned Aerial Systems. In: IEEE. 2018 International Joint Conference on Neural Networks (IJCNN). [S.l.: s.n.], 2018. p. 1–8.

SAN JUAN, Victor; SANTOS, Matilde; ANDUJAR, Jose Manuel. Intelligent UAV map generation and discrete path planning for search and rescue operations. **Complexity**, Hindawi, v. 2018, 2018.

SENGUPTA, Raja et al. **Final report - autonomous search and rescue with ScanEagle**. [S.l.: s.n.], 2010.

SHAH, Shital et al. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: FIELD and Service Robotics. [S.l.: s.n.], 2017. eprint: arXiv:1705.05065.

SIMRAD. **Simrad HS60 GPS Compass — Sensor — Simrad Norge**. Available from: ¡https://https://www.simrad-yachting.com/nb-no/simrad/type/kompass/hs60-gps-compass/¿. (accessed: 10.12.2021).

SISK, Richard. Coast Guard Commandant Boosts Plan to Outfit Cutters with ScanEagle Drones. **Military.com**, 10 Dec. 2018.

SKALOUD, Jan. **Optimizing georeferencing of airborne survey systems by INS/DGPS**. [S.l.]: University of Calgary, 1999.

SKJONG, Espen et al. Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and avionics. In: IEEE. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). [S.l.: s.n.], 2015. p. 904–913.

SOUTH-EAST NORWAY, University of. **AUTODRONE**. Available from: ¡https://www.autodrone.no/¿. (accessed: 09.12.2021).

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY ET AL. **Robotic Operating System**. [S.l.: s.n.], 23 May 2018.

TALAY, Theodore A. **Introduction to the Aerodynamics of Flight**. [S.l.]: Scientific, Technical Information Office, National Aeronautics, and Space . . ., 1975. v. 367.

TENGESDAL, Trym; JOHANSEN, Tor Arne; BREKKE, Edmund. Risk-based Autonomous maritime collision avoidance considering obstacle intentions. In: IEEE. 2020 IEEE 23rd International Conference on Information Fusion (FUSION). [S.l.: s.n.], 2020. p. 1–8.

TRODDEN, Paul; RICHARDS, Arthur. Multi-vehicle cooperative search using distributed model predictive control. In: AIAA Guidance, Navigation and Control Conference and Exhibit. [S.l.: s.n.], 2008. p. 7138.

WANG, Shubo et al. Construction of a virtual reality platform for UAV deep learning. In: IEEE. 2017 Chinese Automation Congress (CAC). [S.l.: s.n.], 2017. p. 3912–3916.

WU, Jianfa et al. Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm. **Aerospace Science and Technology**, Elsevier, v. 70, p. 497–510, 2017.

WU, Jianfa et al. Energy management strategy for solar-powered UAV long-endurance target tracking. **IEEE Transactions on Aerospace and Electronic Systems**, IEEE, 2018.

XIANG, Haitao; TIAN, L. Method for automatic georeferencing aerial remote sensing (RS) images from an unmanned aerial vehicle (UAV) platform. **Biosystems Engineering**, v. 108, p. 104–113, 2011.

YAO, Shouwen et al. Autonomous-driving vehicle test technology based on virtual reality. **The Journal of Engineering**, IET, v. 2018, n. 16, p. 1768–1771, 2018.

ZHOU, Gouqing; LI, Chaokui; CHENG, P. Unmanned aerial vehicle (UAV) real-time video registration for forest fire monitoring. **Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.**, v. 3, p. 1803–1806, 2005.

ZOLICH, Artur et al. Survey on communication and networks for autonomous marine systems. **Journal of Intelligent & Robotic Systems**, Springer, p. 1–25, 2018.

ZOLICH, Artur et al. Unmanned aerial system architecture for maritime missions. design & hardware description. In: IEEE. 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS). [S.l.: s.n.], 2015. p. 342–350.