A Novel Path Planning Optimization Algorithm for Semi-Autonomous UAV in Bird Repellent Systems Based in Particle Swarm Optimization

Ricardo Jorge Mendes Mesquita

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar

julho de 2021

Acknowledgements

This research work is funded by the PrunusBot project - Autonomous controlled spraying aerial robotic system and fruit production forecast, Operation No. PDR2020-101-031358 (leader), Consortium No. 340, Initiative No. 140, promoted by PDR2020 and co-financed by the EAFRD and the European Union under the Portugal 2020 program.

The authors thank the opportunity and financial support to carry on this project to Fundação para a Ciência e Tecnologia (FCT) and R&D Unit "Centre for Mechanical and Aerospace Science and Technologies" (C-MAST), under project UIDB/00151/2020.

Resumo

Anualmente, os danos causados pelas aves em pomares criam perdas monetárias significativas aos agricultores. A aplicação de métodos tradicionais de dispersão de aves, como canhões repelentes de aves e redes nas árvores, torna-se ineficiente a longo prazo, sendo ainda de alta manutenção e de mobilidade reduzida. Devido à sua versatilidade, os Veículos Aéreos Não Tripulados (VANT) podem ser benéficos para resolver este problema. No entanto, devido à baixa capacidade das suas baterias, que se traduz num baixo tempo de voo, é necessário otimizar o planeamento dos caminhos.

Nesta dissertação, é apresentado um algoritmo de otimização para planeamento de caminhos para VANT baseado no Particle Swarm Optimization (PSO). Para se iniciarem os primeiros testes do algoritmo proposto, a técnica utilizada foi a supracitada devido à necessidade de um algoritmo de otimização fácil de implementar. O algoritmo PSO é simples e possuí poucos parâmetros de controlo, mantendo um bom desempenho. Este algoritmo de otimização de planeamento de caminhos propõe-se a gerir a distância e o tempo de voo do drone, aplicando técnicas de otimização e de aleatoriedade para superar a sua desvantagem relativamente aos sistemas tradicionais. O desempenho do algoritmo de planeamento de caminhos foi testado em três casos de estudo: dois deles em simulação para testar a variação de cada parâmetro e outro em campo para testar a capacidade da bateria. Todos os casos foram testados nas três situações possíveis: mesma taxa de incidência, taxas diferentes e taxas diferentes sem danos de aves.

Os resultados apresentados pelo algoritmo proposto demonstram um erro médio muto reduzido na distância total para o planeamento de caminhos obtido e baixo tempo de execução. Porém, é necessário destacar que o algoritmo pode ter dificuldade em encontrar uma solução adequada se houver uma má relação entre a distância total para o planeamento de caminhos e os pontos de interesse. Os testes de campo também foram essenciais para entender o comportamento do algoritmo na prática, mostrando que há menos energia consumida com menos pontos de interesse, sendo que este parâmetro não se correlaciona com o tempo de voo. Além disso, não há associação entre a velocidade horizontal máxima e o tempo da missão, o que significa que a função de cálculo da distância total para o planeamento de caminhos requer ser ajustada.

Palavras-chave

Danos a Pássaros; Agricultura de Precisão; Veículos Aéreos Não Tripulados; Planeamento de Caminho; Meta-heurística; Algoritmo de Otimização para Planeamento de Caminhos

Abstract

Bird damage to fruit crops causes significant monetary losses to farmers annually. The application of traditional bird repelling methods such as bird cannons and tree netting became inefficient in the long run, keeping high maintenance and reduced mobility. Due to their versatility, Unmanned Aerial Vehicles (UAVs) can be beneficial to solve this problem. However, due to their low battery capacity that equals low flight duration, it is necessary to evolve path planning optimization.

A path planning optimization algorithm of UAVs based on Particle Swarm Optimization (PSO) is presented in this dissertation. This technique was used due to the need for an easy implementation optimization algorithm to start the initial tests. The PSO algorithm is simple and has few control parameters while maintaining a good performance. This path planning optimization algorithm aims to manage the drone's distance and flight time, applying optimization and randomness techniques to overcome the disadvantages of the traditional systems. The proposed algorithm's performance was tested in three study cases: two of them in simulation to test the variation of each parameter and one in the field to test the influence on battery management and height influence. All cases were tested in the three possible situations: same incidence rate, different rates, and different rates with no bird damage to fruit crops.

The proposed algorithm presents promising results with an outstanding reduced average error in the total distance for the path planning obtained and low execution time. However, it is necessary to point out that the path planning optimization algorithm may have difficulty finding a suitable solution if there is a bad ratio between the total distance for path planning and points of interest. The field tests were also essential to understand the algorithm's behavior of the path planning algorithm in the UAV, showing that there is less energy discharged with fewer points of interest, but that do not correlates with the flight time. Also, there is no association between the maximum horizontal speed and the flight time, which means that the function to calculate the total distance for path planning needs to be adjusted.

Keywords

Bird Damage to Fruit Crops; Precision Agriculture; Unmanned Aerial Vehicles; Path Planning; Meta-heuristic; Path Planning Optimization Algorithm

Index

List of Figures

List of Tables

List of Acronyms

UAVs	Unmanned Aerial Vehicles
VTOL	Vertical Take-Off and Landing
ESC	Electronic Speed Control
INS	Inertial Navigation System
LiDAR	Light Detection and Ranging
GCS	Ground Control Station
FAAS	Fully Autonomous Aerial Systems
AI	Artificial Intelligence
GNSS	Global Navigation Satellite System
ARM	Advanced Reduced Instruction Set Computer Machine
RPi	Raspberry Pi (RPi)
SITL	Software-in-the-loop
RAM	Random Access Memory
DC	Direct Current
LED	Light-emitting Diode
UART	Universal Asynchronous Receiver-Transmitter
PWM	Pulse-Width Modulation
LCD	Liquid Crystal Display
SPI	Serial Peripheral Interface
I2C	Inter-integrated Circuit
ADC	Analog to Digital Converter
CAN Bus	Controller Area Network
USB	Universal Serial Bus
PPM	Pulse-position Modulation
RSSI	Received Signal Strength Indication
RC	Radio Control
GPS	Global Positioning System
RGB	Red, Green and Blue
CPPM	Chaotic Pulse-position Modulation
FRAM	Ferroelectric RAM
SD	Secure Digital
CPU	Central Processing Unit
HDMI	High-Definition Multimedia Interface
OSD	On-screen Display
OS	Operating System
GPIO	General-purpose Input/Output
GPU	Graphic Processor Unit
HD	High Definition
PSO	Particle Swarm Optimization
GWO	Grey Wolf Optimizer
IEEE	Institute of Electrical and Electronics Engineers
FA	Flower Algorithm
PV	Photo-voltaic
MPP	Maximum Power Point

BI	Bat-inspired
CI	Computational Intelligence
GA	Genetic Algorithm
ACO	Ant Colony Optimization
ANN	Artificial Neural Networks
LB	Learning-based Methods
FL	Fuzzy Logic Algorithms
Li-Po	Lithium Polymer
CH	Chanel
RF	Radio Frequency
CAD	Computer-aided Design
FDM	Fused Deposition Modeling
PWM	Pulse-width Modulation
PoI	Points of Interest
GDP	Gross Domestic Product

1. Introduction

1.1. Contextualization

Agriculture is not only a source of food but also a source of vast employment and rural development. However, the contribution of agriculture to national economies has decreased over the years, as countries have moved upward to upper-income classes. Still, 26.5% of the world's total employment is in this sector. Increased productivity contributes to lowering food prices, which will benefit the consumers, particularly the low income since food expenses represent a large share of their total budget. Hence, its development and growth have always been and will remain one of the topmost priority agendas of policymakers [1], companies, and researchers.

Precision Agriculture is an old concept being mentioned in 1999 as applying technologies and principles to manage spatial and temporal variability associated with all aspects of agricultural production to improve crop performance and environmental quality [2]. In essence, this concept represents the importance of introducing new technologies and techniques to optimize the farming process, increase quality, and reduce production prices for both growers and consumers. Among these new technologies, Unmanned Aerial Vehicles (UAVs), commonly known as drones, have been increasingly used in agricultural activities. UAVs are unmanned aircrafts that are already being applied in agriculture and used to overcome traditional systems' failures. Several parameters characterize its types, such as the structure, method to depart and land, and the number of motors. Horizontal Take-Off and Landing, multirotor, helicopter, and Vertical Take-Off and Landing (VTOL) are the main configurations. Its basic architecture consists of a frame, brush-less motors, Electronic Speed Control (ESC) modules, control board, Inertial Navigation System (INS), and a transmitter/receiver module [3]. Depending on its function, these may include cargo compartments, actuators, and sensors such as Light Detection and Ranging (LiDAR) modules and multispectral cameras. The possibility of systematic data collection, mapping field variability, and better decisions lead farmers and companies to invest in UAVs for agriculture. These are used in early soil analysis to acquire image-based data helpful in irrigation and ground-level management by determining the strength of nutrients. Appling hyperspectral, multispectral, thermal, or LiDAR sensors mounted to the UAVs, farmers can identify which parts of crops need improvements and react appropriately on time [4]. Another good example is the commercially available DJI AGRAS T20 shown in Figure 1. This UAV system can autonomously spread seeds, fertilizer, and other solid materials, using a 360º digital radar to avoid obstacles and a P4 multispectral camera, gathering precise plant-level data, which improves general efficiency in all processes [5].

Figure 1 – DJI AGRAS T20 [5].

Bird damage to fruit and other horticultural crops is a well-documented agricultural problem [6]-[7], due to being costly and persistent to farmers worldwide. Flocks destroy trees and feed on fruits and grains. In addition to consumption, diseases can appear, leading to decreased product quality and quantity [8]. Bird damage to orchards is a long-term problem. In 1974, Clark [9] addressed the issue, making an overview of the control techniques commonly used back then in California for each bird. With the advancement of technology, new techniques have emerged to reduce the environmental impact, such as chemical pollutants and lethal systems. Bird cannons usually use propane to randomly imitate a loud sound like an explosion, consisting of a support, control system, feeding cylinder, and a cone to disperse the sound, making this heavy, low mobility, and predictable. These systems are the most popular of numerous mechanical, visual, and auditory methods used for scaring birds away from crop fields [10]. Another popular method is speakers who use various species-specific distress signals and predator calls to send a danger alert to birds in the area. Despite its small size, this method presents low mobility because it is necessary to move it not to become predictable. An alternative to these active and mechanical methods are cultural methods such as protecting crops with netting, but this only works on a small scale and may divert birds to crops with no netting present or planting and harvest date manipulation but is not always possible and required knowledge of bird movement [11].
The central region of the Portugal countryside has good climatic and edaphic conditions for fruit species production, mainly Prunus fruit, specifically cherry and peach trees, with Beira Interior being the leading grower of this type of fruit in the Country [12]. In this area, flocks of birds, such as shown in Figure 2, find in these orchards' food and shelter, destroying fruits and trees, creating a financial and management problem since these are intelligent animals that can detect patterns, are highly mobile and persist after a food source is discovered [13].

Affecting producers worldwide, in the U.S.A., farmers lose tens of millions of dollars each year through direct losses and often ineffective efforts to deter birds [14].

Figure 2 – Flock of birds in Soalheira, Fundão, Portugal.

Due to this current problem for national and international farmers', modern technologies emerge to minimize it. Drones merged with repelling systems become tools to solve bird damage to orchards problem. Wang et al. [15] studied the impact of these technologies using bird taxidermy and loud distress calls in ravens, starlings, and cockatoos. The results strongly indicated that the UAV is an effective bird deterrent for the target species and showed that several of these systems may be needed in larger fields and may need to be operated frequently if the birds return after short periods. Once it is confirmed that this technology is a solution to the problem, it is necessary to highlight its flaws to increase its efficiency. Thus, creating an unpredictable autonomous algorithm that controls the drone according to the movement of the birds should be the main focus of future works.

1.2. Objectives and Contribution of the Dissertation

As stated above, farmers use different methods to deal with the challenge of bird damage to fruit crops. The most used in Beira Interior region in cherry and peach production is timed bird cannons, loudspeakers with sounds of predators, netting, and planting and harvesting manipulation. These techniques present the same problems: low mobility, predictability, and high maintenance, becoming inefficient in the long run. UAVs or drones can fly without pilots' onboard presence [16] and can be autonomous via an onboard electronic flight controller and a path map or remote-controlled from the ground. This technology is already widely used in agriculture in different applications due to its high mobility, task versatility, low maintenance, and cost. All the advantages of this technology fill the disadvantages of traditional systems, so drones with repelling systems attached start to emerge as good solutions against bird damage to fruit crops. Although these problems have been solved, new ones arise, such as short flight time. These aircraft can be controlled through flight optimization using trajectory planning and expanding batteries' capacity, representing one of the heaviest parts of the UAVs. Increasing its capacity also increases volume and weight, making the drone heavier and less efficient, and in most off-the-shelf products, it is not possible to modify them. So, it is necessary to optimize the path to ensure the most energy-efficient flight accordingly to the final objective.

This dissertation presents a novel path planning optimization algorithm for UAVs to help bird damage in agriculture, using metaheuristic optimization techniques and flight planning based on points of interest to focus the path to the most affected areas and random waypoints to avoid patterns. Different scenarios are tested in simulation and field, studying variables such as processing time, number of iterations, and energy consumption. This study focuses on energy efficiency and flight time optimization, which can be an asset in autonomous bird repelling UAVs, besides many other applications.

1.3. Overview and Organization of the Dissertation

Chapter 1 describes an overall overview of some essential concepts such as precision agriculture. Then, the bird damage to fruit crops problem was contextualized and explained how UAVs are used in agriculture and how they can help the study case. Finally, the dissertation's objectives and contribution are addressed, stating the need for a novel path planning optimization algorithm for UAVs, which is applied mainly to autonomous bird repelling systems, but it may include many other applications.

Chapter 2 describes the State of the Art of some tools and techniques necessary to develop the path planning optimization algorithm. Initially, were shown the most used and essential autopilots and Ground Control Station (GCS). Then it was explained what optimization algorithms are and some real-world applications, following by defining the concept of UAV path planning and some examples of optimization techniques in this field. In the end, a summary of the entire State of the Art is made.

Chapter 3 describes in detail all the UAV components built to test the novel path planning optimization algorithm, including its configuration and calibration, and GCS used in this study is also explored. After that, the mathematical methods and techniques used in the new algorithm are explained in detail.

Chapter 4 describes the path planning optimization algorithm and explains each of the four main steps: Parameter Setting, Minimization Between Points of Interest (PoIs), Maximization of Random Waypoints, and Creation of Pre-Planned Mission File.

Chapter 5 includes the analysis and discussion of the three case study, the first two in simulation and the last in the field with the test UAV. All cases were tested in three possible situations: same incidence rate, different rates, and different rates with no bird damage. The study cases were used to understand the path planning optimization algorithm performance and the effect of each parameter variation.
In Chapter 6 a global conclusion is made where all the work developed is presented. Then, a specific conclusion is presented, where the conclusions regarding the performance of the algorithm are presented. In the end, some indications of future work to be developed are made.

2. State of the Art

2.1. Autopilots and Ground Control Stations

Fully Autonomous Aerial Systems (FAAS) are an emerging workload wherein UAVs execute dynamic missions defined wholly by software. End users do not support pilot FAAS, nor do they define preset waypoints. Instead, they provide goals, constraints, and software that execute missions. Like edge-driven video analytics, FAAS processes images in real-time and leverages Artificial Intelligence (AI) for scene analysis. However, FAAS also controls aircraft flight, making flight paths dynamic [17]. Semi-autonomous systems can sense their environment and perform their tasks autonomously, but they may also be supervised by humans [18].

As previously mentioned in Section 1.1, conventional UAVs contain a flight controller system mechanically connected to the respective cockpit controls to define the aircraft direction, speed, and altitude assisted by an INS and external sensors. The manual control can be replaced or assisted by autopilot systems. These are crucial and provide semi-autonomous navigation and assist the remote piloting of the aircraft when required. Semi-automated piloting is possible using a remote control. The operator also can use a computer, tablet, or mobile phone to provide Global Navigation Satellite System (GNSS) waypoints that can be saved in the autopilot to navigate. Open hardware autopilots provide complete information about the electronic components of which they are composed and the code running on the system. On the other hand, the information provided by the closed hardware autopilots depends on the manufacturer's strategy and typically is not open to users [19].

There are few examples of autopilot boards at the academics and research level, but these controllers are not reliable, neither easy to buy nor make. However, some companies produce readily available, reliable, cheaper, and simple-to-use boards. Table 1 to Table 2 shows some of the most commercially available of both open and closed hardware used [20], [21], [30]–[39], [22], [40]–[44], [23]–[29]. Each board is different in weight, dimensions, processor, internal sensors, and interfaces and has its advantages and disadvantages, depending on the application. The same table shows that most controllers use the STM32 processor family with Advanced Reduced Instruction Set Computer Machine (ARM) architecture and 32-bits. Due to the global characteristics of the processors, information, and most run software's such as PX4, an open-source flight control software for drones and other unmanned vehicles [45], known for its ecosystem and hardware compatibility. Another type of processor widely use is the Raspberry Pi (RPi) with a shield to accommodate external sensors and additional input and output. These processors run some versions of

Linux and usually are popular due to being easy to program. The Intel Aero is another example of a Linux base autopilot developed by Intel and was discontinued in February of 2019 [46]. Both the Erle-Brain 2 and the Emlid Edge are also discontinued controllers, but they are still widely used. There are several software options for Ground Control Stations (GCS), and this controls the autonomous flights, the visualization of the flight map, make video streaming in real-time, among others. Mission Planner [47] is widely used due to having the full feature and lots of information. This open-source system is compatible with Windows and macOS, and some of its features are: point-and-click waypoint, using Google Maps/Bing/Open Street maps/Custom WMS; Select mission commands from drop-down menus; Download mission log files and analyze them; Configure autopilot settings; Interface with a PC flight simulator to create a full software-in-the-loop (SITL) UAV simulator; Run its own SITL simulation of many frame types for all the ArduPilot vehicles. Another popular GCS is the APM Planner 2.0 [48], because of its open-source system and compatibility with macOS and Linux. It has a smaller user base and a reduced feature set than the previous platforms, which can be advantageous for new users. QGroundControl works with MAVLink [49], a very lightweight messaging protocol for communicating with drones, capable of autopilots, including ArduPilot [50]. It is unique among the GCS offerings as it runs on all platform's desktop and mobile [51]. The last platform will be Litchi, a closed software available on Windows, macOS, Android, and iOS, known for being the most trusted autonomous flight app for DJI drones [52]. Other similar softwares are Drone Harmony [53], Rainbow [54], Red Waypoint [55], among others.

It is essential to highlight that autopilot hardware and software have much information and are available, reliable, easy to use, and have many open and closed sources depending on the applications and price.

Table 1 - AutoPilot Hardware - Open hardware.

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
Beagle Bone Blue Weight: 36g Dimensions:175mm x 112mm x 40mm	• AM335x 1GHz ARM® Cortex-A8 processor; • 512MB Random Access Memory (RAM); • Integrated power management; • 2×32-bit 200-MHz programmable real-time units; • NEON floating-point accelerator; • ARM Cortex-M3; • Programmed with Debian Linux.	• Nine axis INS (accelerometer / gyroscope / magnetometer); • Barometer; • Thermometer.	• Wireless: 802.11 bgn; • Bluetooth 4.1; • 8 x 6V servo output • 4x bidirectional direct current (DC) motor output; • 4x quadrature encoder input; • 11 x user programmable light-emitting diode (LED); • 2 x user programmable buttons; • 4x universal asynchronous receiver-transmitter (UART); • 12x Pulse-Width Modulation (PWM)/Timers; • Liquid crystal display (LCD); • MMC1; • 2x Serial Peripheral Interface (SPI); • 2x Inter-Integrated Circuit (I2C); • Analog to Digital Converter (ADC); • 2x Controller Area Network (CAN Bus); • Universal Serial Bus (USB) client for power & communications, USB host.
CUAV Nora Weight: 75g Dimensions: 46mm x 64mm x 22mm	• 32-bit STM32H743 main processor; • 480Mhz; • 1MB RAM; • 2MB Flash.	• InvenSense ICM20689 (accelerometer / gyroscope); • InvenSense ICM20649 (accelerometer / gyroscope); • Bosch BMI088 (accelerometer / gyroscope); • 2x MS5611 (barometer); • RM3100 (Industrial grade magnetometer).	• 14x PWM servo outputs; • Analog/ PWM RSSI input; • 2x Global Positioning System (GPS) ports; • 4x I2C buses; • 2x CAN Bus ports; • 2x Power ports; • 2x ADC input ports; • 2x USB ports.
OpenPilot CC3D Revolution (Revo) Weight: 9g Dimensions: 36 x 36mm	• STM32F405RGT6 ARM Cortex-M4; • 168 Mhz; • 1 MB Flash.	• InvenSense MPU6000 IMS (accelerometer / gyroscope); • Honeywell HMC5883L (compass); • MS5611 (barometer).	• 8x PWM outputs; • RC input; • ADC; • GPS; • Telemetry; • USB port; • Serial Wire Debug port; • HopeRF RFM22B 100mW 433MHz;

Table 1 - AutoPilot Hardware - Open hardware (cont).

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
Hex/ProfiCNC Cube Black Weight: 75g Dimensions: 95mm x 45mm x 32mm	• 32-bit ARM Cortex M4 core; • 168 Mh; • 256 KB RAM; • 2 MB Flash; • 32-bit failsafe co-processor; • PX4 FMUv3 generation.	• 3x IMS (accelerometers / gyroscopes /compass); • InvenSense MPU9250, ICM20948 and/or ICM20648 (accelerometer / gyroscope); • ST Micro L3GD20+LSM303D or InvenSense ICM2076xx (accelerometer / gyroscope); • 2x MS5611 (barometers).	• 14x PWM servo outputs; • SBus servo output; • RC inputs Chaotic PPM (CPPM), Spektrum SBus; • Analogue / PWM RSSI input; • 5x general purpose serial ports; • 2x I2C ports; • SPI port; • 2x CAN Bus interface; • 3x Analogue inputs (3.3V and 6.6V); • High-powered piezo buzzer driver. • High-power RGB LED; • Safety switch / LED; • Optional carrier board for Intel Edison.
F4BY FMU Dimensions: 50mm x 50mm	• 32-bit ARM Cortex M4 core with STM32 F407.	• MPU6000 (accelerometer / gyroscope); • MEAS 5611 (barometer); • HMC 5983 (compass).	• 5x UART serial ports; • 1x inverter for frsky telemertry; • 12x PWM outputs; • Spektrum input; • Futaba SBUS input support; • PPM; • RSSI input; • I2C; • SPI; • CAN; • USB; • 3.3V and 6.6V ADC inputs.
Drotek Pixhawk3 Pro Weight: 45g Dimensions: 71mm x 49mm x 23 mm	• 32-bit STM32F469 main processor; • 384KB RAM; • 2MB Flash; • PX4 FMUv4-PRO generation.	• ICM-20608-G (accelerometer / gyroscope); • MPU-9250 (accelerometer / gyroscope/ magnetometer); • LIS3MDL (compass).	• 6-14x PWM servo outputs; • RC inputs, Spektrum / DSM and SBus; • Analog / PWM RSSI input; • SBus servo output; • 6x general purpose serial ports; • 2x I2C ports; • 2x SPI ports; • 2x CAN Bus interfaces; • 2x Analog inputs for voltage and current; • High-power red, green and bue (RGB) LED; • Safety switch / LED.

Table 1 - AutoPilot Hardware - Open hardware (cont).

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
mRo Pixracer Weight: 10.54g Dimensions: 36mm x 36mm	• STM32F427VIT6 rev.3; • Ferroelectric RAM (FRAM) - FM25V02-G; • PX4 FMUv4 generation.	• Invensense MPU9250 (accelerometer / gyroscope / magnetometer); • Invensense ICM-20608 (accelerometer / gyroscope); • MS5611 (barometer); • Honeywell HMC5983 (magnetometer).	• Wifi: ESP-01 802.11bgn Flashed with MavESP8266; • Micro Secure Digital (SD) card reader; • Micro USB; • RGB LED; • GPS (serial + I2C); • TELEM1/TELEM2; • Wifi serial; • FrSky Telemetry serial; • Debug connector (serial + SWD); • GPS+I2C; • RC-In; • PPM-In; • RSSI; • SBus-In; • Spektrum-In; • UART; • FRSky-In; • FRSky-Out; • CAN; • ESP8266; • 6x servos output; • Joint Test Action Group; • Safety switch/LED. • High-powered piezo buzzer driver;
Erle-Brain 2 Weight: 100g Dimensions: 96mm x 70mm x 25mm	• Quad-core ARM Cortex-A7 Central Processing Unit (CPU); • 900MHz; • VideoCore IV 3D graphics core; • 1GB RAM.	• Gravity sensor; • Gyroscope; • Digital compass; • Pressure sensor; • Temperature sensor; • 5MP Camera.	• 12x PWM Output; • PPM input; • I2C; • UART; • Ethernet; • ADC • 4x USB; • MicroUSB Power; • High-Definition Multimedia Interface (HDMI); • Audio Jack; • SD Card slot.

Table 1 - AutoPilot Hardware - Open hardware (cont).

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
Holybro Durandal Weight: 64g Dimensions: 80mm x 45mm x 20.5mm	• 32-bit STM32H743 main processor; • 400Mhz; • 1MB RAM; • 2MB Flash; • 32-bit co-processor.	• InvenSense ICM20689 (accelerometer / gyroscope); • Bosch BMI088 (accelerometer / gyroscope); • MS5611 (barometer); • IST8310 (magnetometer).	• USB-C and JST_GH USB ports; • 16 PWM outputs; • Dual power module inputs; • SBus servo output; • RC inputs for CPPM and SBus; • DSM input port; • Analogue / PWM RSSI input; • 5x general purpose serial ports plus debug port; • 3x I2C ports; • 4x SPI buses; • 2x CAN Bus ports; • 2x analog inputs; • Safety switch/LED.
CUAV V5 Plus Weight: 90g Dimensions: 85.5mm x 42mm x 33mm	• 32-bit ARM Cortex M7 core; • 216 Mhz; • 512 KB RAM; • 2 MB Flash; • 32-bit IOMCU co-processor; • PX4 FMUv5 generation.	• InvenSense ICM20689 (accelerometer / gyroscope); • InvenSense ICM20602 (accelerometer / gyroscope); • Bosch BMI055 (accelerometer / gyroscope); • MS5611 (barometer); • IST8310 (magnetometer).	• 8 – 14x PWM servos outputs; • 3x PWM/Capture inputs on FMU; • SBus servo output; • Pulse-position modulation (PPM) connector; • SBUS/ Received Signal Strength Indication (RSSI) connector supports all radio control (RC) protocols; • Analog / PWM RSSI input; • 5x general purpose serial ports; • 4x I2C ports; • 4x SPI bus; • 2x CAN Bus ports; • 2x Analog battery monitor ports.

Table 2 - AutoPilot Hardware - Closed hardware.

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
Emlid Edge Weight 59g Dimensions: 97mm x 46mm x 15mm	• ARM Cortex-A53 64-bit quad-core CPU; • ARM Cortex-M3 co-processor; • 1GB RAM.	• 2x InvenSense ICM20602 (IMS); • MS5611 (barometer); • 2x IsenTek IST8310 (compasses); • uBlox NEO-M8N.	• 1x PPM; • 1xSBUS input; • 12x PWM servo outputs; • 2x CAN; • 2x USB; • 1x Serial+I2C; • 1x Serial+ADC; • 5.180 ~ 5.825 Ghz datalink at up to 27 dBm; • Power monitor up to 12S, 200A;
FlyWoo F745 AIO BL_32 Weight:8.5 g Dimensions: 33.5mm x 333.5mm	• STM32F745VG ARM; • 216 MHz; • 1MB Flash.	• InvenSense MPU6000 IMS (accelerometer / gyroscope); • BMP280 (barometer); • Voltage and current sensor.	• 7x UART; • 10x PWM outputs; • I2C; • USB port; • Camera input/output; • Built-in on-screen display (OSD);
NAVIO2 Weight: 23g (shield) + 54g (RPi2) Dimensions: 55x65mm (shield only)	• 1.2GHz 64-bit quad-core ARMv8 CPU (RPi3); • 1GB RAM.	• MPU9250 9DOF (IMS); • LSM9DS1 9DOF (IMS); • MS5611 (Barometer); • U-blox M8N; • Glonass/GPS/Beidou;	• UART; • I2C; • ADC; • PPM/SBus input; • 14x PWM servo outputs.
Intel Aero Weight: 30g (without heatsink, 60g (with heatsink) Dimensions: 88mm x 63mm x 20mm	• Intel® Atom™ x7-Z8700 Processor - 2.4 GHz burst, quad core, 2M cache, 64 bit - 4GB RAM LPDDR3-1600 - 32GB eMMC; • Linux 4.4.3-yocto-standard Operating System (OS) powered with Yocto Project* 2.1 (Krogoth).	• Bosch BMI160 6-axis (IMS); • Bosch BMC150 6-axis (compass); • MS5611 (barometer).	• 2x I2C; • UART; • SPI; • CAN; • 5x analog inputs; • 25x programmable general-purpose input/output (GPIO) pins; • Wi-Fi (802.11ac); • Micro HDMI 1.4b; • USB 3.0; • Camera Serial Interface - 2; • MicroSD memory card slot;

Table 2 - AutoPilot Hardware - Closed hardware (cont.).

AutoPilot Hardware	Processor	Internal Sensors	Interfaces
Mateksys F405-SE Weight: 10g Dimensions: 46mm x 36mm	• STM32F405RGT6 ARM; • 168MHz.	• InvenSense MPU6000 IMS (accelerometer / gyroscope); • DPS310 (barometer); • Voltage and current sensor.	• 6x UARTS; • 10x PWM outputs; • RC input PWM/PPM, SBUS; • 2x I2C port. • USB port; • Built-in OSD; • 3x ADC; • Micro SD slot.
Parrot Bebop Weight: 400g (with hull) Dimensions: 33x38x3.6cm (with hull)	• Parrot P7 dual-core CPU Cortex 9 • Quad-core Graphic Processor Unit (GPU); • 8GB flash; • Linux (Busybox).	• MPU6050 (accelerometer / gyroscope); • AKM 8963 (compass); • MS5607 (barometer); • Furuno GN-87F GPS; • Sonar; • Optical flow; • High Definition (HD) camera;	• UART serial ports; • USB; • Built-in Wifi.
RadioLink MiniPix Weight: 12g Dimensions: 39 mm x 39 mm x 12 mm	• STM32F405VGT6 ARM.	• InvenSense MPU6500 (accelerometer / gyroscope); • QMC5883L (compass); • LPS22HB (barometer).	• RC input (PWM/PPM, SBUS); • 3x UART; • I2C; • 2 x ADC for voltage and current sensor; • ADC; • MicroSD card slot; • Micro USB connector; • Buzzer; • Safety switch connector; • Power module.
QioTek Zealot F427 Weight: 65g Dimensions: 42mm x 65mm x 25mm	• STM32F427VIT6; • 16KB FRAM - FM25V01.	• ICM20689, ICM20602, and BMI088 (accelerometer / gyroscope); • MS5611 and DPS3018 (barometers); • IST8310 or QMC5883L (Compass).	• 14x PWM Outputs; • 4x relay outputs; • MicroSD card reader; • Micro USB or remote USB via a JST_GH connect; • Built-in RGB LED; • External buzzer interface • 2x 6.6V tolerant ADC inputs; • 5x UART; • Safety switch connector.

2.2. Optimization Algorithms

Optimization techniques, or algorithms, are used to find the solution to the problem specified in Equation (1). The procedure consists of finding the combination of design variable values that results in the best objective function value while satisfying all the equality, inequality, and side constraints.

$$
\begin{array}{llll}
\text{Minimize:} & f(x) & & \\
\text{Subject to:} & g_j(x) \leq 0 & j = 1, m & \\
 & h_k(x) = 0 & k = 1, p & \quad (1) \\
 & x_{iL} \leq x_i \leq x_{iU} & i = 1, n &
\end{array}
$$

Equation (1) expresses the standard form for a single-objective, non-linear, constrained optimization problem, where $f(x)$ is the object or goal function, $g_j(x)$ is an inequality constraint, $h_x(x)$ an equality constraint function and x represents the n design variables that are modified to obtain the optimum solution. The upper x_{iU} and lower x_{iL} bounds define the searchable design space, referred to as the side constraints [56]. Real-world optimization problems are often very challenging. As a result, many problems must be solved by trial and error using various optimization techniques. In addition, new algorithms are developed to eventually cope with these challenging optimization problems. Nature has motivated many researchers in different ways and thus is a rich source of inspiration. Nature-Inspired algorithms, such as particle swarm optimization (PSO), cuckoo search, and firefly algorithm, have gained popularity due to their high efficiency [57] and are all based on the principle of biology, physics, ethology, or swarm intelligence. Currently, Nature-Inspired optimization methods are applied in most areas, and their development and review are continuous.

Hajihassani et al. [58] performed a comprehensive review of PSO application in Geotechnical Engineering, presenting real examples as slope stability analysis, pile and foundation design, rock and soil mechanics, and tunneling and underground space technology, among others. In their study, the authors conclude that complex and not well-understood problems are the common obstacles in geotechnical engineering, where finding the optimum solution is difficult and even impossible in some cases. Consequently, based on the available literature, PSO has been extensively used in the field as a powerful optimization technique to find the optimum solution. The simplicity of the operations and reasonability of the results have paved the way to use PSO in various areas of geotechnical engineering. The PSO is an optimization algorithm that employs a swarm of particles to

traverse a multidimensional search space to seek out optima. Each particle is a potential solution and is influenced by the experiences of its neighbors as well as itself [59].
The Grey Wolf Optimizer (GWO) [60] is a technique inspired by the hunting method of the grey wolf (*Canis Lupus*). This animal is at the top of its food chain and usually lives in packs ranging from 5 to 12 elements having a strict social hierarchy. The mathematical models are based on the social hierarchy (α, β, δ, ω), tracking, encircling, and attacking prey. In GWO, α, β, and δ lead ω wolves toward the areas of the search space that are promising for finding the optimal solution. This behavior may lead to entrapment in a locally optimal solution. Another side-effect is the reduction of the diversity of the population and causes GWO to fall into the local optimum. Mohammad et al. [61] proposed an improved GWO for solving engineering problems to overcome these issues. The improvements include a new search strategy associated with selecting and updating steps. This improved technique was tested in benchmark functions and experimental environments. In the end, the authors show the applicability of the new algorithm for solving four engineering problems, including pressure vessel design, the welded beam design, and the optimal power flow problems for the Institute of Electrical and Electronics Engineers (IEEE) 30-bus and IEEE 118-bus systems.
Benkercha et al. [62] proposed a modified flower algorithm (FA) for extraction of the photovoltaic (PV) module parameters with maximum power point (MPP) estimation. The FA [63] is a metaheuristic algorithm that imitates the behavior of a plant's pollination. This process has four main rules which can aid to describe the flower algorithm, being global pollination (first rule), local pollination (second rule), flower constancy factor which can be expressed by a reproduction probability while this latter is proportional to the similarity among two working flowers (third rule), switching factor that allows passing from the local pollination to the global pollination or vice versa (fourth rule). The modified algorithm has the fundamental rules of the FA expect the fourth rule is modified in such a way that the switching probability is changed at each iteration. The main idea in this algorithm is to change the value of the switching probability to increase the accuracy of the solution so that the solution found will be close to the global solution. In addition, the probability becomes variable in this new algorithm at each iteration. The authors also simulated and experimented with the new technique for both single diode and double diode models, comparing it with three other optimization algorithms and concluded that the new technique has better optimization performance (accuracy, fast convergence, minimum iterations, and lower error than the other algorithms, therefore the identified parameters by the modified FA are more accurate than the one obtained from other algorithms. In predicting current, voltage, and power at the MPP, the parameters for the single diode model show the effectiveness of prediction in both features' days, and a high matching between the measured MPP values and the predicted one is achieved.

Typically, in practical design optimization of truss structures, the aim is to find a minimum cost or weight design by selecting cross-sectional areas of structural members from a table of available sections. The final design satisfies strength and serviceability requirements determined by technical. Hasançebi et al. [64] proposed a bat-inspired algorithm for structural optimization. Bat-inspired (BI) algorithm derived from the echolocation behavior of bats. Echolocation is an advanced hearing-based navigation system used by bats and some other animals to detect objects in their surroundings by emitting a sound to the environment. After testing the presented technique in four truss structure examples, the results demonstrate the algorithm's efficiency, which found the best-known design for the first two test problems and converged to improved designs in the last two test problems. The results also show that the BI algorithm also has a good convergence speed compared to most other metaheuristic techniques.

There are many optimization algorithms, and the ones presented above are just a few examples. Due to the need and this being a very researched area, new techniques Nature-inspired will continue to emerge to perform mathematical benchmarks and real-life applications.

2.3. UAV Path Planning Algorithms

Path planning is one of the most critical problems explored in UAVs to find an optimal path between source and destination. The path determination should be free from all collisions from the surrounding obstacles. To have low computational cost and time for optimal path planning is the primary objective of these techniques. The path generated should be optimal to consume minimum energy, take less time, and reduce collision between the UAVs. On the other hand, it needs to satisfy the robustness and completeness criterion during path planning techniques. The significant challenges for optimal path planning of UAVs are path length, optimality, completeness, cost, time and energy efficiency, robustness, and collision avoidance [65].

In 2018, Zhao et al. [66] surveyed computational-intelligence-based UAV path planning. Computational intelligence (CI) is a set of nature-inspired computational methodologies and approaches that can address complex real-world problems for which mathematical or traditional modeling is not practical. In their study, 231 articles were collected and classified in three orthogonal dimensions, being classification from the aspect of algorithms, classification from the aspect of the time domain, and classification from the aspect of the space domain. Based on the aspect of algorithms, the authors presented Figure 3. This pie chart shows the percentages of various CI algorithms used for UAV path planning from 2008 to 2017 through the authors' research. The genetic algorithm (GA) was the most common, accounting for 21%. Ant colony optimization (ACO) and artificial neural networks

(ANN), as two of the most famous intelligence algorithms, occupying the second and third positions with 16% and 15%, respectively, followed by learning-based methods (LB), PSO, and fuzzy logic algorithms (FL).

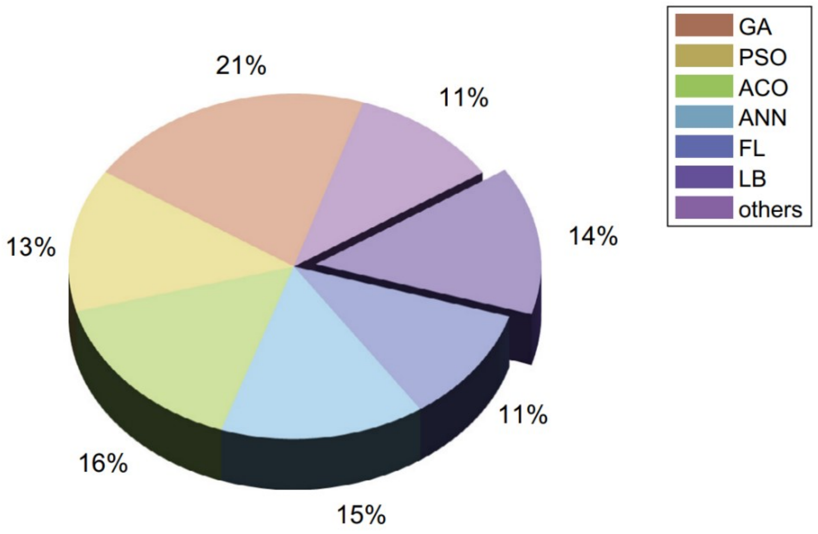


Figure 3 – Classification of CI methods in UAV path planning by algorithm [66].

From time-domain classification, the algorithms are divided into two categories online or offline. An online method is a method that can plan the UAV paths in real-time. In other words, the UAV can identify changes in the environment and react to them. In contrast, an offline method performs path planning based on offline information instead of real-time information. In all articles studied, only 29.9% of methods are online, and most methods focus on offline algorithms and their improvements, accounting for approximately 70.1% of methods. According to their research, for offline path planning, most studies focus on minimizing the length of a path with obstacle or threat constraints in the environment. The optimization problem is multi-modal because there can be multiple sets of paths with varying costs for every set of obstacles or threats. Thus, CI methods such as GA or PSO would be highly suitable for optimizing the generated paths. Researchers have shown that CI methods have obtained high-quality solutions because they are computationally efficient on a wide variety of multi-modal unconstrained problems. Online path planning is a dynamic multi-objective optimization problem. The most popular online path planning algorithms are based on GA and FL. In the last classification, the algorithms are divided into 2D and 3D environments. Traditional path planning methods has usually been described by a 2D scene. 55.8% of articles studied explore 2D path planning methods, while 44.2% of articles explore 3D methods. In a 2D scene, it is supposed that the UAV flies by maintaining its height or manual adjustment. From the optimization point of view, there are no standard solutions in a 2D path planning problem. Fortunately, CI algorithms reduce the requirements of computing the gradients of cost functions and constraint functions, enabling the problem to be solved and optimized. Path planning algorithms for 3D environments are urgently needed due to an increasing range of fields, such as transportation, detection, navigation, and operations. One classical problem is modeling the environment while considering the kinematic constraints to plan a collision-free path. Thus, a comprehensive analysis of the most optimization methods for UAV path planning and the different aspect of its applications was presented, given a complete guideline that can help related researchers.

Tseng et al. [67] developed flight planning with recharging optimization for battery-operated autonomous UAVs. This is an example of path planning applied to a specific area. They start to conduct an empirical study to model the energy consumption of drones, considering various flight scenarios. Then, a joint problem of flight mission planning and recharging optimization for drones was studied, using the calibrated power consumption model of a drone, to complete a tour mission for a set of sites of interest in the shortest time, considering uncertainty in dynamic environments, such as wind conditions. They also presented and implemented algorithms for solving flight mission planning and recharging optimization in a drone management system, supporting real-time flight path tracking and re-computation in dynamic environments. At the end, the authors evaluated the results for the algorithms using data from empirical studies and proposed a robotic charging station prototype that can recharge drones autonomously. In conclusion, the authors highlight the importance of automated drone management systems for practical applications of drones, and future work should incorporate a variety of additional features, such as restrictions of no-fly zones and attitude, and wind speed forecast.

Li et al. [68] present another example applied to precision agriculture, using a hybrid PSO algorithm to optimize flight paths in a UAV group. Farmers need to spray daily different orchard blocks. Using a single drone becomes an impossible task due to endurance and battery change. The authors developed a hybrid optimization algorithm that combined PSO with the variable neighborhood descent technique as the local search to overcome this problem. The focus of this article is to obtain the optimal paths for the group of UAVs so that the flight time of a mission is minimized. After presenting the structure and architecture of the proposed algorithm, the authors show the simulation results made in two agriculture regions of Shaanxi, for both approach A (minimizing the total flight distance) and approach B (optimize the flight paths of the whole UAVs group with minimum make-span) with two and three groups of drones. In all path planning simulations, the total path length was longer in approach B than in approach A, yet the time was always shorter in approach B. Therefore, the authors conclude that the proposed hybrid algorithm can effectively shorten the UAV group's flight time, enabling multiple agricultural UAVs to be better applied in precision agriculture.

2.4. Final Remarks

FAAS is an emerging area based on complex control systems, sensors, and computer vision that can complete an objective without human interventions. This technology is still not commercially available nor fully implemented due to regulation and safety reasons. On the other hand, autopilot controllers and waypoints software are well established as semi-autonomous systems. Nowadays, there are commercially available hardware and software that is easy to use for both open and closed systems, with a lot of information, and reliable.
Optimization is a commonly encountered mathematical problem in all engineering disciplines and means finding the best possible solution. Optimization algorithms can be either deterministic or stochastic. Former methods to solve optimization problems require enormous computational efforts, which tend to fail as the problem size increases. This is the motivation for employing bio-inspired stochastic optimization algorithms as computationally efficient alternatives to the deterministic approach. Meta-heuristics are based on the iterative improvement of either a population of solutions or a single solution and mainly employ randomization and local search to solve a given optimization problem [69]. Optimization problems are wide-ranging and numerous. Hence, methods for solving these problems ought to be an active research topic. The emergence and evolution of new and old optimization techniques improve benchmarks, dynamic optimization problems, and real-world applications [70].
The main objective in UAV path planning is to find the solution route between source and destination based on a goal (cost, time, distance, among others). Using an optimization algorithm, it is possible to calculate the best flight path for UAVs applying a suitable cost function to the problem, with less time and computer power than traditional systems.

3. Materials and Methods

3.1. Materials

3.1.1. UAV

Multirotor drones being low cost, with good maneuverability and VTOL capability, are suitable for precision agriculture remote sensing tasks [71]. The UAV frame used in this study was a multirotor with four motors (quadcopter), equipped with self-tightening propellers that make it very simple to assemble and disassemble in the field and for transportation. This configuration is also one of the most used in agriculture nowadays and can lift its weight and equipment to repel birds without compromising flight time. However, because of the short flight endurance from lithium polymer (Li-Po) batteries, UAVs have a smaller field area coverage per flight than airplanes [71]. This work used four Li-Po batteries in series with the capacity of 5000 mAh. Due to this study being of path planning is not essential to have the best UAV/battery weight ratio, but this requirement is advised in a real-life application. Figure 4 shows the multirotor used in this study during a test flight.

Figure 4 – Multirotor used in this study.

The optimization algorithm presented in this study generalizes to any UAV configuration and flight time capabilities. Thus, the essential component to consider is the flight controller that needs to have an autopilot compatible with the chosen GCS, suitable internal sensors, and the necessary input/output accordingly to the UAV configuration. In this case, was chosen Hex Cube Black [72], previously known as Pixhawk 2.1, which was already

characterized in the last chapter. This autopilot is flexible, intended primarily for manufacturers of commercial systems. It is based on the Pixhawk-project FMUv3 open hardware design and runs PX4 on the NuttX OS [73]. The Hex Cube Black is reliable and has a premium built quality with much information. As a GPS receiver, the HERE+ [74] was chosen due to quality construction, compatibility, and information. Another essential component is the power brick that comes with the flight controller that provides power to all the UAV components and measures current consumption and battery voltage. Thus, it makes it possible to test the path planning optimization algorithm in real-world scenarios.
It is also important to mention that during the construction of this system, it was necessary to use computer-aided design (CAD) techniques and 3D printing via fused deposition modeling (FDM) to accommodate all the electronics into the frame. The components manufactured through FDM were made in Acrylonitrile Butadiene Styrene (ABS) because it is resistant to temperature changes and impact. ABS is the most utilized material after Polylactic Acid (PLA), and in addition to the features highlighted above, it also has good mechanical properties, low price, and long-life services [75]. One example is shown in Figure 5 with both the CAD design (left) and the 3D printed component (right) developed to secure the GPS antenna into the quadcopter frame.

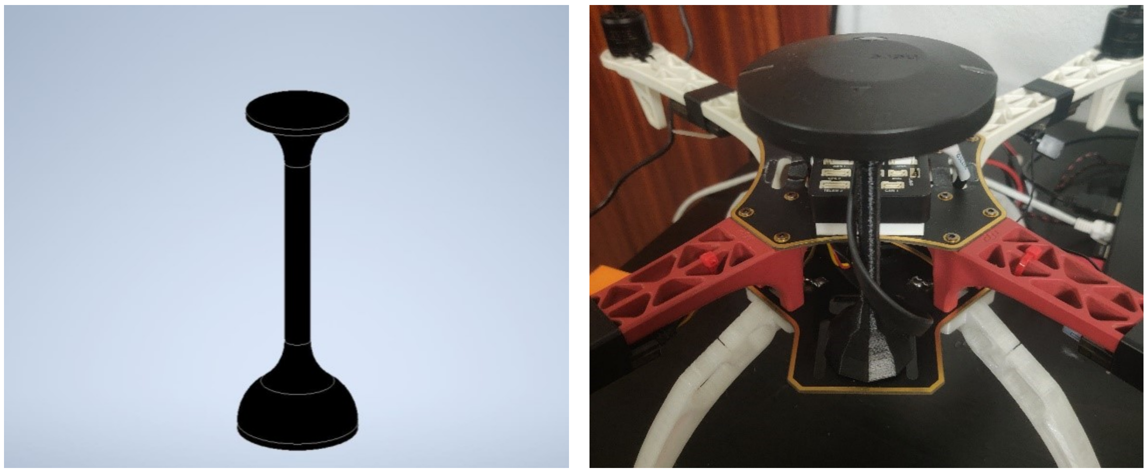


Figure 5 – CAD design (left) and 3D printed component (right) developed to secure the GPS antenna into the quadcopter frame.

To activate the flight plan, the RadioKing TX18S was used, a 2.4 GHz, 16 channel (CH) multi-protocol radio frequency (RF) system transistor with the open-source firmware OpenTX [76] for radio transmitters. A multi-protocol radio was chosen because it can communicate with the FrSky X8R, which is the receiver that the UAV of this study was built with but is also compatible with several telemetry receivers for future work. However, those components are not the most important for this research work, so it is only necessary to ensure the compatibility transmitter/receiver and an OpenTX model with all the configurations for a safe flight. It is essential to mention that sometimes accidents related to the arm and disarm of the drone happen unintentionally. For that reason, a sequence of switches was programmed in the radio transmitter instead of the pre-defined throttle stick, possible through the logical functions in OpenTX and changing the pre-defined parameters in the GCS.

3.1.2. Ground Control Station

GCS has several different functions that vary from software to software, as mentioned in Section 2.1. In this study, Mission Planner was used to updating firmware, initial configuration, advance parameter configuration, and flight planning. Although this software is easy to use in most of its features, it presents some complexity in others due to being very complete. Particularly in the visualization and analysis of flight logs, APMPlanner2 was used, which has a more straightforward and equally complete graphical interface environment.

During the UAV setup start, the autopilot board was connected to the computer via a USB port to load the firmware that matched the chosen frame. Mission Planner has all the versions classified as: stable (passed all stages and it is good to use), beta (prior stable that may have some bugs), and latest (passed development team and all automated tests and are ready to be tested by users). To ensure a safe flight and easy configuration, a stable version was selected. In this case, it was chosen ArduCopter V4.0.7 Quad, being the newest stable version. Then the setup itself starts, and it was in this step, the UAV is configured and calibrated, ensuring a precise and safe flight. The setup menu of Mission Planner as three main sections of settings: mandatory hardware, optional hardware and advanced. The mandatory hardware allows the user to calibrate sensors like the accelerometer and compass, with a set of drone moves and the radio, moving the sticks and switches of each channel to their minimum and maximum. Apart from the sensor calibration, it also lets the user define flight and failsafe modes. Three flight modes were selected: loiter, holds altitude and position; auto, executes pre-defined mission; and return to launch (RTL), returns above takeoff location and lands. Failsafe modes ensure a safe flight if anything happens and are triggered when the values are lower than those defined and activate a flight configuration like landing or RTL. For this case, a low battery, controlled by the voltage measurement in the power brick and radio failsafe, previously called throttle failsafe, was chosen because it uses the throttle channel to signal the loss of contact. Other parameters can be adjusted in mandatory hardware, but they weren't used like servo output calibration, automatic dependent surveillance-broadcast, and ECS calibration. Important to mention that although the last parameter is crucial for an efficient flight, the systems used are blocked and do not allow calibration, thus having been manually adjusted the PWM channel in the radio transmitter. Except for the battery monitor calibration found in the second setup section, changes were not made in the two other sections. To calibrate the power brick in the battery monitor set, two known measures provided by a power supply were applied. The tools and connections used in this system calibration are represented in Figure 6. First, the propellers were removed, and the system was connected to the computer. Next, the UAV was powered by the power supply with a constant voltage of 16.80 V, measured with an accuracy of equal or less to 0.1% plus two digits. In the end, the drone was armed and activated with full throttle, and the current consumption was measured in the power supply also with an accuracy of equal or less to 0.1% plus three digits. Both measures were inserted in the GC software, and the values were loaded to serve as references.

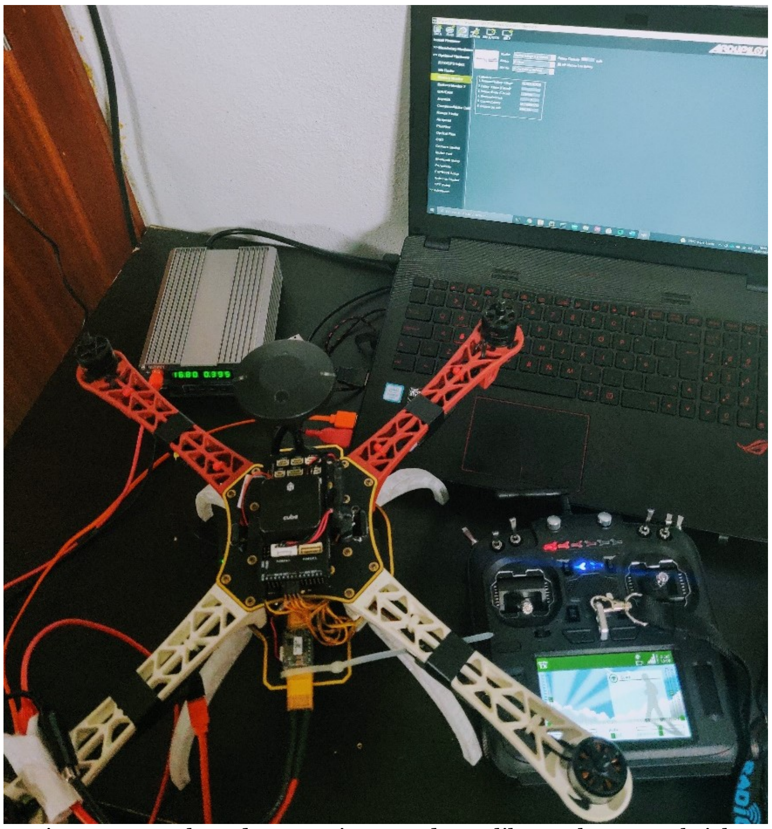
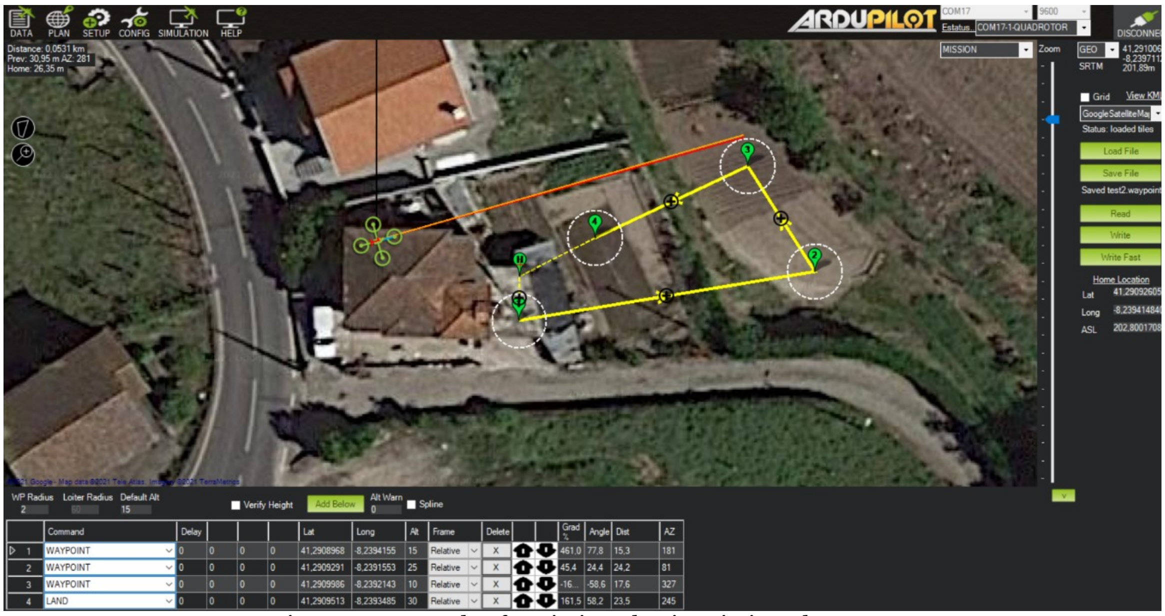


Figure 6 – Tools and connections used to calibrate the power brick.

An essential function from Mission Planner for this work is mission planning. Together with the autopilot, the software allows the UAV to flight through a waypoint map file without human intervention. After setting the home location where the vehicle is armed, the user can point and click a list of waypoints and create a mission. The generated mission can be written into the autopilot, but it can also be read from it. Each mission can be saved as mission plain-text file format that a text editor can read, and the files can also be loaded into the GCS software. Important to mention that not all software's share the same file configuration. So, it is not always possible to make a waypoint file in one software and read it in another. In the planning section of Mission Planner, it is possible to visualize a map with the waypoints created and their configurations and the default parameters that can be changed. Whenever a point is created, the software shows a bar with its parameters that can be modified manually. The most important for this case are command (landing, delay, RTL), delay, latitude and longitude coordinates, and altitude. Figure 7 shows an example of a pre-planned mission on Mission Planner, and it is possible to visualize the previously described functions.



Distance: 0,0531 km
Prev: 30,95 m AZ: 281
Home: 26,35 m

41,291006
-8,23971
201,89m

Status: loaded tiles

Saved test2.waypoint

Home Location

Lat 41,29092605

Long -8,23941484

ASL 202,8001708

WP Radius 2 | Loiter Radius 60 | Default Alt 15 | Alt Warn 0

	Command	Delay				Lat	Long	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
▷ 1	WAYPOINT	0	0	0	0	41,2908968	-8,2394155	15	Relative	X	⬆	⬇	461,0	77,8	15,3	181
2	WAYPOINT	0	0	0	0	41,2909291	-8,2391553	25	Relative	X	⬆	⬇	45,4	24,4	24,2	81
3	WAYPOINT	0	0	0	0	41,2909986	-8,2392143	10	Relative	X	⬆	⬇	-16...	-58,6	17,6	327
4	LAND	0	0	0	0	41,2909513	-8,2393485	30	Relative	X	⬆	⬇	161,5	58,2	23,5	245

Figure 7 – Example of a mission plan in Mission Planner.

Through the menus available in the software's graphical interface environment, it is easy to ensure that all the general parameters are set, but sometimes it is necessary to change specific values, and for that, Mission Planner provides a complete list. Outside this method, command-line interface configuration or Python programming language scripts with pre-developed libraries are available.

3.2. Methods

3.2.1 Particle Swarm Optimization

Initially proposed in 1995 by Kennedy et al. [77] , PSO is a method for optimizing continuous nonlinear functions. This evolutionary computation technique was developed to simulate a simplified social system and has been used for approaches across a wide range of applications or specific requirements.

The optimization algorithm is initialized with a population (swarm) defined as n of random solutions called particles, that have the dimension of the problem defined as dim. Each position x_{ij} of the jth dimension of the ith particle keeps track of its best solution in the problem space (fitness) and the corresponding coordinates, this value is called p_{best}. The overall fitness of the swarm is also tracked, and it is called g_{best}. Every iteration defined as it changes the velocity v_{ij} of each particle toward its p_{best} and g_{best} locations. Velocity is weighted by two different random numbers in the interval [0, 1] defined as $r_1$and r_2 and two constants named c_1 and c_2. The random numbers control the acceleration, and the constants control the stochastic acceleration terms towards p_{best} and g_{best} [78]. In the original form proposed by Kennedy et al. [77], both c_1 and c_2 are set to two, making the search to cover the region centered in $p_{best}$and g_{best}. These values can be changed to achieve better performance, and over the year, new common values appear [79]. Equations 2 and 3 show how the velocity and position of each particle is updated.

$$v_{ij}^{it+1} = v_{ij}^{it} + c_1 \cdot r_1 \cdot \left(p_{best\,ij} - x_{ij}^{it}\right) + c_2 \cdot r_2 \cdot \left(g_{best\,ij} - x_{ij}^{it}\right) \tag{2}$$

$$x_{ij}^{it+1} = x_{ij}^{it} + v_{ij}^{it+1} \tag{3}$$

Since the initial version of PSO was not very effective in the optimization problem, a modified PSO algorithm [80] appeared soon after the initial algorithm was proposed. Inertia weight was introduced to the velocity update formula, and the new velocity update formula became Equation 4. Although this modified algorithm has almost the same complexity as the initial version, it has dramatically improved the algorithm performance. Therefore, it has achieved extensive applications. Generally, the modified algorithm is called the canonical PSO algorithm, and the initial version is called the original PSO algorithm [81].

$$v_{ij}^{it+1} = w \cdot v_{ij}^{it} + c_1 \cdot r_1 \cdot \left(p_{best\,ij} - x_{ij}^{it}\right) + c_2 \cdot r_2 \cdot \left(g_{best\,ij} - x_{ij}^{it}\right) \tag{4}$$

After the new particle position is calculated, it is tested to ensure constriction. If it is not within limits, it will have to be modified through a condition. Figure 8 shows the pseudocode of the PSO algorithm with the same conditions as the new algorithm proposed in this dissertation. The pseudocode uses the maximum number of iterations it_{max} as the final condition and constricts the generation of a new random particle.

```
Initialize the swarm with $n$ and $dim$
Initialize $c_1$, $c_2$, $w$, $it_{max}$
while it less or equal $it_{max}$:
   for each particle
      Calculate the fitness of each particle
      Update the $p_{best}$ and $g_{best}$
   end for
   for each particle
      r1 = rand
      r2 = rand
      Equations 4
      Equations 3
      if particle < lower bound or particle > upper bound
            Random new particle
      end if
   end for
   it = it +1
end while
```

Figure 8 – Pseudo code of PSO algorithm.

PSO advantages can be summarized as follows: It does not require the differential of the optimized function, derivative and continuous; its convergence rate is fast, and the algorithm is simple and easy to execute through programming. The same also has some disadvantages as it probably falls into the local extreme and cannot get a correct result; does not sufficiently use the information obtained in the calculation procedure. During each iteration, instead, it only uses the information of the swarm optima and individual optima; though PSO algorithm provides the possibility of global search, it cannot guarantee convergence to the global optima; and is a meta-heuristic bionic optimization algorithm, furthermore, there is no rigorous theory foundation so far. It is designed only by simplifying and simulating some swarms' search phenomenon, but neither explains why this algorithm is effective from the principle nor specifies its applicable range. Therefore, the PSO algorithm is generally suitable for a class of optimization problems that are high dimensional and do not require very accurate solutions [81].

3.2.2 Haversine Formula

Several methods have been tested to measure the distance between two geographic coordinates, including the Google Maps Platform APIs [82], Haversine Formula [83], Spherical Law of Cosines [84], and Equi-rectangular Approximation [85]. The first approach uses Google resources but proved to be very time-consuming, dependent on an internet connection, and just works on road routes. The other three formulas all had the same results and, although they are less accurate than the first method, the error is proportional to the distance, which is not relevant for the study case. Thus, Haversine Formula was chosen due to simplicity, precision, and previous work already developed [83], [86].

The Haversine Formula is an essential equation in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes [87]. It is necessary to know the geographic coordinates of each point to apply this method. The latitude and longitude of each point is represented as $lat1$, $lat2$, and $lon1$, $lon2$, respectively, and the earth's radius is defined as r in Equations 5 and 6, representing the Haversine Formula. The distance between the two points is defined as d in Equation 6.

$$c = \sqrt{\sin^2\left(\frac{\text{lat1-lat2}}{2}\right) + \cos(\text{lat1}) \cdot \cos(\text{lat2}) \cdot \left(\frac{\text{lon1-lon2}}{2}\right)} \tag{5}$$

$$d = 2 \cdot r \cdot \arcsin(c) \tag{6}$$

3.2.3 Generate Random Geographic Coordinates within a Circle

To generate a random point (x, y) over a disk with radius R, Equations 7 and 8 are used:

$$x = r \cdot \cos\theta \tag{7}$$

$$y = r \cdot sin\ \theta \tag{8}$$

$r \in [0,\ R]$, and is a random value calculated through Equation 9:

$$r = R \cdot \sqrt{random()} \tag{9}$$

And θ another random value, where $\theta \in [0,\ 2\pi]$ and it is calculated by Equation 10:

$$\theta = 2 \cdot \pi \cdot random() \tag{10}$$

Figure 9 shows the unit disk and two thousand random points generated with Equations 7 and 8. This graph demonstrates that all points are evenly distributed, with no disk area being underloaded.

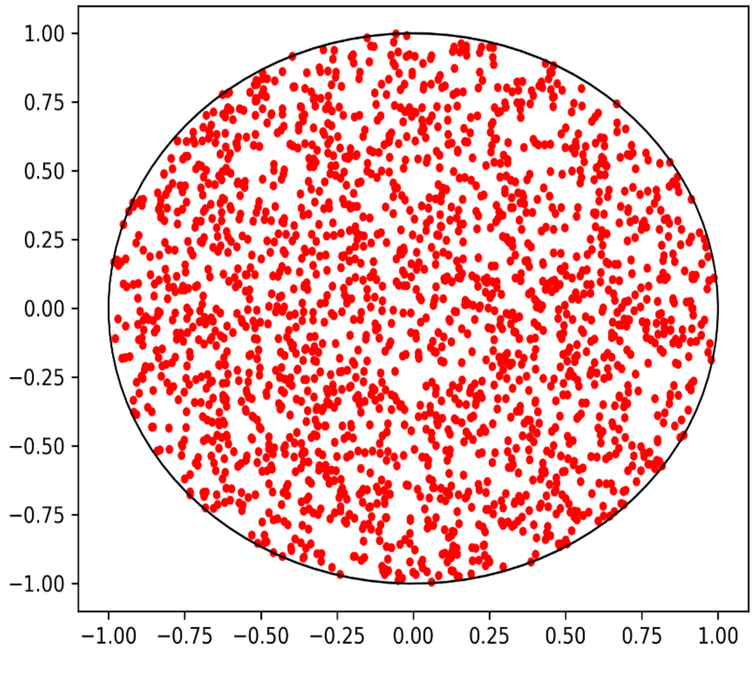


Figure 9 – Two thousand random points in the unit disk.

Lastly, it is necessary to convert the coordinates of the points to geographical coordinates (latitude and longitude). One-degree latitude corresponds to approximately 111.2 km, and one-degree longitude corresponds to approximately 111.2 km at the equator but 0 km at the poles. To generate random geographic coordinates within a circle, Equation 11 to 13 where used.

$$OneDegree = \frac{Earth\ Radius \cdot 2\pi}{360} \tag{11}$$

$$RandomLatitude = \frac{latCenter + x}{OneDegree} \tag{12}$$

$$RandomL\text{ongitude} = \frac{lonCenter + y}{OneDegree * \cos\left(\frac{latCenter \cdot \pi}{180}\right)} \tag{13}$$

$Earth\ Radius$ is given in meters, and $latCenter$ and lonCenter are the latitude and longitude of the center of the circle, respectively.

3.3 Final Remarks

A UAV capable of flying semi-autonomously was built to test the novel optimization algorithm. The configuration of this system is not important for the problem studied, as it is only necessary to ensure that it can fly with the dispersion system without significantly damaging the flight time. Thus, a quadcopter was chosen, equipped with Hex Cube Black autopilot, and a Here+ as a GPS module. A GCS software was used to upload firmware, initial configuration, modifying advanced parameters, pre-planned flights, and review data logs. Because Mission Planner is complete, intuitive, and full of information, it was used as GCS for this study except to read and analyze data logs where APMPlanner2 was used due to its simplicity. This chapter also presented the mathematical methods and techniques used in the novel path planning optimization algorithm. PSO is a population-based meta-heuristic search algorithm that has been widely applied to a variety of problems [88]. The Haversine formula is used to calculate the geographical distance on earth between two coordinates [89]. In the end, is shown a method to generate random geographical coordinates within a circle with disk point picking.

4. Novel Path Planning Algorithm

4.1. Global Architecture

Before developing the proposed optimization algorithm, it was necessary to study the general problem and the tools needed to solve it. Birds damage trees and eat fruits to producers around the world, causing quantity and quality to decrease. Traditional repelling systems work in the short term, but because they possess low mobility and as birds can easily detect patterns, they became ineffective. UAVs have already proven to be essential tools to solve this problem, bridging traditional systems disadvantages. However, because drones have limited flight time, optimizing the path according to the bird's pattern is necessary.

After analyzing the problem, the algorithm was developed in Python programming language, in version 3.8, due to its versatility and pre-existing libraries in all areas of the algorithm, such as Graphical User Interface (GUI), math functions, and file manipulation. The fields were divided into plots assigned by the producer or detection sensors positioned along the field. These segments will be mentioned as points of interest (PoI) and have different damage proportions and will be divided accordingly. In this way, and since the problem needs consistent application of the repelling systems, the optimization algorithm needs to minimize the flight distance between sections and maximize it according to the percentage of damage in each plot. Yet, birds can detect patterns and learn how to avoid them, so is also required different numbers of random waypoints according to the area. As already mentioned, Mission Planner was chosen as pre-planning software, so it is essential to generate a compatible file of waypoints with various parameters, being the most important for this work the coordinated geographical system (latitude and longitude), height above takeoff and command. Figure 10 represents the four main steps of the novel path planning optimization algorithm, and they will be further explained in the following sections.

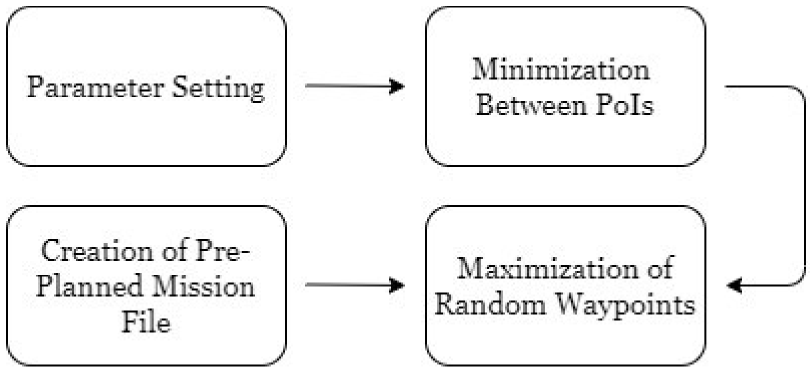
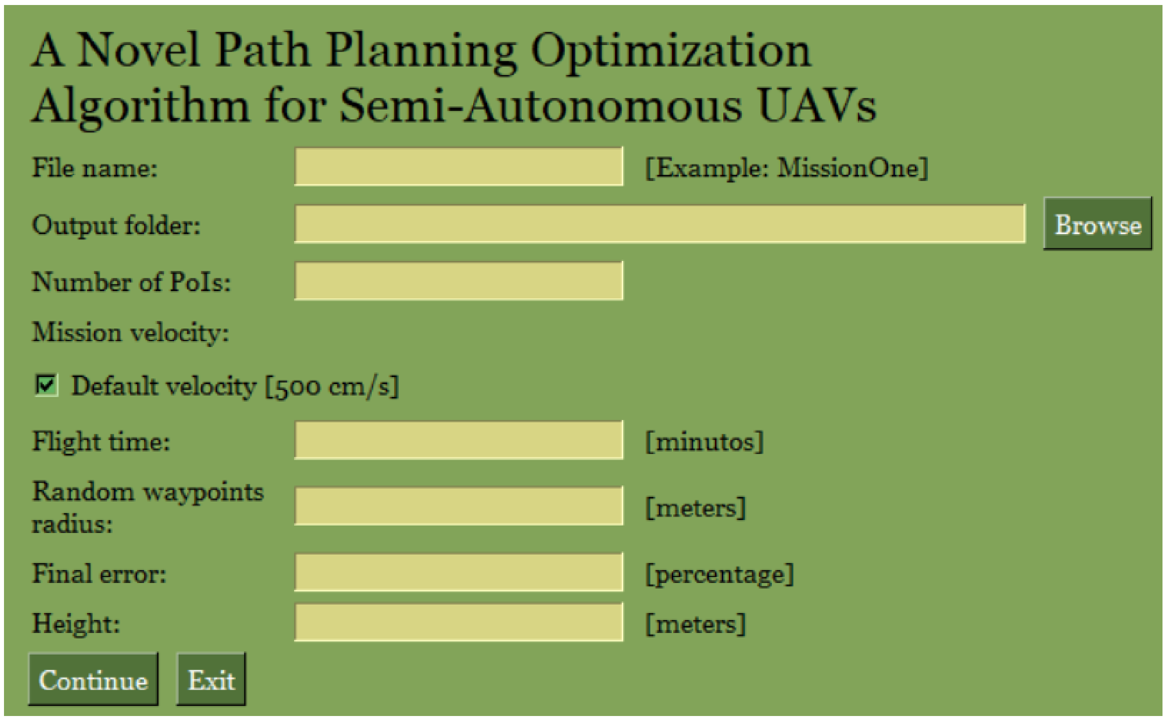

Parameter Setting
Minimization Between PoIs
Maximization of Random Waypoints
Creation of Pre-Planned Mission File

Figure 10 – Path planning optimization algorithm main steps.

4.2. Parameter Setting

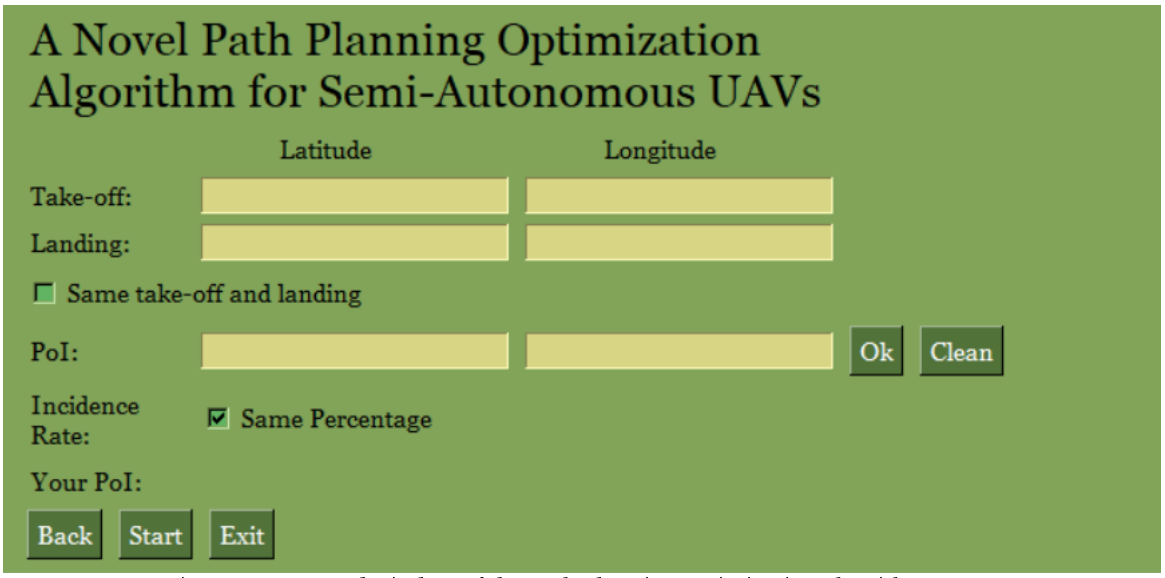
This path planning optimization algorithm can be applied to any UAV, orchard, and farmer, so it was essential to develop a GUI to import the specific parameters. The graphical interface was built around PySimpleGUI, a friendly pythonic interface library that runs on Windows, Linux, and macOS. The GUI was built to be visually simple and easy to use and can be divided into two main windows. It is possible to insert the file name (mission plain-text file format generated) along with the destination path on the first page. A button was also developed to browse folders on the computer so the user can choose quickly. Additionally, it is further possible to enter the number of PoIs, the speed of the autonomous mission in cm/s, and the drone's flight time in minutes. The speed matches the parameter `WPNAV_SPEED` on Mission Planner, where the unit is cm/s. If the user does not want to modify it, a check button to activate the default velocity of 500 cm/s was built. These parameters are used to calculate the total distance for path planning. The user can also add the radius of the randomly generated waypoints around the PoIs (random waypoints radius), the final error in the total distance for path planning in percentage, and the height at which the drone will fly. All these parameters, except for the final error, are in meters. When everything is complete, it is possible to continue to the next page or exit. Figure 11 represents the first window of the path planning optimization algorithm GUI developed.


A Novel Path Planning Optimization Algorithm for Semi-Autonomous UAVs

File name:		[Example: MissionOne]
Output folder:		Browse
Number of PoIs:		
Mission velocity:		
[✓] Default velocity [500 cm/s]		
Flight time:		[minutos]
Random waypoints radius:		[meters]
Final error:		[percentage]
Height:		[meters]

Figure 11 – First window of the path planning optimization algorithm GUI.

The second window is used to insert geographical coordinates of take-off, landing, and PoIs, with the percentage of bird damage incidence at each point. All coordinates need to be described in latitude and longitude, respectively, and a point instead of a coma must be used. In some cases, the take-off and landing points are the same, so a check button has been included not to enter twice the same values. To insert a new PoI, the user writes each value and clicks the `Ok` button, and the values will appear below. If the user mistake entering some value, the `Clean` button undo action and allows writing again. The incidence rate is the bird damage rate at each PoI, which can be assigned the same value to all points through the `Check` button, causing the same number of random waypoints to be generated, or a value can be entered, distributing the points according to the damage. These values can be entered into an input if the check button is disabled, and it will explain how the scale works. At the bottom, this page contains three buttons where the user can go back to the previous page, start the optimization algorithm, or exit. Figure 12 represents the second window of the path planning optimization algorithm GUI developed.


A Novel Path Planning Optimization Algorithm for Semi-Autonomous UAVs

	Latitude	Longitude		
Take-off:				
Landing:				
☐ Same take-off and landing				
PoI:			Ok	Clean
Incidence Rate:	☑ Same Percentage			
Your PoI:				

Figure 12 – Second window of the path planning optimization algorithm GUI.

4.3. Minimization Between PoIs

Depending on the type of field and the position of the PoI, the UAV must fly according to the needs. It is crucial that the algorithm receives the data and establishes the shortest path to save the battery for the areas next to the PoI. The PSO to minimization is used to calculate the fastest route or the minimum distance. Each particle contains a permute sequence of PoIs, and the objective function is the sum of the distances, using Haversine Formula, between the points, the take-off, and landing. In the end, this function will send the minimum distance and the order sequence of PoIs that the drone needs to fly. If a farmer inserts the value zero in the incidence rate, in other words, without any bird damage, the algorithm will eliminate it from that flight. Figure 13 represents the importance of minimizing the path between PoIs with two cases through an example. One with optimization represented as a) and the other a random order without optimization represented as b). The path of the first case scenario has a total distance of 216.3 m, while the path of the second case scenario has 268.3 m, which represents a reduction of 52 m.

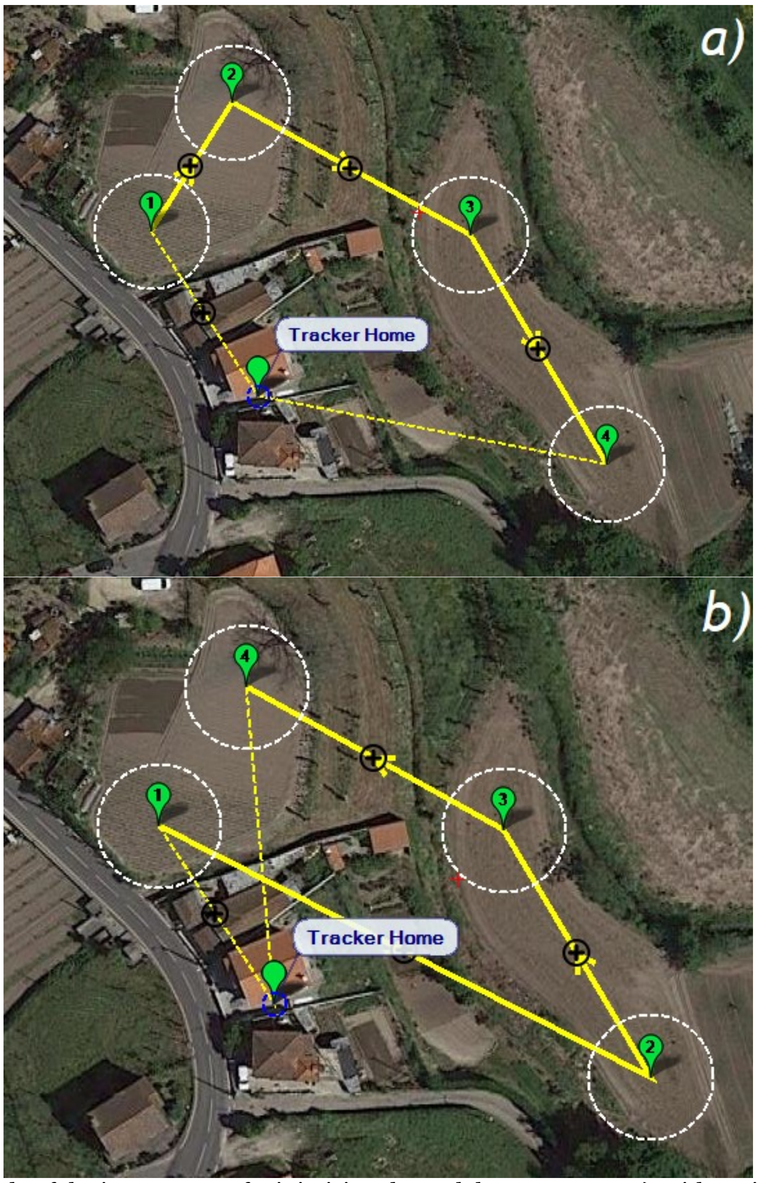


Figure 13 – Example of the importance of minimizing the path between PoIs: a) With optimization function b) Without optimization function.

4.4. Maximization of Random Waypoints

After planning the order of the PoIs, the proposed algorithm will calculate the distance available to the random waypoints subtracting the minimum path between PoIs from the total distance for path planning. To calculate the number of points, the maximum number of points one meter apart is calculated with the remaining distance. Then, the ratio between the total number and the incidence rate is made to find the individual number for each PoI. If the farmer uses bird detection sensors, it can indicate the percentage of detections per PoI. Otherwise, the incidence rate must be inserted manually, as mentioned above, through the parameters. These values can be entered into an input if the `Check` button is disabled and is assigned from zero (there is no damage and the PoI does not need points) to five (severe damage, the maximum number of points need to be generated). This scale ensures differentiation in the number of random waypoints between PoIs. Still, if the farmer needs more diversity, he can introduce any maximum value because the incidence rate is calculated by dividing the value by the sum of the total values assigned, obtaining a unit scale. Then, random waypoints around the PoI will be generated based on the method explained in Subsection 3.2.3, where the radius of the circle is the random waypoints radius parameter inserted by the user in the GUI. After each waypoint is generated PSO will be applied again but in this case for maximization. Each particle corresponds to a sequence of random waypoints and for the objective function the Haversine Formula is used again where the distance from the first and last points are tested with the position of the PoI itself. In the end, the sum of all distances is performed and compared with the total distance for path planning. If this value is within the acceptable interval with the final error defined by the user, a mission plain-text file format file will be generated with all geographical coordinates. Otherwise, the algorithm will compare whether the difference is greater or less than the final error and will adjust the total number of waypoints based on this factor, generating new waypoints, and rerun the PSO maximization.

4.5. Creation of Pre-Planned Mission File

In the end, the Python programming languages file handling functions are used to write all waypoints parameters. If there is a file in the predefined folder with the same name inserted on the parameter settings, the path planning optimization algorithm will open it and writing over; otherwise, it creates a new file. The Plain-text file format is a standard applied in many GCS and developer APIs for storing mission information. The data included: mission plans, geofence definitions, rally points, parameters, logs, among others. The first line contains the file format and version information, while subsequent the line(s) are mission items in each column [90]. Table 3 lists the columns and their parameters. The spaces between the fields above are <tab> or 8 spaces (\t Python programming languages).

Table 3 - Mission plain-text file format column parameters.

Column	Representation
1	<INDEX>
2	<CURRENT WP>
3	<COORD FRAME>
4	<COMMAND>
5	<PARAM1>
6	<PARAM2>
7	<PARAM3>
8	<PARAM4>
9	<PARAM5/X/LATITUDE>
10	<PARAM6/Y/LONGITUDE>
11	<PARAM7/Z/ALTITUDE>
12	<AUTOCONTINUE>

4.6. Final Remarks

As mentioned in previous chapters, although optimization algorithms for path planning are not new, their use in UAVs to control bird damage to fruit crops problems in orchards is. So far, this is the only known algorithm with this objective. Despite the path planning optimization algorithm for semi-autonomous UAVs in bird repellent systems based on PSO is presented in more detail in this dissertation and with the newest features, some work is available in Mesquita and Gaspar [91], and some simulation results were presented. The original algorithm has some disadvantages compared to the final version presented in this dissertation since parameters were inserted on the command line, and the processing time was longer because the disk point picking technique was not used, among others. Since then, it has been improved to obtain better results, become faster, and be more user-friendly. The latest version can be accessed on GitHub with the flowing link: https://github.com/RJMesquita/Path-Planning-Optimization-Algorithm-for-Semi-Autonomous-UAV-in-Bird-Repellent-Systems.git. The path planning optimization algorithm has the advantages of its graphical interface that facilitates its use for producers, scalability being built by functions, compatibility with Mission Planner, and other used GCS. Notwithstanding having the objective of optimizing paths for bird damage to fruit crops, being one of its main features, the random generation of waypoints can be applied to optimize flights with PoIs in general.

5. Analysis and Discussion of Results

5.1. Introduction

Three case studies were developed to study the novel path planning optimization algorithm (two in simulation and one in the field). Case study #1 uses several PoIs with a low radius, and Case study #2 has fewer PoIs, with a high random waypoint's radius to serve as a comparison. Both simulations are used to understand better the algorithm's performance and the influence of each parameter. The Case study #3 is developed in the field to analyze the real-world application and the impact on drone battery savings and height influence. A peach orchard in Orjais, Covilhã, in Portugal, marked in red line in Figure 14, was used as a search field in both simulation and real-world tests.

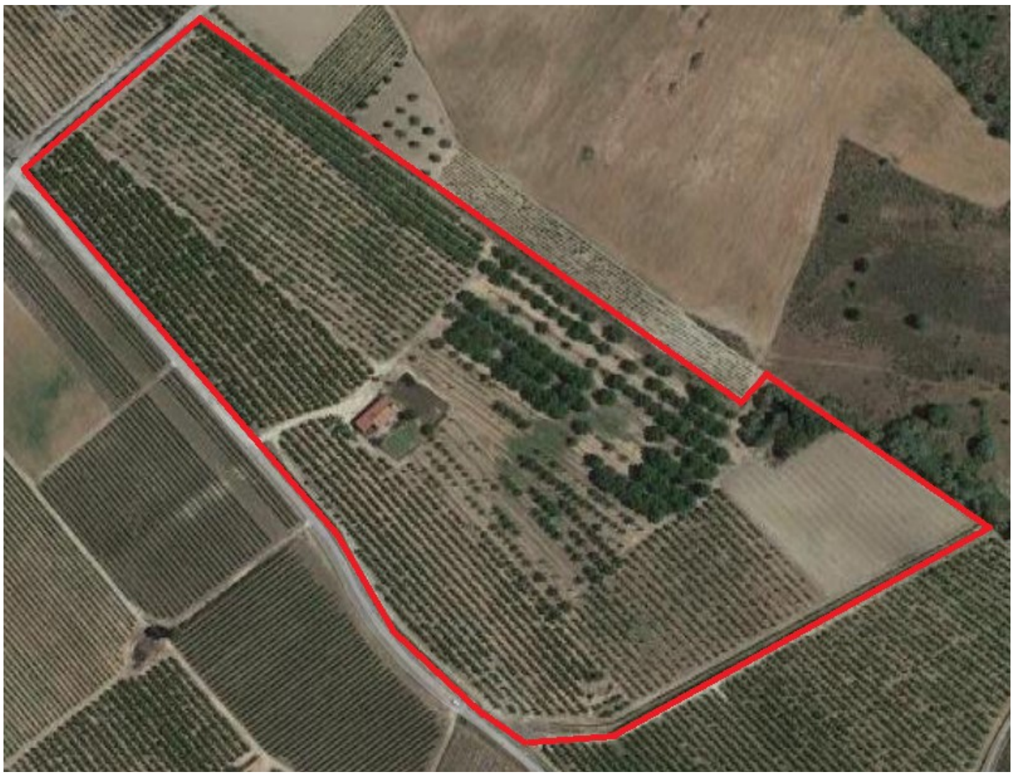


Figure 14 – Search study field.

During simulations, all values chosen are real-life representations of each parameter, and during their individual study, the limits and average values were used. The parameter variations were tested in the three possible scenarios: same incidence rate, different rates, and different rates with no bird damage. Because it is a simulation, random vectors of damage were generated. In Scenario 1, the same number of waypoints were assigned to each PoI. In Scenario 2, a random vector of numbers of incidence rate, on a scale between one and five. In Scenario 3, within the same vector as Scenario 2, some PoIs became zero because they did not have bird incidence. Due to the heuristic nature of the PSO, all cases run thirty-five times, and the total distance for path planning obtained by the optimization algorithm in meters, the total number of waypoints, the number of iterations that the code had to re-run to find a solution within the acceptance range, and the execution time in seconds, were collected, and the average, maximum and minimum values were calculated. In all cases, the PyCharm Community Edition as an interpreter on Windows 10 of 64 bits were used to run the proposed algorithm, on a computer with an Intel Core i7-6700HQ CPU and 16 GB RAM. Table 4 shows the initial configuration parameters used in each PSO, assigned through the study of convergence curves to provide the best overall performance to the path planning optimization algorithm.

Table 4 – PSO initial configuration parameters.

Parameter	Minimization	Maximization
c_1	2	2
c_2	2	2
w	Random Inertia Weight	Random Inertia Weight
Number of Particles	5	5
Initial Velocity	Random	Random
$itmax$	50	200

It is noteworthy the same value for c_1 and c_2, giving equal weight to the experience of the individual and the group, the low number of particles so that the algorithm process faster, and different values in the maximum number of iterations due to the complexity of each problem. The Random Inertia Weight, shown in Equation 14, was selected due to being the best for efficiency [92].

$$w = 0.5 + \frac{random()}{2} \tag{14}$$

5.2. Case study #1

The goal of the first study case was to understand the performance of the algorithm with multiple PoI and low maximum random waypoints radius, varying each parameter (total distance for path planning, final error, and take-off and landing position). To cover the entire field of study, thirty-nine PoIs were chosen with random waypoints radius of twenty meters. Figure 15 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

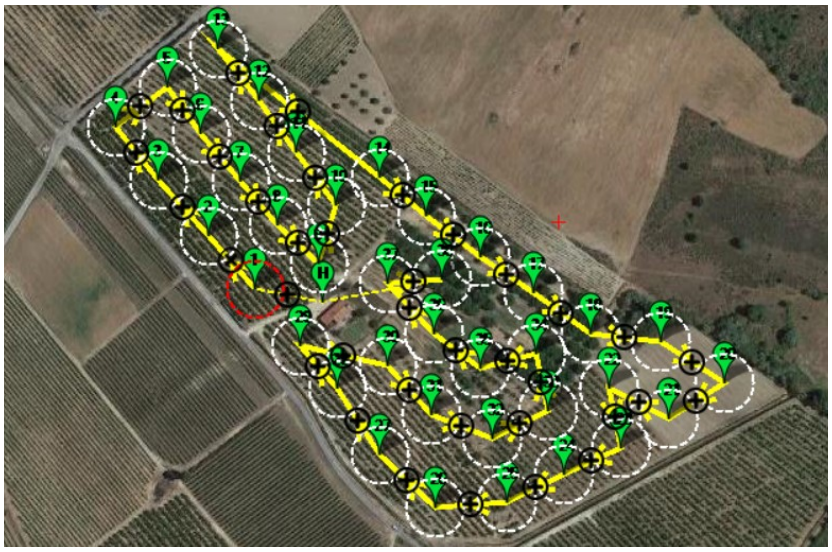


Figure 15 – PoIs of the Case study #1.

As mentioned above, in all simulations, the three possible cases were performed with the same incidence rate vectors represented in Table 5.

Table 5 – Incidence rate vectors for Case study #1.

Scenario	Incidence rate vector
Scenario 1	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
Scenario 2	[3,4,1,2,4,4,2,1,5,2,3,2,4,5,5,3,1,1,2,4,2,4,3,1,2,5,2,5,1,5,2,4,2,1,3,1,2,4,3]
Scenario 3	[0,4,1,2,4,4,2,0,5,2,3,2,4,5,5,0,0,1,2,4,2,4,0,1,2,5,2,5,1,0,2,4,2,0,3,0,2,4,3]

The first parameter tested was the total distance for path planning. The values were calculated through the mission speed and the flight time. Since Li-Po batteries last an average between 20 to 30 minutes [71], with the Mission Planner's default speed of 500 cm/s, the average values for the total distance for path planning were between 6,000 and 9,000 meters. Since the total distance for the path planning to be varied, the final error was constant at 5% of the total distance for path planning, and the same take-off and landing waypoint was used, positioned on the center of the field. Table 6 shows the results obtained in the variation of total distances for path planning. Regarding the total distance for path planning, there is nothing to point out. All scenarios for both distances presented identical results, with Scenario 2 having reached an average error in total distance for path planning obtained from both cases of less than 1% of the original value, this being the best result. Fewer points were obtained in the total number of waypoints for the 6,000 meters distance, which is expected as this value is related to the minimum distance between PoIs and the total distance for path planning. Notice that at both distances for Scenario 1, the total number of waypoints did not change in all simulations. It is also essential to highlight the number of iterations in Scenario 1 of the total distance for path planning of 6,000 meters in which it was difficult for the algorithm to find a good distance value, having to rerun an average of 13 times, having a maximum of 96 iterations, much higher than the value presented in the other scenarios. This occurred because the algorithm has an equal number of waypoints across all PoIs with a lower total distance. Concerning the execution time, it can be observed that the greater the total distance, the longer the time will be, as it takes longer to process more waypoints. In Scenario 3 of the total distance for path planning of 9,000 meters, the time is much longer than the other two scenarios. Since there are fewer PoIs, the algorithm must calculate more points within the remaining points of interest.

Table 6 – Results from the variation of total distances for path planning.

	Average	Maximum	Minimum
Total Distance for Path Planning Obtained by the Optimization Algorithm [meters]			
Total Distance for Path Planning: 9,000 meters			
Scenario 1	8,890.980	9,225.393	8,580.921
Scenario 2	8,932.331	9,320.805	8,567.005
Scenario 3	8,849.140	9,117.971	8,558.183
Total Distance for Path Planning: 6.000 meters			
Scenario 1	6,185.257	6,296.571	5,898.484
Scenario 2	5,951.177	6,298.591	5,741.981
Scenario 3	5,936.690	6,295.887	5,702.002
Total Number of Waypoints			
Total Distance for Path Planning: 9,000 meters			
Scenario 1	236	236	236
Scenario 2	233	255	210
Scenario 3	253	264	248
Total Distance for Path Planning: 6,000 meters			
Scenario 1	119	119	119
Scenario 2	110	125	100
Scenario 3	133	156	120
Number of Iterations			
Total Distance for Path Planning: 9,000 meters			
Scenario 1	4	8	4
Scenario 2	7	8	6
Scenario 3	7	11	7
Total Distance for Path Planning: 6,000 meters			
Scenario 1	13	96	2
Scenario 2	2	3	2
Scenario 3	5	9	4
Execution Time [seconds]			
Total Distance for Path Planning: 9,000 meters			
Scenario 1	10.562	32.767	9.021
Scenario 2	16.590	21.884	11.637
Scenario 3	28.853	54.526	24.333
Total Distance for Path Planning: 6,000 meters			
Scenario 1	8.681	53.904	1.625
Scenario 2	1.823	2.960	1.293
Scenario 3	5.226	11.940	3.407

Then the final error was tested, and the results are shown in Table 7. In the beginning, the values of 10% and 1% of the total distance for path planning were used. After performing some tests with the final error at 1%, it was noticed that the optimization algorithm was very slow and quickly blocked in the random waypoints function, needing improvement in future works. Not becoming a valid comparison, the final error was increased until obtaining comparable results, reaching a value of 3%. Since it is the final error to be varied, the total distance for path planning was constant at 7,500 meters, and the same take-off and landing position was used, positioned on the center of the field. Related to the total distance for path planning obtained by the optimization algorithm, although a final error of 10% was used, the average values in the three scenarios have a maximum less than 7%. When the final error is 3%, all scenarios present in average final errors values of less than 1% of the original total distance for path planning. The variation of the final error did not affect the total number of waypoints, being that for both cases and all scenarios, the values were identical. As expected, between the two cases, there was a significant discrepancy in the number of iterations and, consequently, the execution time, since the smaller the error, the longer it takes for the optimization algorithm to find a value of the total distance for the path planning within the acceptance range. Note that with a final error of 10%, the optimization algorithm was faster in the scenarios in the order of 3-2-1 and that in the other case, the opposite occurred because there is a greater total distance for path planning and needs to be divided by more PoIs, generating more random waypoints per PoI. In conclusion, the optimization algorithm with a final error of 3% becomes impractical for scenarios 1 and 2.

Table 7 – Results from the variation of final error.

	Average	Maximum	Minimum
Total Distance for Path Planning Obtained by the Optimization Algorithm [meters]			
Final error: 10% of the total distance for path planning			
Scenario 1	7,136.907	7,945.977	6,757.306
Scenario 2	7,099.347	7,753.496	6,751.171
Scenario 3	7,014.388	7,611.317	6,752.426
Final error: 3% of the total distance for path planning			
Scenario 1	7,528.572	7,721.526	7,275.601
Scenario 2	7,556.262	7,724.228	7,284.137
Scenario 3	7,506.082	7,723.167	7,290.456
Total Number of Waypoints			
Final error: 10% of the total distance for path planning			
Scenario 1	163	197	119
Scenario 2	159	190	145
Scenario 3	178	196	156
Final error: 3% of the total distance for path planning			
Scenario 1	184	197	158
Scenario 2	179	190	151
Scenario 3	199	212	172
Number of Iterations			
Final error: 10% of the total distance for path planning			
Scenario 1	3	4	2
Scenario 2	5	6	4
Scenario 3	5	6	4
Final error: 3% of the total distance for path planning			
Scenario 1	72	737	3
Scenario 2	45	270	5
Scenario 3	6	12	5

Table 7 – Results from the variation of final error (cont).

Execution Time [seconds]			
Final error: 10% of the total distance for path planning			
Scenario 1	3.924	6.604	1.797
Scenario 2	6.067	9.444	4.113
Scenario 3	9.718	13.002	5.782
Final error: 3% of the total distance for path planning			
Scenario 1	203.933	3452.745	3.377
Scenario 2	83.237	514.559	5.633
Scenario 3	15.678	31.765	12.022

The last parameter varied, ending Case study #1, was the take-off and landing position. Table 8 shows the results obtained. Two cases were used to test this parameter. One with the same take-off and landing position placed in the center of the field and the other with different take-off and landing position located at opposite sites in the field. Since the take-off and landing position was varied, both the total distance for path planning and the final error were kept constant at 7,500 meters and 5%, respectively. During the test of this parameter, all values remained identical for the two cases in all situations, except for the total distance for path planning in different take-off and landing positions, having presented a more significant discrepancy to the original value. Since the maximum and minimum values were identical for all scenarios in both cases, it is possible to indicate that there was only a higher variation in the simulation values.

Table 8 – Results from the variation of take-off and landing position.

	Average	Maximum	Minimum
Total Distance for Path Planning Obtained by the Optimization Algorithm [meters]			
Same take-off and landing position			
Scenario 1	7,508.493	7,872.638	7,131.713
Scenario 2	7,551.033	7,866.488	7,138.796
Scenario 3	7,473.269	7,785.815	7,128.395
Different take-off and landing position			
Scenario 1	7,219.699	7,871.990	7,125.665
Scenario 2	7,524.375	7,868.870	7,141.019
Scenario 3	7,399.987	7,786.099	7,142.777
Total Number of Waypoints			
Same take-off and landing position			
Scenario 1	178	197	158
Scenario 2	179	190	145
Scenario 3	198	212	172
Different take-off and landing position			
Scenario 1	177	197	158
Scenario 2	174	190	151
Scenario 3	194	212	172
Number of Iterations			
Same take-off and landing position			
Scenario 1	6	22	3
Scenario 2	7	15	4
Scenario 3	6	8	5
Different take-off and landing position			
Scenario 1	7	27	3
Scenario 2	7	26	5
Scenario 3	6	6	5

Table 8 – Results from the variation of take-off and landing position. (cont)

	Average	Maximum	Minimum
Execution Time [seconds]			
Same take-off and landing position			
Scenario 1	10.889	53.069	3.325
Scenario 2	10.995	28.920	4.323
Scenario 3	14.064	18.804	9.316
Different take-off and landing position			
Scenario 1	12.168	60.809	3.200
Scenario 2	11.673	49.451	5.683
Scenario 3	14.098	19.210	9.860

5.3. Case study #2

After understanding the performance of the optimization algorithm with the variation of each parameter in the first case study, it was developed a second case to acknowledge the difference in the execution of the algorithm with different numbers of PoIs and random waypoint radius. In contrast to the previous case, fewer PoIs were selected with a higher radius of random waypoints to cover the entire field. To cover the entire field of study, ten PoIs were chosen with random waypoints radius of fifty meters. Figure 16 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

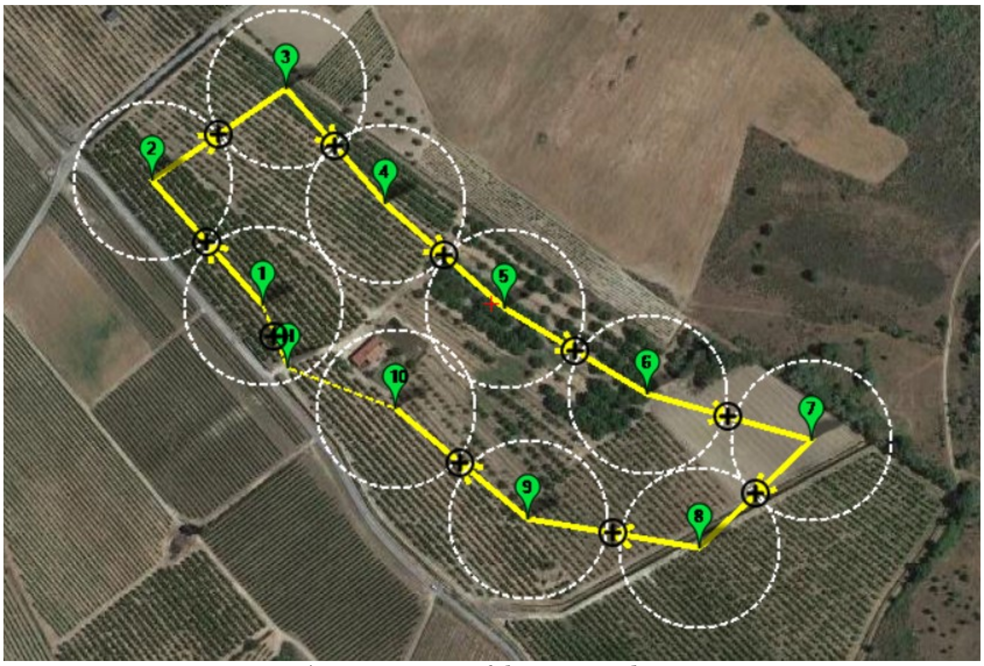


Figure 15 – PoIs of the Case study #2.

In this case study, the performance of the optimization algorithm of the PoIs presented in Figure 15, named hereafter as the case with lower PoIs, and was compared with the PoIs in Figure 16, referred henceforth as the case with higher PoIs. As mentioned above, in all simulations, the three possible cases were performed with the same incidence rate vectors represented in Table 9.

Table 9 – Incidence rate vectors for Case study #2.

Scenario	Incidence rate vector
Scenario 1	[1,1,1,1,1,1,1,1,1,1]
Scenario 2	[3,4,1,2,4,4,2,1,5,2]
Scenario 3	[0,4,1,2,4,4,2,0,5,2]

The average parameters tested in the previous case for both PoIs schemes were kept, along with the test variables (total distance for path planning obtained by the optimization algorithm, total number of waypoints, number of iterations, and execution time). The values corresponding to the parameters were: 7,500 meters for the total distance for path planning, final error of 5%, and the same take-off and landing point. Table 10 shows the results obtained with the PoIs of both cases. The total distance for path planning obtained by the optimization algorithm shows similar values on average, with the case with lower PoIs presenting better results, being each one below 1% of the original value. The case with higher PoIs has more total number of waypoints in all scenarios because the random waypoints are closer to the PoI, so there is less flight distance between the random waypoints. Thus, the path planning optimization algorithm can generate more points. The variables most negatively affected by the reduction in the number of PoIs and consequent increase in the radius of the random waypoint are the number of iterations and the execution time. Clearly, in this case, the optimization algorithm becomes impracticable due to being very slow. One thing to note is the low number of iterations per second of execution compared to the Case study #1 situation with a final error of 3%. The execution time is identical, but there is more number of iterations. This can be explained because the algorithm takes longer to reduce the total number of waypoints until it has a viable value.

Table 10 – Results from Case study #2.

	Average	Maximum	Minimum
Total Distance for Path Planning Obtained by the Optimization Algorithm [meters]			
Higher Number PoIs			
Scenario 1	7,498.936	7,874.698	7,140.985
Scenario 2	7,599.392	7,873.370	7,146.384
Scenario 3	7,419.634	7,840.074	7,125.564
Lower Number PoIs			
Scenario 1	7,556.118	7,873.750	7,161.054
Scenario 2	7,511.199	7,846.897	7,130.927
Scenario 3	7,563.032	7,851.434	7,128.681
Total Number of Waypoints			
Higher Number PoIs			
Scenario 1	180	197	158
Scenario 2	179	190	145
Scenario 3	198	212	172
Lower Number PoIs			
Scenario 1	126	132	122
Scenario 2	126	134	138
Scenario 3	129	138	118

Table 10 – Results from Case study #2 (cont.).

	Average	Maximum	Minimum
Number of Iterations			
Higher Number PoIs			
Scenario 1	6	18	3
Scenario 2	7	21	4
Scenario 3	6	6	5
Lower Number PoIs			
Scenario 1	21	24	20
Scenario 2	16	18	14
Scenario 3	25	27	24
Execution Time [seconds]			
Higher Number PoIs			
Scenario 1	11.570	37.737	3.448
Scenario 2	10.222	34.172	4.059
Scenario 3	12.806	16.423	9.468
Lower Number PoIs			
Scenario 1	209.111	232.803	197.770
Scenario 2	175.687	200.649	146.040
Scenario 3	408.003	443.052	371.005

5.4. Case study #3

Case study #3 was developed to understand the influence of the path planning optimization algorithm and the drone's flight height (wind influence) on the battery life and corresponding flight time. Two cases were then developed, one at the height of 10 meters, near the treetops, and the other at 20 meters. The preliminary tests of this case study were performed in a peach tree field in Orjais, Portugal.

The variables selected to study the influence of the path planning optimization algorithm were: the voltage difference between the initial and final voltage, measured through the voltage sensor of the power brick, the capacity discharged from the batteries, calculated through the battery's charger (Ultramat AC/DC EQ) in mAh, and the flight time in seconds obtained from the flight records. Field tests with a new PoI scheme were performed in a cornfield in Macieira, Lousada, Portugal. Following the exact measurements used in the peach tree field, eight PoIs were then used to ensure equality, with a random waypoint radius of twenty meters. A short path was created so that the environment was as controlled as possible, always keeping the drone in line of sight for security. Figure 17 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

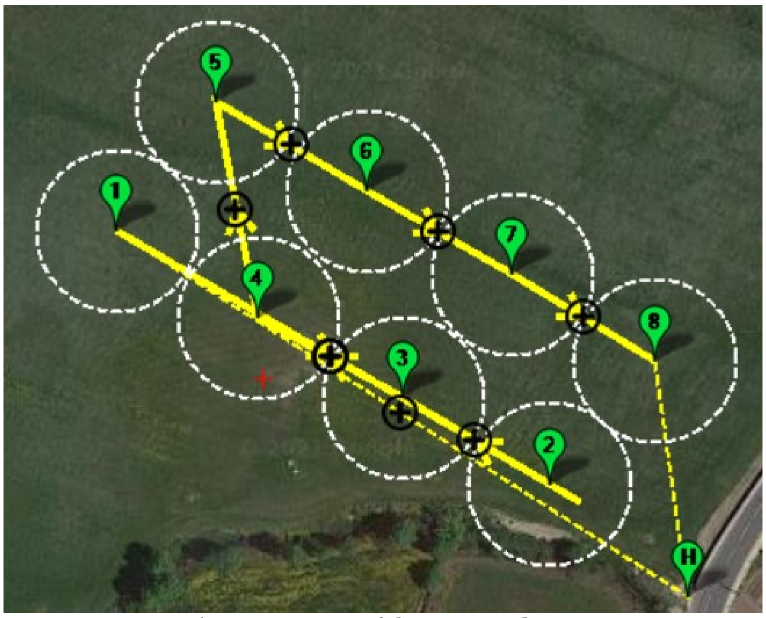


Figure 17 – PoIs of the Case study #3.

The field tests for all cases were also performed in the three possible cases with the same incidence rate vectors represented in Table 11.

Table 11 – Incidence rate vectors for Case study #3.

Scenario	Incidence rate vector
Scenario 1	[1,1,1,1,1,1,1,1]
Scenario 2	[3,4,1,2,4,4,2,1]
Scenario 3	[0,4,1,2,4,5,2,0]

For both cases, the values corresponding to the parameters were: 1,500 meters for the total distance for path planning, final error of 5%, and the same take-off and landing point. It is also important to mention that the Mission Planner default speeds were all kept. In order to obtain statistical data, four flights were performed per scenario. To achieve a more accurate measurement, the same battery should have been used during all flights, but this is not practical, so four batteries of the same brand with the same configuration were used. It must be highlighted that no battery was repeated per scenario and that the flights were carried out over three days at different times sequentially to ensure data reliability. Table 12 shows the results in the field obtained from the quadcopter studied. After analyzing the results obtained, it is possible to indicate that scenario 3 in both cases presents lower voltage difference and capacity discharged from the batteries, although this does not directly transpose into more flight time. This result arises from the fewer corrections between PoIs made by the UAV. Another consideration is that although the optimization algorithm does not use height as a parameter in path planning, the results show more flight time for 10 meters because the cornfield was covered with trees that protected the drone at lower altitudes. It is also important to indicate that there is no direct connection between the discharged capacity of the batteries and the flight time. One would expect less flying time to discharge less battery capacity, but since drones depend on factors such as wind, this doesn't always happen. It is also possible to establish that there is no relationship between the mission velocity and the flight time, considering that for 1,500 meters, an expected 5 minute mission should be obtained, and the time varies between 7 to 14 minutes approximately.

Table 12 – Results from Case study #3.

	Average	Maximum	Minimum
Voltage Difference [V]			
Height: 10 meters			
Scenario 1	1.17	1.23	1.12
Scenario 2	1.23	1.25	1.20
Scenario 3	1.09	1.15	1.02
Height: 20 meters			
Scenario 1	1.22	1.29	1.18
Scenario 2	1.25	1.32	1.16
Scenario 3	1.08	1.19	0.89
Capacity Discharged from the Batteries [mAh]			
Height: 10 meters			
Scenario 1	1,662	1,790	1,499
Scenario 2	1,804	1,834	1,743
Scenario 3	1,569	1,636	1,491
Height: 20 meters			
Scenario 1	1,763	1,872	1,626
Scenario 2	1,837	1,985	1,642
Scenario 3	1,497	1,684	1,127
Flight Time [seconds]			
Height: 10 meters			
Scenario 1	599	643	571
Scenario 2	724	820	674
Scenario 3	588	676	540
Height: 20 meters			
Scenario 1	612	701	437
Scenario 2	742	810	621
Scenario 3	722	839	652

5.5. Final Remarks

In Orjais, Covilhã, Portugal, a peach orchard was selected as a search field to test the novel path planning optimization algorithm. Three study cases, two in simulation and one in the field, were developed to understand the performance of the optimization algorithm, the parameter variation, and the influence on battery management. All parameter variations were tested in the three possible scenarios: same incidence rate, different rates, and different rates with no bird damage. During testing, the average, maximum, and minimum of the total distance for path planning obtained by the optimization algorithm, the total number of waypoints, the number of iterations, and the execution time were collected. Case study #1 used more PoIs and a low random waypoints radius, where all input parameters were varied to understand how the optimization algorithm performed. In Case study#2, a

new PoIs scheme was created with lower number PoIs and a higher random waypoints radius, which was compared to the previous case study. Case study #3 was the field test with the quadcopter described in Chapter 3. During the tests, the height varied in the three possible scenarios and the voltage difference, discharged battery capacity and time were evaluated.
After analyzing the results obtained in Case study #1, it is possible to infer that the algorithm guarantees an outstanding average error of the total distance for path planning, having a maximum error of 7%, where the final error was set at 10%. However, the average error in most examples of the first case (final error at 5%) was 1.3% of the original value of the total distance for path planning. It is also possible to indicate that this algorithm has a reasonable execution time. The final error is the parameter that most influences the results, and the smaller the error, the longer will be the running time, with an average of almost 2 minutes when the error is 3%. For all cases, when the final error is 5%, the average execution time is 12 seconds. In the end, it can be observed that there is a relationship with the variation of the parameter of the total distance for path planning, the number of waypoints, and the execution time. The greater the distance, the greater the other two parameters are. Important to also observe with Case study #1 is that the takeoff and landing position does not influence the other parameters.
Case study #2 shows that the algorithm with fewer PoIs and a larger random waypoint radius increases the execution time of the proposed algorithm, making it very slow with an average execution time of 4 minutes and 24 seconds. This result reveals that will be necessary in the future to create a ratio between the number of PoIs and the total distance for path planning or improve the function that calculates the number of random waypoints. From Case study #2 results, it is also possible to verify that the smaller the number of PoIs, the greater the total number of waypoints. This result comes from the lower distance between the PoIs and because the path planning optimization algorithm can generate more waypoints.
The results in Case study #3 show that fewer PoIs (Scenario 3) causes a lower value in the discharge of the battery capacity with an average of 1,533 mAh comparing to 1,712 mAh and 1,820 mAh of the other two scenarios, a factor that is not directly related to the flight time. The ratio between the mission velocity and the flight time does not relate as anticipated, being expected 5 minutes in mission time and some flights reached 14 minutes. One of the most important conclusions derived from Case study #3 is that height does not influence the flight, and the most influencing factor is terrain and weather conditions.
In conclusion, all case study present satisfactory results, especially the error in the total distance for path planning obtained and the execution time. With some modifications in the future, the proposed algorithm can be tested by producers.

6. Conclusions

6.1. General Conclusions

According to The World Back [93] , agriculture is crucial to economic growth. In 2018, it accounted for 4% of global gross domestic product (GDP), and in some developing countries, it can account for more than 25% of GDP. It is, therefore, crucial to reduced production losses for farmers worldwide with new techniques and technologies. Despite not being recent, a concept that still has much attention from companies and researchers is precision agriculture (PA). PA is a management strategy that employs information technology to improve agricultural quality and production, differing from traditional farming in the sense that this process accurately identifies variations and relates the spatial data to management activities [94]. Bird damage to fruit crops is a significant concern for farmers, causing millions of dollars in lost yield each year. In addition to consumption, a large number of birds can damage fruit, leaving it susceptible to infection and reducing its quality [8]. Although this problem is general worldwide, this work was based and tested on the needs of peach and cherry farmers in the region of Covilhã, Portugal. Most still use traditional methods such as netting, planting, and harvesting manipulation, with the bird cannons and loudspeakers, which are the most technological systems but still primitive. Birds are unique pests because they are highly mobile, resulting in greater spatial and temporal variation in damage levels than mammalian pests. Today, most systems employees are not very mobile and are predictable over time, becoming ineffective in the long run. UAVs have advantages in their versatility and low maintenance, making them potential solutions to this problem when combined with repelling systems. One of the most notable disadvantages is its limited battery capacity turned into low flight time, being essential to improve the efficiency of the mission by planning the path according to the problem.

In this dissertation is presented a novel path planning optimization algorithm for semi-autonomous UAV in bird repellent systems based in Particle Swarm Optimization, developed in Python programming language, with the objective of maximizing the path and randomly generating waypoints according to the bird damage. Nature has been continuously solving challenging problems using evolution. Therefore, it is reasonable to be inspired by nature to solve different challenging problems. Swarm intelligence and evolutionary computation are searching methods based on the physical behavior and natural evolution of social intelligence development of real animals and others in real life [95]. These techniques can optimize complete systems while maintaining the simplicity and efficiency of other algorithms. One of these techniques is Particle Swarm Optimization which is a population-based self-adaptive, stochastic optimization technique [96], used for its performance, accuracy in solving optimization problems, easy implementation, and adaptation [97]. A method to calculate the distance between two geographic coordinates had to be applied in the path planning optimization algorithm, and after research, the haversine formula was used. Another essential method applied is the generation of random geographic coordinates within a circle with disk point picking. To test the mentioned algorithm, it was necessary to build and configure a UAV capable of flying through a pre-planned mission, using autopilots flight controllers and ground control stations software. Despite all the information about drone building and each subsystem within it, the configuration was one of the most challenging aspects. An incorrect or poorly calibrated parameter can cause it not to fly correctly or even crash during flight. UAVs of any structure can be dangerous, so it is necessary to fly safely with proper precautions. During this work, all flights were carried out in a controlled, safe environment, with the drone always on the line of sight.

The path planning optimization algorithm presented in this dissertation can be divided into four main steps: Parameter Setting, using a graphical interface the user can insert all the necessary parameters; Minimization Between PoIs, ensuring that the UAV wastes the minimum time between sensors; Maximization of Random Waypoints, random waypoints are generated around the PoI, accordingly to the bird damage and their path is maximized; Creation of Pre-Planned Mission File, a file is created to be read by the GCS and transferred to the autopilot. As to the author's knowledge, this is a novel path planning optimization algorithm, so it is impossible to compare it with other algorithms. So, three case studies were created to understand the performance and parameter variation of the proposed algorithm. All cases were tested in three possible situations: same incidence rate, different rates, and different rates with no bird damage. Case Study #1 was in simulation and used 32 PoIs with 20 meters random waypoints radius and was done to understand how the proposed algorithm performed when each parameter was varied. In the second case, Case Study#2, also in simulation, a new PoIs scheme was created with 10 PoIs, and 50 meters as random waypoints radius and was compared to the previous case study. For last, a field test with the quadcopter was done. During Case Study #3, the height was varied in the three possible scenarios, and the voltage difference, discharged battery capacity, and time was evaluated. It should be mentioned that, although the focus of this dissertation is on path optimization and not the effectiveness of the path planning optimization algorithm to repealed birds, during the field case study, birds ended up disappearing.

In conclusion, this algorithm intends to overcome the failures of traditional systems in bird damage to fruit crops used by producers today, optimizing UAV flights, distributing points according to the bird damage, and creating random waypoints so that they do not detect patterns. Also, although it is aimed at this agricultural problem, the algorithm can be modified for other scenarios and problems and adapted to any autopilot or GCS.

6.2. Specific Conclusions

Case study #1 was used to understand the performance of the proposed algorithm with the variation of each parameter. After examining the first case, it is possible to indicate that the algorithm has an overall satisfactory performance, obtaining an excellent average error in the total distance for path planning than the original value and a reliable execution time for most cases. After varying each parameter, it is also possible to infer that the total distance for path planning directly affects the number of waypoints and the execution time. The final error influences the total distance for path planning and the number of iterations that affect the execution time per consequence. Finally, it is also possible to indicate that the takeoff and landing position parameter has practically no influence on any other parameter.

The Case study #2 was developed to test the performance in different scenarios, using two PoIs schemes. The first had higher number of PoIs with lower random waypoint radius and the second one a smaller number of PoIs with higher random waypoint radius. Case study #2 shows that decreasing the number of PoIs with a consequent increase in the random waypoint radius can lead to an impractical use of the optimization algorithm, directly affecting the execution time. This can be interpreted as a bad ratio between the total distance for path planning and the number of PoIs and that the function that generates the number of random waypoints needs to be changed. It was also shown that with fewer number of PoIs, the number of total waypoints is fewer than with higher number of waypoints.

Case study #3 was a field test with the studied quadcopter. This case is very important to understand the influence of the proposed algorithm in battery capacity and flight time. After analyzing the results obtained for the last case, it is possible to infer that fewer PoIs cause a reduction in the discharge of the battery capacity, a factor that is not directly related to the flight time. That flight height does not influence the flight, and the most influencing factor is the terrain and weather conditions. There is no established relationship between the mission velocity (maximum horizontal speed) and the flight time, and this function needs to be changed in future works. It is impossible to assign a quality level to the path planning optimization algorithm since there is no relative term. However, it is reasonable to state that the results are generally satisfactory due to execution time and functionality, and the implementation of this system in real-world applications is viable.

6.3. Suggestions for Future Work

As already mentioned, the optimization algorithm presented is still being improved, and at each iteration, a new functionality or performance improvement is added. After completing this dissertation, it is possible to suggest some possibilities for improvements and future work. Due to the Python programming language being high level, it is possible to work quickly and integrate systems more effectively than other programming languages such as C or Rust. Although this language gains in development time, it loses in execution time because code is interpreted at runtime instead of being compiled to native code at compilation. Future work may be related to implementing and comparing this path planning optimization algorithm in other programming languages to ensure faster execution time. Optimization techniques other than PSO should also be implemented and tested. These approaches should improve the execution time and quality of the results obtained by the path planning algorithm. After the field tests, the programming function that relates the mission velocity and the flight time and calculates the total distance for the path planning must be modified considering other speeds besides the maximum horizontal speed. Without this improvement, the farmer must be very conservative when inserting the flight time. More field tests should be carried out with more drones and batteries, where the power consumption and autonomy are studied. The efficiency of the proposed algorithm in bird damage to fruit crops problem to must also be evaluated. Features such as compatibility with other GCS and variable random waypoint radius must also be added to complete the characteristics of the proposed algorithm. It is also necessary to establish a relationship between the total distance for the path planning and the number of PoIs and a better function to control the number of generated waypoints, avoiding a slow algorithm.

References

[1] A. R. Anik, S. Rahman, and J. R. Sarker, “Five decades of productivity and efficiency changes in world agriculture (1969–2013),” *Agric.*, vol. 10, no. 6, pp. 1–21, 2020, doi: 10.3390/agriculture10060200.

[2] F. J. Pierce and P. Nowak, “Aspects of Precision Agriculture,” *Adv. Agron.*, vol. 67, no. C, pp. 1–85, 1999, doi: 10.1016/S0065-2113(08)60513-1.

[3] P. Daponte *et al.*, “A review on the use of drones for precision agriculture,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 275, no. 1, 2019, doi: 10.1088/1755-1315/275/1/012022.

[4] M. Kulbacki *et al.*, “Survey of Drones for Agriculture Automation from Planting to Harvest,” *INES 2018 - IEEE 22nd Int. Conf. Intell. Eng. Syst. Proc.*, pp. 000353–000358, 2018, doi: 10.1109/INES.2018.8523943.

[5] “AGRAS T20 - DJI.” Accessed: May 03, 2021. [Online]. Available: https://www.dji.com/pt/t20.

[6] S. B. Canavelli, L. C. Branch, P. Cavallero, C. González, and M. E. Zaccagnini, “Multi-level analysis of bird abundance and damage to crop fields,” *Agric. Ecosyst. Environ.*, vol. 197, pp. 128–136, 2014, doi: 10.1016/j.agee.2014.07.024.

[7] M. B. Hannay *et al.*, “Bird species and abundances in fruit crops and implications for bird management,” *Crop Prot.*, vol. 120, no. September 2018, pp. 43–49, 2019, doi: 10.1016/j.cropro.2019.02.015.

[8] J. L. Elser *et al.*, “Measuring bird damage to three fruit crops: A comparison of grower and field estimates,” *Crop Prot.*, vol. 123, no. December 2018, pp. 1–4, 2019, doi: 10.1016/j.cropro.2019.05.010.

[9] D. O. Clark, “An overview of depredating bird damage control in California,” *Bird Control Semin. Proc.*, no. November, pp. 21–27, 1976.

[10] G. M. Linz, H. J. Homan, S. J. Werner, H. M. Hagy, and W. J. Bleier, “Assessment of bird-management strategies to protect sunflowers,” *Bioscience*, vol. 61, no. 12, pp. 960–970, 2011, doi: 10.1525/bio.2011.61.12.6.

[11] R. A. Cheke and M. El Hady Sidatt, “A review of alternatives to fenthion for quelea bird control,” *Crop Prot.*, vol. 116, no. July 2018, pp. 15–23, 2019, doi: 10.1016/j.cropro.2018.10.005.

[12] C. Dias, D. Alberto, and M. P. A. F. Simões, “Cap. 01 - Produção de pêssego e nectarina na Beira Interior,” *+Pêssego-Guia Prático de Produção*, no. June, pp. 15–31, 2016.

[13] C. A. Lindell, “Commentary Supporting farmer adoption of sustainable bird management strategies,” *Human-Wildlife Interact.*, vol. 14, no. 3, pp. 442–450, 2020, doi: 10.26077/6dda-b98d.

[14] A. Anderson *et al.*, “Bird damage to select fruit crops: The cost of damage and the benefits of control in five states,” *Crop Prot.*, vol. 52, pp. 103–109, 2013, doi: 10.1016/j.cropro.2013.05.019.

[15] Z. Wang, A. S. Griffin, A. Lucas, and K. C. Wong, “Psychological warfare in vineyard: Using drones and bird psychology to control bird damage to wine grapes,” *Crop Prot.*, vol. 120, no. March, pp. 163–170, 2019, doi: 10.1016/j.cropro.2019.02.025.

[16] R. G. L. Narayanan and O. C. Ibe, “Joint Network for Disaster Relief and Search and Rescue Network Operations,” *Wirel. Public Saf. Networks 1 Overv. Challenges*, pp. 163–193, 2015, doi: 10.1016/B978-1-78548-022-5.50006-6.

[17] J. G. Boubin, N. T. R. Babu, C. Stewart, J. Chumley, and S. Zhang, “Managing edge resources for fully autonomous aerial systems,” *Proc. 4th ACM/IEEE Symp. Edge Comput. SEC 2019*, pp. 74–87, 2019, doi: 10.1145/3318216.3363306.

[18] M. Palmieri, C. Bernardeschi, and P. Masci, “Co-simulation of semi-autonomous systems: The line follower robot case study,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10729 LNCS, pp. 423–437, doi: 10.1007/978-3-319-74781-1_29.

[19] H. González-Jorge, J. Martínez-Sánchez, M. Bueno, and P. Arias, “Unmanned aerial systems for civil applications: A review,” *Drones*, vol. 1, no. 1, pp. 1–19, 2017, doi: 10.3390/drones1010002.

[20] “Mateksys F405-SE/WSE — Copter documentation.” Accessed: May 20, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-matekf405-se.html.

[21] “The Cube Orange Standard Set (ADS-B Carrier Board) : Drones, UAV, OnyxStar, MikroKopter, ArduCopter, RPAS : AltiGator, drones, radio controlled aircrafts: aerial survey, inspection, video & photography.” Accessed: May 18, 2021. [Online]. Available: https://drones.altigator.com/the-cube-orange-standard-set-adsb-carrier-board-p-42736.html?pg=store.

[22] “The Cube Orange With ADSB-In Overview — Copter documentation.” Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-thecubeorange-overview.html.

[23] “The Cube Overview — Copter documentation.” Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-thecube-overview.html.

[24] “NAVIO2 Overview — Copter documentation.” Accessed: May 19, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-navio2-overview.html.

[25] “OpenPilot Revolution and RevoMini — Copter documentation.” Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-openpilot-revo-mini.html.

[26] “Parrot Bebop Autopilot — Copter documentation.” Accessed: May 20, 2021. [Online]. Available: https://ardupilot.org/copter/docs/parrot-bebop-autopilot.html.

[27] “Pixhawk 3 Pro User Guide - Pixhawk 3 Pro.” Accessed: May 18, 2021. [Online]. Available: https://drotek.gitbook.io/pixhawk-3-pro/.

[28] “Pixhawk Series | PX4 User Guide.” Accessed: May 18, 2021. [Online]. Available: https://docs.px4.io/master/en/flight_controller/pixhawk_series.html.

[29] "Pixracer — Copter documentation." Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-pixracer-overview.html.

[30] "QioTek Zealot F427 — Copter documentation." Accessed: May 20, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-qiotek-zealot.html.

[31] "RadioLink MiniPix — Copter documentation." Accessed: May 20, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-radiolink-minipix.html.

[32] "Archived Topic: Erle-Brain Linux Autopilot — Copter documentation." Accessed: May 19, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-erle-brain-linux-autopilot.html.

[33] "Archived: Emlid Edge — Copter documentation." Accessed: May 19, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-emlid-edge.html.

[34] "Autopilot Hardware Options — Copter documentation." Accessed: May 20, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-autopilots.html.

[35] "BeagleBoard.org - blue." Accessed: May 18, 2021. [Online]. Available: https://beagleboard.org/blue.

[36] "BeagleBone Blue - Seeed Studio." Accessed: May 18, 2021. [Online]. Available: https://www.seeedstudio.com/BeagleBone-Blue-p-2809.html.

[37] "CUAV Nora Overview — Copter documentation." Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-cuav-nora-overview.html.

[38] "CUAV V5 Plus Overview — Copter documentation." Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-cuav-v5plus-overview.html.

[39] "Cube Orange Flight Controller | PX4 User Guide." Accessed: May 18, 2021. [Online]. Available: https://docs.px4.io/master/en/flight_controller/cubepilot_cube_orange.html.

[40] "Erle-Brain 2, the newest Linux autopilot from Erle Robotics - Blogs - diydrones." Accessed: May 19, 2021. [Online]. Available: https://diydrones.com/profiles/blogs/erle-brain-2-the-newest-linux-autopilot-from-erle-robotics-1.

[41] "F4BY FMU — Copter documentation." Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-f4by.html.

[42] "FlyWoo F745 AIO BL_32 — Copter documentation." Accessed: May 19, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-flywoo-f745.html.

[43] "Holybro Durandal — Copter documentation." Accessed: May 18, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-durandal-overview.html.

[44] "Intel Aero Overview — Copter documentation." Accessed: May 19, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-intel-aero-overview.html.

[45] "Open Source Autopilot for Drones - PX4 Autopilot." Accessed: May 20, 2021. [Online]. Available: https://px4.io/.

[46] “Support for Intel® Aero Compute Board.” .

[47] “Mission Planner Home — Mission Planner documentation.” Accessed: May 21, 2021. [Online]. Available: https://ardupilot.org/planner/.

[48] “APM Planner 2 Home — APM Planner 2 documentation.” Accessed: May 21, 2021. [Online]. Available: https://ardupilot.org/planner2/index.html#home.

[49] “Introduction · MAVLink Developer Guide.” https://mavlink.io/en/ (accessed Jul. 01, 2021).

[50] “ArduPilot Documentation — ArduPilot documentation.” https://ardupilot.org/ardupilot/index.html# (accessed Jul. 01, 2021).

[51] “Choosing a Ground Station — Copter documentation.” Accessed: May 21, 2021. [Online]. Available: https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html.

[52] “Litchi for DJI Mavic / Phantom / Inspire / Spark.” Accessed: May 21, 2021. [Online]. Available: https://flylitchi.com/.

[53] “Data Capture Platform for Drones & UAVs.” https://droneharmony.com/ (accessed Jul. 01, 2021).

[54] “Rainbow for DJI Drones (Mavic Mini compatible) – Apps no Google Play.” https://play.google.com/store/apps/details?id=com.rainbow.drone.pro&hl=pt_PT&gl=US (accessed Jul. 01, 2021).

[55] “Red Waypoint APP.” http://redwaypoint.com/index.html (accessed Jul. 01, 2021).

[56] G. Venter, “Review of Optimization Techniques,” *Encycl. Aerosp. Eng.*, pp. 1–10, 2010, doi: 10.1002/9780470686652.eae495.

[57] I. Fister, X. S. Yang, J. Brest, and D. Fister, “A brief review of nature-inspired algorithms for optimization,” *Elektroteh. Vestnik/Electrotechnical Rev.*, vol. 80, no. 3, pp. 116–122, 2013.

[58] M. Hajihassani, D. Jahed Armaghani, and R. Kalatehjari, “Applications of Particle Swarm Optimization in Geotechnical Engineering: A Comprehensive Review,” *Geotech. Geol. Eng.*, vol. 36, no. 2, pp. 705–722, 2018, doi: 10.1007/s10706-017-0356-z.

[59] S. Sengupta, S. Basak, and R. Peters, “Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives,” *Mach. Learn. Knowl. Extr.*, vol. 1, no. 1, pp. 157–191, 2018, doi: 10.3390/make1010010.

[60] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[61] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, “An improved grey wolf optimizer for solving engineering problems,” *Expert Syst. Appl.*, vol. 166, no. August 2020, p. 113917, 2021, doi: 10.1016/j.eswa.2020.113917.

[62] R. Benkercha, S. Moulahoum, and B. Taghezouit, “Extraction of the PV modules parameters with MPP estimation using the modi fi ed fl ower algorithm,” *Renew. Energy*, vol. 143, pp. 1698–1709, 2019, doi: 10.1016/j.renene.2019.05.107.

[63] X. Yang, M. Karamanoglu, and X. He, “Flower pollination algorithm : A novel approach for multiobjective optimization Flower pollination algorithm : A novel approach for multiobjective optimization,” *Eng. Optim.*, vol. 46, no. 9, pp. 1222–1237, 2014, doi: 10.1080/0305215X.2013.832237.

[64] O. Hasançebi, T. Teke, and O. Pekcan, “A bat-inspired algorithm for structural optimization,” *Comput. Struct.*, vol. 128, pp. 77–90, 2013, doi: 10.1016/j.compstruc.2013.07.006.

[65] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Comput. Commun.*, vol. 149, no. July 2019, pp. 270–299, 2020, doi: 10.1016/j.comcom.2019.10.014.

[66] Y. Zhao, Z. Zheng, and Y. Liu, “Survey on computational-intelligence-based UAV path planning,” *Knowledge-Based Syst.*, vol. 158, no. March, pp. 54–64, 2018, doi: 10.1016/j.knosys.2018.05.033.

[67] C.-M. Tseng, C.-K. Chau, K. Elbassioni, and M. Khonji, “Autonomous Recharging and Flight Mission Planning for Battery-operated Autonomous Drones,” no. March, 2017, [Online]. Available: http://arxiv.org/abs/1703.10049.

[68] X. Li, Y. Zhao, J. Zhang, and Y. Dong, “A hybrid PSO algorithm based flight path optimization for multiple agricultural UAVs,” *Proc. - 2016 IEEE 28th Int. Conf. Tools with Artif. Intell. ICTAI 2016*, pp. 691–697, 2017, doi: 10.1109/ICTAI.2016.0107.

[69] B. S and S. S. S, “A Survey of Bio inspired Optimization Algorithms,” *Int. J. Soft Comput. Eng.*, vol. 2, no. 2, pp. 2231–2307, 2012, doi: 10.1007/s11269-015-0943-9.

[70] M. Mavrovouniotis, C. Li, and S. Yang, “A survey of swarm intelligence for dynamic optimization: Algorithms and applications,” *Swarm Evol. Comput.*, vol. 33, no. December 2016, pp. 1–17, 2017, doi: 10.1016/j.swevo.2016.12.005.

[71] P. Pradeep, S. G. Park, and P. Wei, “Trajectory optimization of multirotor agricultural UAVs,” *IEEE Aerosp. Conf. Proc.*, vol. 2018-March, pp. 1–7, 2018, doi: 10.1109/AERO.2018.8396617.

[72] “Hex Cube Black Flight Controller | PX4 User Guide.” https://docs.px4.io/master/en/flight_controller/pixhawk-2.html (accessed Jul. 01, 2021).

[73] “Home.” https://nuttx.apache.org/ (accessed Jul. 01, 2021).

[74] “Here+ RTK GPS — Copter documentation.” https://ardupilot.org/copter/docs/common-here-plus-gps.html (accessed Jul. 02, 2021).

[75] A. Rodríguez-Panes, J. Claver, and A. M. Camacho, “The influence of manufacturing parameters on the mechanical behaviour of PLA and ABS pieces manufactured by FDM: A comparative analysis,” *Materials (Basel).*, vol. 11, no. 8, 2018, doi: 10.3390/ma11081333.

[76] “Welcome to OpenTX.” https://www.open-tx.org/ (accessed Jul. 02, 2021).

[77] R. E. James Kennedy, “Particle Swarm Optimization,” 1995.

[78] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," *Proc. IEEE Conf. Evol. Comput. ICEC*, vol. 1, pp. 81–86, 2001, doi: 10.1109/cec.2001.934374.

[79] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002, doi: 10.1109/4235.985692.

[80] Y. Shi and R. Eberhart, "A modified particle swarm optimizer algorithm," 1998, doi: 10.1109/ICEC.1998.699146.

[81] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, 2018, doi: 10.1007/s00500-016-2474-6.

[82] "Plataforma Google Maps | Google Developers." https://developers.google.com/maps (accessed Jul. 05, 2021).

[83] P. Dauni, M. D. Firdaus, R. Asfariani, M. I. N. Saputra, A. A. Hidayat, and W. B. Zulfikar, "Implementation of Haversine formula for school location tracking," *J. Phys. Conf. Ser.*, vol. 1402, no. 7, 2019, doi: 10.1088/1742-6596/1402/7/077028.

[84] N. E. Daidzic, "Long and short-range air navigation on spherical Earth," *Int. J. Aviat. Aeronaut. Aerosp.*, vol. 4, no. 1, 2017, doi: 10.15394/ijaaa.2017.1160.

[85] M. A. Hoque, X. Hong, and B. Dixon, "Analysis of mobility patterns for urban taxi cabs," *2012 Int. Conf. Comput. Netw. Commun. ICNC'12*, pp. 756–760, 2012, doi: 10.1109/ICCNC.2012.6167524.

[86] C. N. Alam, K. Manaf, A. R. Atmadja, and D. K. Aurum, "Implementation of haversine formula for counting event visitor in the radius based on Android application," *Proc. 2016 4th Int. Conf. Cyber IT Serv. Manag. CITSM 2016*, pp. 3–8, 2016, doi: 10.1109/CITSM.2016.7577575.

[87] N. Chopde and M. Nichat, "Landmark Based Shortest Path Detection by Using A* and Haversine Formula," *GH Raisoni Coll. Eng. ...*, vol. 1, no. 2, pp. 298–302, 2013, [Online]. Available: http://www.ijircce.com/upload/2013/april/17_V1204030_Landmark_H.pdf.

[88] D. Tian and Z. Shi, "MPSO: Modified particle swarm optimization and its applications," *Swarm Evol. Comput.*, vol. 41, no. January, pp. 49–68, 2018, doi: 10.1016/j.swevo.2018.01.011.

[89] T. Monawar, B. S. Mahmud, and A. Hira, "Anti-theft vehicle tracking and regaining system with automatic police notifying using Haversine formula," 2017, doi: 10.1109/ICAEE.2017.8255459.

[90] "File Formats · MAVLink Developer Guide." https://mavlink.io/en/file_formats/ (accessed Jul. 12, 2021).

[91] R. Mesquita and P. D. Gaspar, "A Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling - Simulation Results," *2020 Int. Conf. Decis. Aid Sci. Appl. DASA 2020*, pp. 1144–1148, 2020, doi: 10.1109/DASA51403.2020.9317271.

[92] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," *Proc. 2011 3rd World*

Congr. Nat. Biol. Inspired Comput. NaBIC 2011, pp. 633–640, 2011, doi: 10.1109/NaBIC.2011.6089659.

[93] "Agriculture Overview." https://www.worldbank.org/en/topic/agriculture/overview (accessed Jul. 23, 2021).

[94] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail, "Energy-efficient wireless sensor networks for precision agriculture: A review," *Sensors (Switzerland)*, vol. 17, no. 8, 2017, doi: 10.3390/s17081781.

[95] L. Abualigah, "Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications," *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2949–2972, 2021, doi: 10.1007/s00521-020-05107-y.

[96] N. K. Jain, U. Nangia, and J. Jain, "A Review of Particle Swarm Optimization," *J. Inst. Eng. Ser. B*, vol. 99, no. 4, pp. 407–411, 2018, doi: 10.1007/s40031-018-0323-y.

[97] M. S. Kiran, "Particle swarm optimization with a new update mechanism," *Appl. Soft Comput. J.*, vol. 60, pp. 670–678, 2017, doi: 10.1016/j.asoc.2017.07.050.