processes

MDPI

Article

A Novel Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling

Ricardo Mesquita [1,2] and Pedro D. Gaspar [1,2,*]

[1] Department of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilha, Portugal; ricardo.mesquita@ubi.pt
[2] C-MAST–Centre for Mechanical and Aerospace Science and Technologies, 6201-001 Covilha, Portugal
* Correspondence: dinis@ubi.pt

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 become inefficient in the long run, requiring high maintenance and reducing 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 novel path planning optimization algorithm of UAVs based on Particle Swarm Optimization (PSO) is presented in this paper. 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 field tests were also essential to understand the algorithm's behavior of the path planning algorithm in the UAV, showing that there is less efficiency with fewer points of interest, but this does not correlate with the flight time. In addition, 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. Thus, 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, being suited for this and other applications.

Keywords: bird damage to fruit crops; unmanned aerial vehicles; path planning; meta-heuristic; path planning optimization algorithm

Citation: Mesquita, R.; Gaspar, P.D. A Novel Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling. *Processes* **2022**, *10*, 62. https://doi.org/10.3390/pr10010062

Academic Editors: Chia-Nan Wang and Nguyen Van Thanh

Received: 12 December 2021
Accepted: 26 December 2021
Published: 28 December 2021

1. Introduction

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 one of the topmost priority agendas of policymakers [1], companies, and researchers.

New technologies, such as Unmanned Aerial Vehicles (UAVs), commonly known as drones, have been increasingly used in agricultural activities. UAVs are unmanned aircrafts already being applied in agriculture and used to overcome traditional systems' failures. Several parameters characterize its types, such as the structure, method to take-off and land, and the number of motors. The main structure configurations are Horizontal Take-Off and

Landing, multirotor, helicopter, and Vertical Take-Off and Landing (VTOL), and 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 [2]. Depending on its functions, these may include cargo compartments, actuators, 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.

Bird damage to fruit and other horticultural crops is a well-documented agricultural problem [3,4] 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 [5]. With the advancement of technology, new techniques emerged to reduce the environmental impact. Bird cannons usually use propane to randomly imitate a loud sound, such as 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 mechanical, visual, and auditory methods to scare birds away from crop fields [6]. Another popular method is speakers which 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 in order not to become predictable. Alternatives 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, which is not always possible requiring knowledge of bird movement [7]. The central region of Portugal's 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 [8]. In this area, flocks of birds, such as shown in Figure 1, 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 [9].

Figure 1. Flock of birds in Soalheira, Fundão, Portugal.

In the USA, farmers lose tens of millions of dollars each year through direct losses and often ineffective efforts to deter birds [10], so drones with repelling systems emerge as solution tools. Wang et al. [11] 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 focus of future works.

As already stated, farmers use different methods to deal with bird damage to fruit crops. 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 [12] 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, and its advantages fill the disadvantages of traditional systems. Hence, drones with repelling systems start to emerge as valid solutions against bird damage to fruit crops. Although these problems have been solved, new ones arise, such as short flight time. This problem can be worked through flight optimization using path 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 paper 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.

The paper is structured as follows: Section 2 includes the state of the art of autopilots and ground control stations (GCS), optimization algorithms and UAV path planning algorithms. Next, Section 3 describes the materials and methods applied and used during the research. It includes the hardware description, the UAV and GCS, and the base path planning algorithm. Section 4 describes in detail the novel path planning algorithm. It includes its global architecture, parameters setting, how is performed the minimization between Points of Interest, how is performed the maximization of random waypoints, and how created the pre-planned mission file. Section 5 includes the analysis and discussion of results of three case studies, two of these simulated and other performed experimentally, Section 6 discusses the overall results of the novel path planning algorithm and last section, Section 7, includes general and specific conclusions of the study, showing the advantages of this novel path planning algorithm and its scalability.

2. State of the Art

2.1. Autopilots and Ground Control Stations

Fully Autonomous Aerial Systems (FAAS) is 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. Similar to 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 [13]. Semi-autonomous systems can sense their environment and perform their tasks autonomously, but humans may also supervise them [14].

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

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 [15].

Nowadays, some companies produce readily available, reliable, cheaper, and simple-to-use boards. Each autopilot is different in weight, dimensions, processor, internal sensors, and interfaces and has its advantages and disadvantages, depending on the application. After researching the most commercially available open and closed hardware used, it is possible to infer the following factors [16–39]. One of the most commonly used controllers is the STM32 processor family, due to the global characteristics of the processors, information, and run most software such as PX4, an open-source flight control software for drones and other unmanned vehicles [40], known for its ecosystem and hardware compatibility. Another processor widely applied is the Raspberry Pi (RPi), with a shield to accommodate external sensors and additional input and output. This microcomputer typically runs some versions of Linux and is popular due to being easy to program. The Intel Aero is an example of closed hardware, Linux base autopilot developed by Intel but was discontinued in February of 2019 [41]. Regarding the internal sensors, most autopilots have one or more accelerometers, gyroscopes, magnetometers, and barometers, and for the interfaces all present Pulse-Width Modulation (PWM) output, RC input, and the most common communication protocol (Serial Peripheral Interface, Inter-Integrated Circuit, Universal Asynchronous Receiver-transmitter, among others). Depending on the model and maker, the autopilot may have other characteristics, but these are the most common for open and closed hardware.

As Ground Control Stations (GCS), there are several software options. These systems control the autonomous flights, the visualization of the flight map, and make video streaming in real-time, among others. Mission Planner [42] is a GCS widely used due to having the full feature and lots of information. It is open-source and 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 [43] 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, advantageous for new users. QGroundControl [44] works with MAVLink [45], a lightweight messaging protocol for communicating with drones, capable of autopilots, including ArduPilot [46]. It is unique among the GCS offerings as it runs on all platform desktops and mobiles [47]. 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 [48]. Other similar software are Drone Harmony [49], Rainbow [50], and Red Waypoint [51].

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.

2.2. Optimization Algorithms

Optimization techniques, or algorithms, are used to find the combination of design variable values that results in the best objective function value while satisfying all the equality, inequality, and side constraints [52]. 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 cope with these challenging optimization problems eventually. Nature has motivated many researchers in different ways and thus is a rich source of inspiration. Nature-Inspired optimization methods are applied in all areas, and their development and review are continuous.

Hajihassani et al. [53] 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 and itself [54].

The Grey Wolf Optimizer (GWO) [55] 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 of five to 12 elements with 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. [56] 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. [57] proposed a modified flower algorithm (FA) for extraction of the photovoltaic (PV) module parameters with maximum power point (MPP) estimation. The FA [58] 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 so 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. [59] proposed a bat-inspired algorithm for structural optimization. Bat-inspired (BI) algorithm is 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 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 hot research topic, 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 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 [60].

In 2018, Zhao et al. [61] 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: algorithms, time domain, and space domain. Based on the aspect of algorithms, the authors presented Figure 2. 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).

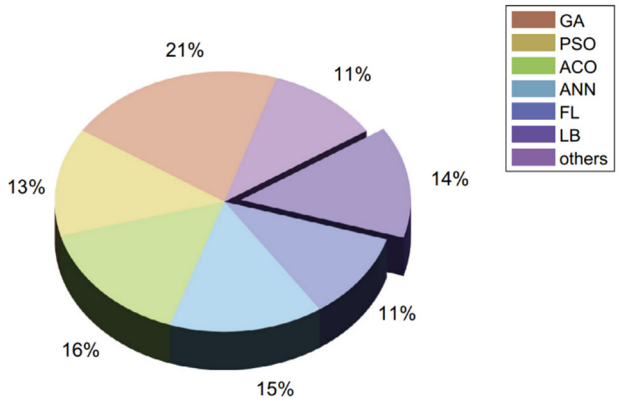


Figure 2. Classification of CI methods in UAV path planning by algorithm [61].

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, most studies focus on minimizing the length of a

path with obstacle or threat constraints in the environment for offline path planning. The optimization problem is multi-modal because there can be multiple 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 many 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. For the last classification, the algorithms are divided into 2D and 3D environments. Traditional path planning methods have been described by a 2D scene. Of the articles, 55.8% 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.

An example of path planning applied to a specific area is presented in Li et al. [62] applied to precision agriculture, using a hybrid PSO algorithm to optimize flight paths in a UAV group. Farmers need to spray different orchard blocks daily. 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. This article aims to obtain the optimal paths for the group of UAVs so that the flight time 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.

3. Materials and Methods

3.1. Materials

3.1.1. Unmanned Aerial Vehicle (UAV)

Multirotor drones being low cost, with good maneuverability and VTOL capability, are suitable for precision agriculture remote sensing tasks [63]. 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 [63]. 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 3 shows the multirotor used in this study during a test flight.

Figure 3. Multirotor used in this study.

The optimization algorithm presented in this study generalizes to any UAV configuration and flight time capability. 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, Cubepilot, Geelong, Australia [64], 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 [65]. The Hex Cube Black is reliable and has a premium built quality with much information. As a GPS receiver, the HERE+ 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 [66].

To activate the flight plan, the RadioKing TX18S, ChangZhou, China was used, a 2.4 GHz, 16 channel (CH) multi-protocol radio frequency (RF) system transistor with the open-source firmware OpenTX [67] 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 in this step the UAV is configured and calibrated, ensuring a precise and safe flight. The setup menu of Mission Planner has three main sections of settings: mandatory hardware, optional hardware and advanced. The mandatory hardware allows the user to calibrate sensors such as 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 such as 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 were not used as 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. 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 GCS software, and the values were loaded to serve as references.

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.

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. [68], 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}. In the original form proposed by Kennedy et al. [69], 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 [70].

Equations (1) and (2) show how the velocity and position of each particle is updated.

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

$$x_{ij}^{it+1} = x_{ij}^{it} + v_{ij}^{it+1} \quad (2)$$

Since the initial version of PSO was not very effective in the optimization problem, a modified PSO algorithm [71] 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 (3). 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 [72].

$$v_{ij}^{it+1} = w \cdot v_{ij}^{it} + c_1 \cdot r_1 \cdot \left(p_{best\ ij} - x_{ij}^{it} \right) + c_2 \cdot r_2 \cdot \left(g_{best\ ij} - x_{ij}^{it} \right) \quad (3)$$

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. As a stopping condition, the PSO used the maximum number of iterations.

3.2.2. Haversine Formula

Haversine Formula has its law: all equations are used based on the shape of spherical earth by eliminating the fact that it is slightly elliptical (ellipsoidal factor). This is a particular case of a general formula in spherical trigonometry related to the sides and angles of a spherical triangle. A certain degree of curvature affects the calculation of the distance from one point to another on the earth's surface [73]. 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 (4) and (5), representing the Haversine Formula. The distance between the two points is defined as d in Equation (5).

$$c = \sqrt{\sin^2\left(\frac{lat1 \ - \ lat2}{2}\right) + \cos(lat1) \cdot \cos(lat2) \cdot \left(\frac{lon1 \ - \ lon2}{2}\right)} \tag{4}$$

$$d = 2 \cdot r \cdot \arcsin(c) \tag{5}$$

3.2.3. Generate Random Waypoints within a Circle

To generate random waypoints within a circle, it is necessary two steps. Initial random points are created with coordinates (x, y) over a disk with radius R. For that, Equations (6) and (7) are used:

$$x = r \cdot \cos \theta \tag{6}$$

$$y = r \cdot \sin \ \theta \tag{7}$$

where $r \ \in [0, \ R]$, and is a random value calculated through Equation (8):

$$r = R \cdot \sqrt{random()} \tag{8}$$

Moreover, θ is another random value, where $\theta \ \in \ [0, \ 2\pi]$ and it is calculated by Equation (9):

$$\theta = 2 \cdot \pi \cdot random() \tag{9}$$

Then, 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. Equations (10)–(12) were used to convert the points to waypoints.

$$OneDegree = \frac{Earth \ Radius \cdot 2\pi}{360} \tag{10}$$

$$RandomLatitude = \frac{latCenter + x}{OneDegree} \tag{11}$$

$$RandomLongitude = \frac{lonCenter + y}{OneDegree * \cos\left(\frac{latCenter \cdot \pi}{180}\right)} \tag{12}$$

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

4. Novel Path Planning Algorithm

4.1. Global Architecture

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

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. These segments will be mentioned as points of interest (PoI) and have different damage proportions. 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. However, birds can detect patterns and learn how to avoid them, so different numbers of random waypoints are required 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. So, the proposed algorithm can be divided into four main steps, and they will be explained in the following sections.

4.2. Parameter Setting

Farmers will use this path planning optimization algorithm, so developing a GUI to import the specific parameters was essential. The graphical interface was built, around PySimpleGUI, to be visually simple and easy to use and can be divided into two main windows. On the first page, the user can insert the file name, the destination path. Additionally, it is further possible to enter the number of PoIs, the speed of the autonomous mission, and the drone's flight time in minutes. The speed matches the parameter WPNAV_SPEED on Mission Planner. If the user does not want to modify it, a check button was built to activate the default velocity. These last two 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, and the height at which the drone will fly. When everything is complete, it is possible to continue to the next page or exit. Figure 4 represents the first window of the path planning optimization algorithm GUI developed.

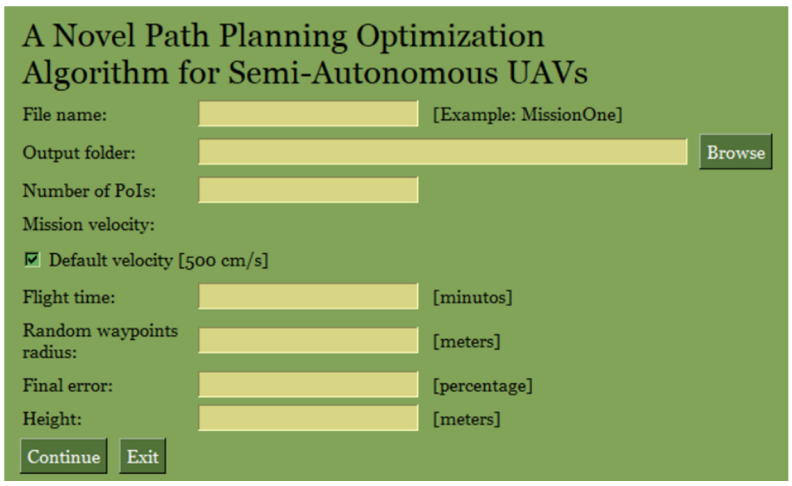


Figure 4. First window of the path planning optimization algorithm GUI.

The second window is used to insert the geographical coordinates of take-off, landing, and PoIs, with the percentage of bird damage incidence at each point. In some cases, the take-off and landing points are the same, so a check button has been included not to enter the same values twice. To insert a new PoI, the user writes each value and clicks the Ok button, and the values will appear below. The Clean button undoes 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, or a value can be entered. This rate is assigned from zero (there is no damage, and the PoI does not need points) to five (severe damage, the maximum number of points must be generated). This scale ensures differentiation in the number of random waypoints between PoIs. Still, if the farmer needs more diversity, it can introduce any maximum value because the incidence rate is calculated by dividing the value by the sum of the assigned values, obtaining a unit scale. 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 5 represents the second window of the path planning optimization algorithm GUI developed.

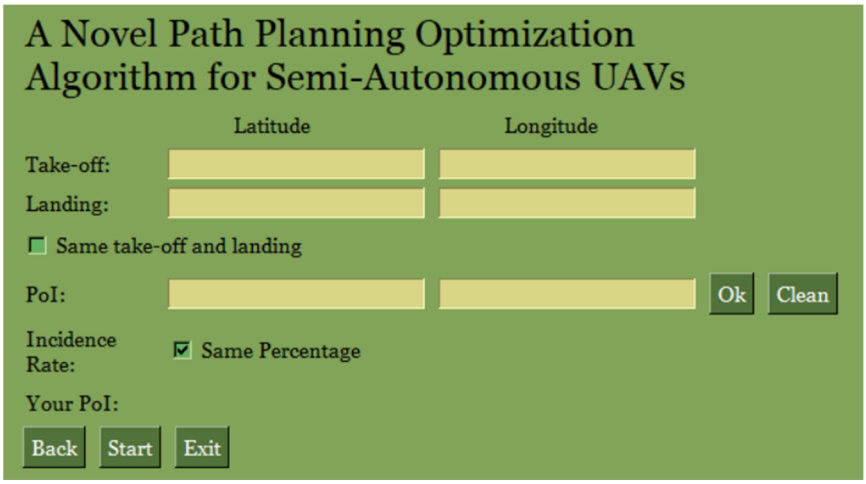
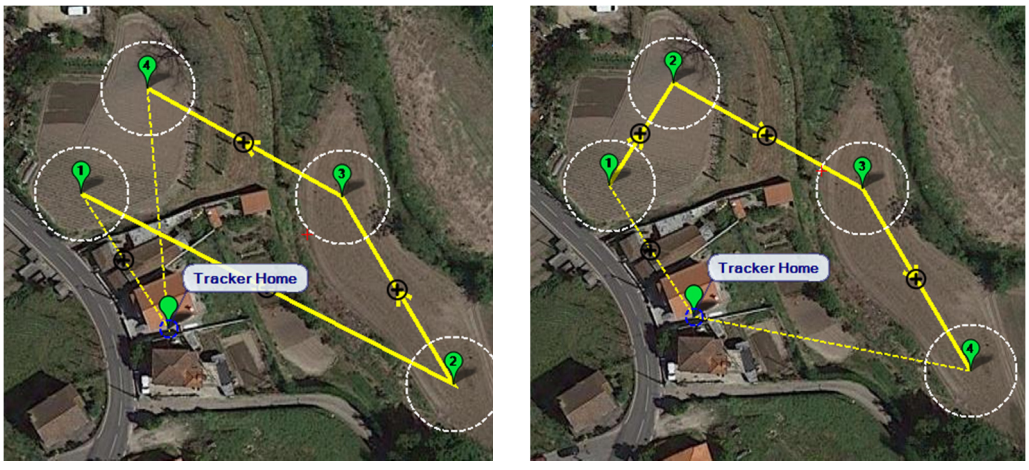


Figure 5. Second window of the path planning optimization algorithm GUI.

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 6 represents the importance of minimizing the path between PoIs with two cases through an example. Figure 6a show a random path between PoI and Figure 6b shows an optimized path between PoI. The path of the first case scenario has a total distance of 268.3 m, while the path of the second case scenario has 216.3 m, which represents a reduction of 52 m.



(**a**) (**b**)

Figure 6. Example of the importance of minimizing the path between PoIs: (**a**) Without the optimization function (**b**) With the 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. The maximum number of points is calculated with the

remaining distance. Then, the ratio between the total number and the incidence rate is processed to find the unique number for each PoI. Then, random waypoints around the PoI will be generated based on the method explained in the previous section, 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, a mission plain-text file format file will be created with all waypoints. Otherwise, the algorithm will compare the difference and adjust the total number of waypoints, generating new waypoints and rerunning 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 write it 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 line(s) are mission items in each column [74].

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. Case study #3 is developed to analyze the real-world application and the impact on battery autonomy and height influence. A peach orchard in Orjais, Covilhã, in Portugal, marked in the red line in Figure 7, was used as a search field in both simulation and real-world tests.

Figure 7. Search study field.

All cases used values real-life representations of each parameter, and the limits were used to compare. Each variation was tested in the three possible scenarios:

- Scenario 1, same incidence rate (the same number of waypoints assigned per PoI);
- Scenario 2, different rates (different number of waypoints per PoI);

- Scenario 3, no bird damage (in the vector of the previous scenario, were assigned zero waypoints to some PoIs).

For all cases in the three scenarios, random vectors of damage were attributed. Due to the heuristic nature of the PSO, for each scenario, all simulations ran thirty-five times, and the average of the: total distance for path planning obtained by the proposed 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. In all cases, the PyCharm Community Edition as an interpreter on Windows 10 of 64 bits was used to run the proposed algorithm on a computer with an Intel Core i7-6700HQ CPU and 16 GB RAM. In the field case, each of the scenarios and parameters were tested individually with four flights. The values obtained in this case were: the voltage difference in V, measured through the voltage sensor of the power brick, the autonomy from the batteries, calculated through the battery's charger in mAh, and the flight time in seconds obtained from the flight logs.

Table 1 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 1. 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 (13), was selected due to being the best for efficiency [75].

$$w = 0.5 + \frac{random()}{2} \tag{13}$$

5.2. Case Study #1

The goal of the first study case was to understand the algorithm's performance with multiple PoI and a 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 8 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

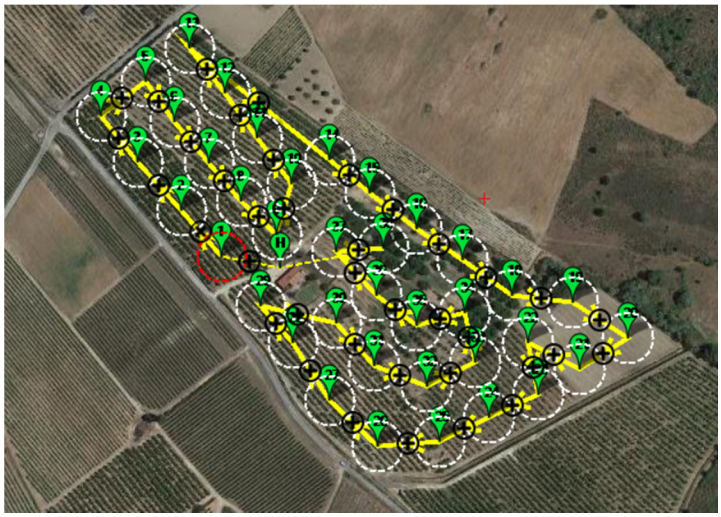


Figure 8. PoIs of the Case Study #1.

As mentioned above, in all simulations, the three possible cases were performed with the same incidence rate vectors represented in Table 2.

Table 2. Incidence rate vectors for Case Study #1.

Scenarios	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]
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, calculated through the mission speed and the flight time. Since Li-Po batteries last an average between 20 to 30 min, with the Mission Planner's default speed of 500 cm/s, the average values for the total distance for path planning were between 6000 and 9000 m. In this condition, 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 3 shows the results obtained in the variation of total distances for path planning. There is nothing to point out about the total distance for path planning obtained by the proposed algorithm. All scenarios for both distances presented identical results, with Scenario 2 having the best result reaching an average error in total distance for path planning obtained from both cases of less than 1% of the original value. Few points were obtained in the total number of waypoints for the 6000 m distance, which is expected as this value is related to the minimum distance between PoIs and the total distance for path planning. State 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 6000 m in which it was difficult for the algorithm to find a good distance value, having to rerun an average of 13 times and 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 9000 m, the time is much longer than the other two scenarios. Since there is a lower number PoIs, the algorithm must calculate more waypoints per point of interest.

Table 3. Average results from the variation of total distances for path planning.

Total Distance	Parameter Obtained	Scenario 1	Scenario 2	Scenario 3
9000 m	Total Distance Obtained by the Optimization Algorithm [m]	8890.980	8932.331	8849.140
6000 m		6185.257	5951.177	5936.690
9000 m	Total Number of Waypoints	236	233	253
6000 m		119	110	133
9000 m	Number of Iterations	4	7	7
6000 m		13	2	5
9000 m	Execution Time [min]	10.562	16.590	28.853
6000 m		8.681	1.823	5.226

Then the final error was tested, and the results are shown in Table 4. Initially, the values were set at 10%, and 1% of the total distance for path planning. After performing some tests 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. Due to not becoming a valid comparison, the value was defined as 3%. Since it is the final error to be varied, the total distance for path planning was constant at 7500 m, and the same take-off and landing position was used, positioned on the center of the field. In the case that the final error was set at 10%, the average values in the three scenarios have a maximum of less than 7% related to the total distance for path planning obtained by the proposed algorithm. When the final error is 3%, all scenarios average errors of less than 1% of the actual 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.

Table 4. Average results from the variation of final error.

Final Error	Parameter Obtained	Scenario 1	Scenario 2	Scenario 3
10%	Total Distance Obtained by the Optimization Algorithm [m]	7136.907	7099.347	7014.388
3%		7528.572	7556.262	7506.082
10%	Total Number of Waypoints	163	159	178
3%		184	179	199
10%	Number of Iterations	3	5	5
3%		72	45	6
10%	Execution Time [min]	3.924	6.067	9.718
3%		203.933	83.237	15.678

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.

The last parameter varied, ending Case Study #1, was the take-off and landing position. Table 5 shows the results obtained. Two cases were used to test this parameter: first with the same take-off and landing position placed in the center of the field and then with different take-off and landing located at opposite sites in the field. Since the take-off and landing positions varied, the total distance for path planning and final error was constant at 7500 m 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 5. Average results from the variation of the take-off and landing position.

Take-Off and Landing Position	Parameter Obtained	Scenario 1	Scenario 2	Scenario 3
Same	Total Distance Obtained by the Optimization Algorithm [m]	7508.493	7551.033	7473.269
Different		7219.699	7524.375	7399.987
Same	Total Number of Waypoints	178	179	198
Different		177	174	194
Same	Number of Iterations	6	7	6
Different		7	7	6
Same	Execution Time [min]	10.889	10.995	14.064
Different		12.168	11.673	14.098

5.3. Case Study #2

After understanding the performance of the optimization algorithm with the variation of each parameter in the first case study, a second case was developed 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. Ten PoIs were chosen to cover the entire field of study with random waypoints radius of fifty meters. Figure 9 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

Figure 9. PoIs of the Case Study #2.

In this case study, the performance of the optimization algorithm of the PoIs presented in Figure 9, named hereafter as the case with lower PoIs, and was compared with the PoIs in Figure 8, 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 6.

Table 6. Incidence rate vectors for Case Study #2.

Scenarios	Minimization
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 proposed algorithm, total number of waypoints, number of iterations, and execution time). The values corresponding to the parameters were: 7500 m for the total distance for path planning, a final error of 5%, and the same take-off and landing point. Table 7 shows the results obtained with the PoIs of both cases. The total distance for path planning obtained shows similar values on average, with lower PoIs presenting better results, 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 are more iterations because the algorithm takes longer to reduce the total number of waypoints until it has a viable value.

Table 7. Average results from Case Study #2.

Number of PoIs	Parameter Obtained	Scenario 1	Scenario 2	Scenario 3
Higher	Total Distance Obtained	7498.936	7599.392	7419.634
Lower	by the Optimization Algorithm [m]	7556.118	7511.199	7563.032
Higher	Total Number of Waypoints	180	179	198
Lower		126	126	129
Higher	Number of Iterations	6	7	6
Lower		21	16	25
Higher	Execution Time [min]	11.570	10.222	12.806
Lower		209.111	175.687	408.003

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 m, near the treetops, and the other at 20 m. The field tests 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 10 shows the PoIs (green dots) used with the corresponding radius from which the random waypoint will be generated (white circle).

Figure 10. 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 8.

Table 8. Incidence rate vectors for Case Study #3.

Scenarios	Minimization
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 heights, the values corresponding to the parameters were: 1500 m 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. Four batteries were then used of the same brand with the same configuration, and no battery was repeated per scenario. All flights were carried out over three days at different times sequentially to ensure data reliability. Table 9 shows the results obtained from the quadcopter studied. After analyzing the outcomes, it is possible to indicate that scenario 3 presents lower voltage difference and higher autonomy, although this does not directly transpose into more flight time but 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 m because the cornfield was covered with trees that protected the drone at lower altitudes. It is also essential to indicate that there is no direct connection between the batteries' autonomy and the flight time. One would expect less flying time with higher autonomy, but this does not always happen since drones depend on factors such as wind. It is also possible to establish that there is no relationship between the mission velocity and the flight time, considering that for 1500 m, an expected 5 min mission should be obtained, and the time varies between 7 to 14 min, approximately.

Table 9. Average results from Case Study #3.

Height	Parameter Obtained	Scenario 1	Scenario 2	Scenario 3
10 m	Voltage Difference [V]	1.17	1.23	1.09
20 m		1.22	1.25	1.08
10 m	Autonomy [m]	1662	1804	1569
20 m		1763	1837	1497
10 m	Flight Time [sec]	599	724	588
20 m		612	742	722

6. Discussion

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 tests were made in the three possible scenarios: same incidence rate, different rates, and no bird damage. 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 a lower number PoIs and a higher random waypoints radius compared to the previous case study. Case study #3 was the field test where the height was varied.

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 min when the error is 3%. When the final error is 5%, the average execution time is 12 s for all cases. 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 observe with Case study #1 that the takeoff and landing position do 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 min and 24 s. This result reveals that it will be necessary 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.

Case study #3 shows that fewer PoIs (Scenario 3) cause a higher autonomy with an average of 1533 mAh than 1712 mAh and 1820 mAh of the other two scenarios. A factor 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 min in mission time and some flights reached 14 min. One of the most critical conclusions derived from Case study #3 is that height does not influence flight, and terrain and weather conditions are the most influencing factors.

After performing all case studies, it is possible to conclude that the proposed algorithm present satisfactory results, especially in the final 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.

7. Conclusions

Birds eat and damage fruit in orchards, leaving it susceptible to infection and reducing its quality [5]. 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 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 versatility and low maintenance, making them potential solutions to this problem when combined with repelling systems. One of the most notable disadvantages is 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.

This article proposes a novel path planning optimization algorithm for semi-autonomous UAVs in bird repellent systems based on Particle Swarm Optimization, developed in Python programming language, to maximize the path and randomly generate 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 [76]. These techniques can optimize complete systems while maintaining the simplicity and efficiency of other algorithms. One of these techniques is Particle Swarm Optimization, a population-based self-adaptive, stochastic optimization technique [77], used for its performance, accuracy in solving optimization problems, easy implementation, and adaptation [78]. Recently, there are more effective variants of PSO algorithms such as the cooperative coevolutionary particle swarms (CCPSO) proposed by [79,80] where its superiority in terms of efficiency, performance and scalability in relation to the original PSO has been demonstrated. A novel approach is described in this paper. It starts by considering that a method to calculate the distance between two geographic coordinates has 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. A UAV was built and configured to fly through a pre-planned mission to test the proposed algorithm and, all flights were carried out in a controlled, safe environment, in line of sight.

The path planning optimization algorithm proposed in this paper can be divided into four main steps: Parameter Setting; Minimization Between PoIs; Maximization of Random Waypoints; Creation of Pre-Planned Mission File. 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 no bird damage. Case Study #1 was in simulation and used thirty-two PoIs with twenty meters random waypoints radius 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 ten PoIs, and fifty meters as random waypoints radius and was compared to the previous case study. For last, a field test with the quadcopter was undertaken. During Case Study #3, the height was varied in the three possible scenarios, and the voltage difference, autonomy, and time were evaluated. It should be mentioned that, although the focus of this study is on path optimization and not the effectiveness of the path planning optimization algorithm to repeal 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. In addition, although it is aimed at this agricultural problem, the algorithm

can be modified for other scenarios and problems and adapted to any autopilot or ground control station.

Author Contributions: Conceptualization, P.D.G.; Formal analysis, P.D.G. and R.M.; Investigation: R.M.; Methodology: R.M.; Software: R.M.; Project administration: P.D.G.; Supervision, P.D.G.; Writing—original draft, R.M.; Writing—review and editing, P.D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is within the activities of 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.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors are thankful for the opportunity and financial support to conduct this project from Fundação para a Ciência e Tecnologia (FCT) and R&D Unit "Center for Mechanical and Aerospace Science and Technologies" (C-MAST), under project UIDB/00151/2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anik, A.R.; Rahman, S.; Sarker, J.R. Five Decades of Productivity and Efficiency Changes in World Agriculture (1969–2013). *Agriculture* **2020**, *10*, 200. [CrossRef]
2. Daponte, P.; Vito, L.D.; Glielmo, L.; Iannelli, L.; Liuzza, D.; Picariello, F.; Silano, G. A review on the use of drones for precision agriculture. In Proceedings of the IOP Conference Series: Earth and Environmental Science, Ancora, Italy, 1–2 October 2018; 2019. [CrossRef]
3. Canavelli, S.B.; Branch, L.C.; Cavallero, P.; González, C.; Zaccagnini, M.E. Multi-level analysis of bird abundance and damage to crop fields. *Agric. Ecosyst. Environ.* **2014**, *197*, 128–136. [CrossRef]
4. Hannay, M.B.; Boulanger, J.R.; Curtis, P.D.; Eaton, R.A.; Hawes, B.C.; Leigh, D.K.; Rossetti, C.A.; Steensma, K.M.; Lindell, C.A. Bird species and abundances in fruit crops and implications for bird management. *Crop Prot.* **2019**, *120*, 43–49. [CrossRef]
5. Elsera, J.L.; Lindell, C.A.; Steensma, K.M.M.; Curtis, P.D.; Leigh, D.K.; Siemer, W.F.; Boulanger, J.R.; Shwiffa, S.A. Measuring bird damage to three fruit crops: A comparison of grower and field estimates. *Crop Prot.* **2019**, *123*, 1–4. [CrossRef]
6. Linz, G.M.; Homan, H.J.; Werner, S.J.; Hagy, H.M.; Bleier, W.J. Assessment of bird-management strategies to protect sunflowers. *Bioscience* **2011**, *61*, 960–970. [CrossRef]
7. Cheke, R.A.; Sidatt, M.E.H. A review of alternatives to fenthion for quelea bird control. *Crop Prot.* **2019**, *116*, 15–23. [CrossRef]
8. Dias, C.; Alberto, D.; Simões, M.P.A.F. Cap. 01–Produção de pêssego e nectarina na Beira Interior. In *+Pêssego-Guia Prático de Produção*; Centro Operativo e Tecnológico Hortofrutícola Nacional: Alcobaça, Portugal, 2016; Volume 1, pp. 13–31.
9. Lindell, C.A. Supporting Farmer Adoption of Sustainable Bird Management Strategies. *Hum. –Wildl. Interact.* **2020**, *14*, 442–450. [CrossRef]
10. Anderson, A.; Lindell, C.A.; Moxcey, K.M.; Siemer, W.F.; Linz, G.M.; Curtis, P.D.; Carroll, J.E.; Burrows, C.L.; Boulanger, J.R.; Steensma, K.M.M.; et al. Bird damage to select fruit crops: The cost of damage and the benefits of control in five states. *Crop Prot.* **2013**, *52*, 103–109. [CrossRef]
11. Wang, Z.; Griffin, A.S.; Lucas, A.; Wong, K.C. Psychological warfare in vineyard: Using drones and bird psychology to control bird damage to wine grapes. *Crop Prot.* **2019**, *120*, 163–170. [CrossRef]
12. Narayanan, R.G.L.; Ibe, O.C. 6–Joint Network for Disaster Relief and Search and Rescue Network Operations. In *Wireless Public Safety Networks 1*; Câmara, D., Nikaein, N., Eds.; ISTE Press–Elsevier: London, UK, 2015; pp. 163–193. [CrossRef]
13. Boubin, J.G.; Babu, N.T.R.; Stewart, C.; Chumley, J.; Zhang, S. Managing edge resources for fully autonomous aerial systems. In Proceedings of the SEC '19: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, Arlington, VA, USA, 7–8 November 2019. [CrossRef]
14. Palmieri, M.; Bernardeschi, C.; Masci, P. Co-simulation of semi-autonomous systems: The line follower robot case study. In Proceedings of the SEFM 2017: Software Engineering and Formal Methods, Trento, Italy, 4–8 September 2017. [CrossRef]
15. González-Jorge, H.; Martínez-Sánchez, J.; Bueno, M.; Arias, A.P. Unmanned Aerial Systems for Civil Applications: A Review. *Drones* **2017**, *1*, 2. [CrossRef]
16. Mateksys F405-SE/WSE-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-matekf405-se.html (accessed on 20 May 2021).
17. The Cube Orange with ADSB-In Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-thecubeorange-overview.html (accessed on 18 May 2021).

18. The Cube Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-thecube-overview.html (accessed on 18 May 2021).
19. NAVIO2 Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-navio2-overview.html (accessed on 19 May 2021).
20. OpenPilot Revolution and RevoMini-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-openpilot-revo-mini.html (accessed on 18 May 2021).
21. Parrot Bebop Autopilot-Copter Documentation. Available online: https://ardupilot.org/copter/docs/parrot-bebop-autopilot.html (accessed on 20 May 2021).
22. Pixhawk 3 Pro User Guide–Pixhawk 3 Pro. Available online: https://drotek.gitbook.io/pixhawk-3-pro/ (accessed on 18 May 2021).
23. Pixhawk Series | PX4 User Guide. Available online: https://docs.px4.io/master/en/flight_controller/pixhawk_series.html (accessed on 18 May 2021).
24. Pixracer-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-pixracer-overview.html (accessed on 18 May 2021).
25. QioTek Zealot F427-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-qiotek-zealot.html (accessed on 20 May 2021).
26. RadioLink MiniPix-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-radiolink-minipix.html (accessed on 20 May 2021).
27. Archived Topic: Erle-Brain Linux Autopilot-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-erle-brain-linux-autopilot.html (accessed on 19 May 2021).
28. Archived: Emlid Edge-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-emlid-edge.html (accessed on 19 May 2021).
29. Autopilot Hardware Options-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-autopilots.html (accessed on 20 May 2021).
30. BeagleBoard.org-Blue. Available online: https://beagleboard.org/blue (accessed on 18 May 2021).
31. BeagleBone Blue—Seeed Studio. Available online: https://www.seeedstudio.com/BeagleBone-Blue-p-2809.html (accessed on 18 May 2021).
32. CUAV Nora Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-cuav-nora-overview.html (accessed on 18 May 2021).
33. CUAV V5 Plus Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-cuav-v5plus-overview.html (accessed on 18 May 2021).
34. Cube Orange Flight Controller | PX4 User Guide. Available online: https://docs.px4.io/master/en/flight_controller/cubepilot_cube_orange.html (accessed on 18 May 2021).
35. Erle-Brain 2, the Newest Linux Autopilot from Erle Robotics-Blogs-Diydrones. Available online: https://diydrones.com/profiles/blogs/erle-brain-2-the-newest-linux-autopilot-from-erle-robotics-1 (accessed on 19 May 2021).
36. F4BY FMU-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-f4by.html (accessed on 18 May 2021).
37. FlyWoo F745 AIO BL_32-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-flywoo-f745.html (accessed on 19 May 2021).
38. Holybro Durandal-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-durandal-overview.html (accessed on 18 May 2021).
39. Intel Aero Overview-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-intel-aero-overview.html (accessed on 19 May 2021).
40. Open-Source Autopilot for Drones-PX4 Autopilot. Available online: https://px4.io/ (accessed on 20 May 2021).
41. Support for Intel® Aero Compute Board. Available online: https://www.intel.com/content/www/us/en/support/products/97178/drones/development-drones/intel-aero-products/intel-aero-compute-board.html (accessed on 20 May 2021).
42. Mission Planner Home-Mission Planner Documentation. Available online: https://ardupilot.org/planner/ (accessed on 21 May 2021).
43. APM Planner 2 Home-APM Planner 2 Documentation. Available online: https://ardupilot.org/planner2/index.html#home (accessed on 21 May 2021).
44. QGC-QGroundControl-Drone Control. Available online: https://http://qgroundcontrol.com/ (accessed on 2 November 2021).
45. Introduction · MAVLink Developer Guide. Available online: https://mavlink.io/en/ (accessed on 21 May 2021).
46. ArduPilot Documentation-ArduPilot Documentation. Available online: https://ardupilot.org/ardupilot/index.html# (accessed on 21 May 2021).
47. Choosing a Ground Station-Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html (accessed on 21 May 2021).
48. Litchi for DJI Mavic/Phantom/Inspire/Spark. Available online: https://flylitchi.com/ (accessed on 21 May 2021).
49. Data Capture Platform for Drones & UAVs. Available online: https://droneharmony.com/ (accessed on 1 July 2021).

50. Rainbow for DJI Drones (Mavic Mini Compatible)—Apps no Google Play. Available online: https://play.google.com/store/apps/details?id=com.rainbow.drone.pro&hl=pt_PT&gl=US (accessed on 1 July 2021).
51. Red Waypoint APP. Available online: http://redwaypoint.com/index.html (accessed on 1 July 2021).
52. Venter, G. Review of Optimization Techniques. *Encycl. Aerosp. Eng.* **2010**, 1–10. [CrossRef]
53. Hajihassani, M.; Jahed Armaghani, D.; Kalatehjari, R. Applications of Particle Swarm Optimization in Geotechnical Engineering: A Comprehensive Review. *Geotech. Geol. Eng.* **2017**, *36*, 705–722. [CrossRef]
54. Sengupta, S.; Basak, S.; Peters, R.A., II. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 157–191. [CrossRef]
55. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
56. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. [CrossRef]
57. Benkercha, R.; Moulahoum, S.; Taghezouit, B. Extraction of the PV modules parameters with MPP estimation using the modified flower algorithm. *Renew. Energy* **2019**, *143*, 1698–1709. [CrossRef]
58. Yang, X.; Karamanoglu, M.; He, X. Flower pollination algorithm: A novel approach for multiobjective optimization. *Eng. Optim.* **2014**, *46*, 1222–1237. [CrossRef]
59. Hasançebi, O.; Teke, T.; Pekcan, O. A bat-inspired algorithm for structural optimization. *Comput. Struct.* **2013**, *128*, 77–90. [CrossRef]
60. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
61. Zhao, Y.; Zheng, Z.; Liu, Y. Survey on computational-intelligence-based UAV path planning. *Knowl. -Based Syst.* **2018**, *158*, 54–64. [CrossRef]
62. Li, X.; Zhao, Y.; Zhang, J.; Dong, Y. A Hybrid PSO Algorithm Based Flight Path Optimization for Multiple Agricultural UAVs. In Proceedings of the IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 6–8 November 2016. [CrossRef]
63. Pradeep, P.; Park, S.G.; Wei, P. Trajectory optimization of multirotor agricultural UAVs. In Proceedings of the IEEE Conference on Aerospace, Big Sky, MT, USA, 3–10 March 2018. [CrossRef]
64. Hex Cube Black Flight Controller | PX4 User Guide. Available online: https://docs.px4.io/master/en/flight_controller/pixhawk-2.html (accessed on 1 July 2021).
65. Apache NuttX. Apache NuttX documentation. The Apache Software Foundation. Available online: https://nuttx.apache.org/ (accessed on 1 July 2021).
66. Rodríguez-Panes, A.; Claver, J.; Camacho, A.M. The Influence of Manufacturing Parameters on the Mechanical Behaviour of PLA and ABS Pieces Manufactured by FDM: A Comparative Analysis. *Materials* **2018**, *11*, 1333. [CrossRef]
67. Welcome to OpenTX. Available online: https://www.open-tx.org/ (accessed on 2 July 2021).
68. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. Proceeding of the ICNN'95 -International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995. [CrossRef]
69. Eberhart, R.; Shi, Y. Particle swarm optimization: Developments, applications and resources. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001. [CrossRef]
70. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
71. Shi, Y.; Eberhart, R. A Modified Particle Swarm Optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998. [CrossRef]
72. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
73. Dauni, P.; Firdaus, M.D.; Asfariani, R.; Saputra, M.I.N.; Hidayat, A.A.; Zulfikar, W.B. Implementation of Haversine formula for school location tracking. *J. Phys. Conf. Ser.* **2019**, *1402*, 077028. [CrossRef]
74. File Formats · MAVLink Developer Guide. Available online: https://mavlink.io/en/file_formats/ (accessed on 12 July 2021).
75. Bansal, J.C.; Singh, P.K.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia Weight Strategies in Particle Swarm Optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011. [CrossRef]
76. Abualigah, L. Group search optimizer: A nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. *Neural Comput. Appl.* **2021**, *33*, 2949–2972. [CrossRef]
77. Jain, N.K.; Nangia, U.; Jain, J. A Review of Particle Swarm Optimization. *J. Inst. Eng. Ser. B* **2018**, *99*, 407–411. [CrossRef]
78. Kiran, M.S. Particle swarm optimization with a new update mechanism. *Appl. Soft Comput.* **2017**, *60*, 670–678. [CrossRef]
79. Hsieh, F.S. A comparative study of several metaheuristic algorithms to optimize monetary incentive in ridesharing systems. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 590. [CrossRef]
80. Hsieh, F.-S. A comparison of three ridesharing cost savings allocation schemes based on the number of acceptable shared rides. *Energies* **2021**, *14*, 6931. [CrossRef]