Pulmonary Hypertension Detection from Heart Sound Analysis

Optimal Path Planning for Autonomous Spraying UAS framework in Precision Agriculture

Lorenzo Becce[1], Nicoletta Bloise[1], Giorgio Guglieri[1]

***Abstract*— This paper presents a novel guidance and control strategy for multirotor Unmanned Aerial Systems (UASs) which aims to provide an autonomous and safe mission in precision agriculture applications.**

In the last few years, the research in this field has always improved thanks to the advent of new technologies and with the launch of the first smart farms. Precision aerial spraying of Plant Protection Products (PPP) in vineyards is the focus of this work, highlighting several advantages in terms of quality management, time and cost. In particular, we propose a combination of a Traveling Salesman Problem (TSP) solver with the well-know Theta* algorithm to investigate optimal UAS trajectories in order to visit a specific number of plants that require intervention. The final goal is to demonstrate the fulfillment of the evaluated trajectory with the on-board control system of the vehicle in provision for UAS field testing.

Finally, the planning strategy is applied to two case studies so as to present the feasibility of a more efficient autonomous UAS path planning.

I. Introduction

During the past decades, the progress of new and powerful technologies in the field of Internet of Things (IoT), Big Data, Artificial Intelligence (AI) and Robotics has sparked the so-called *fourth agricultural revolution*, which relies on essential innovation to ease the strive for food security in the spite of demographic expansion, scarcity of natural resources and climate transformations by increasing productivity while reducing its environmental impact.

In this context, various research areas propose studies on future sustainable agriculture with new instruments and operations in order to optimize crop monitoring and management in the new Smart Farms, as in [1]. In particular, aerial robots, commonly named Unmanned Aerial Systems (UASs), are programmable vehicles able to perform several functions in this sector, such as mapping crops or precisely administering Plant Protection Products (PPP), as discussed in [2], [3]. This specific application is growing significantly for its numerous advantages in terms of precision and operational flexibility. In addition, aerial vehicles collaborating with Unmanned Ground Vechicles (UGVs) provide an excellent alternative to traditional farming methods, as discussed in depth in [4]. In this framework, authors propose a cooperation among UAVs, dedicated to crop survey, and a fleet of UGVs and UAVs for operations such as spraying for pest control.

Nowadays, the most common UAS applications in agriculture are crop mapping and monitoring, with a progressive shift towards new challenging applications such as crop spraying. Anyway, restrictions for the use of drones as carriers of spraying systems for agriculture still apply in many countries while certification and regulatory organizations are working to remove these limitations. The use of these vehicles in agricultural areas is less restrictive than in urban scenarios due to the minor threats to human safety and privacy protection, as outlined in [5], but a dedicated regulation on aerial spraying applications has not yet been coded. Basically, the European Aviation Safety Agency (EASA) identifies three categories of UAS operations (Open, Specific and Certified) with different safety requirements depending on weight and mission. Aerial spraying operations fall under the *Specific* category: they involve specialized flight operations for which an authorisation is mandatory, as discussed in [6]. This category places a limit on vehicle size and Maximum Take-Off Weight (MTOW) to 25 kg, therefore limiting the payload to be carried. On the other hand, Beyond Visual Line-of-Sight (BVLOS) operation is allowed. The advantages of BVLOS flight are described in [7], where operational modes, communication technologies and automation levels of UAS operations in urban areas are reviewed. In particular, the study underlines the capability of semi-autonomous (where human supervision is involved in the decision making process) and autonomous flight operations of minimizing human factors, which is the major cause of UAS accidents [8]. As for the PPP spray application, the minimization of drift problems, the improvement of spray precision and the differentiated crop treatment will be most essential in ensuring flight permission even in countries like Italy where, as indicated in [9], aerial spraying of pesticides is prohibited, except in extreme cases where its advantages in terms of human and environmental impact are evident.

In particular, [10] carries out the preliminary design of an agricultural UAS, equipped with a spray system and whose MTOW is smaller than 25 kg, through a trade off analysis between drone market and mission requirements. Moreover, the spray system is designed by aerodynamics analysis, with a focus on nozzles position and geometries, and on the definition of spray operative modes. The precision of spraying is strongly influenced by parameters such as flight speed and altitude and nozzle configuration, to take advantage of rotors downwash in order to minimize the drift problem.

As for the present work, the reference scenario is the vineyard where the benefits of employing aerial vehicles are evidenced by the hilly terrain and irregular rows. In this situation, a careful initial mapping, producing a high-resolution

[1]L. Becce, N. Bloise and G. Guglieri are with the Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. Corresponding author: Nicoletta Bloise (e-mail: nicoletta.bloise@polito.it)

map of the work area, is of paramount importance to the precise fulfilment of the task. [11] presents an interesting algorithm for vineyard detection based on 3D point-cloud maps processed from multispectral imagery obtained by UAS surveys.

Once achieved an accurate vineyard map, the timely detection of grapevine symptoms is essential to plan dedicated flights with spraying drones to prevent larger outbreaks of plant diseases with potentially disastrous impacts on yield. Convolutional Neural Network (CNN) architecture, a class of deep neural networks introduced in [12], has great success in various applications including pattern recognition, and hence can be useful for our project in defining target waypoints that must be visited during the mission The main idea is to automate the detection of vineyard problems exploiting leaf color information, as discussed in [13]. However, the complex aspect of a Deep Learning Network lies outside of the scope of the present work. On the other hand, the limited endurance of the contemporary multi-rotor UASs calls for an optimised schedule of the trajectory touring through the plants in need of intervention that have been located, keeping terrain morphology, obstacles (buildings and plants) and vehicle constraints into account while minimising the distance flown. We propose a combination of a Traveling Salesman Problem (TSP) solver and Theta* algorithm. The TSP is widely studied for its many applications in modern technology, but nonetheless is currently an open problem. In the typical notation, the points to visit take the name of *cities*. [14] provides a comprehensive survey of the main approaches to TSP solution, some of which have been put to use in this work, namely the Nearest Neighbor, also detailed in [15] and used to provide a first, rough estimate, and the more sophisticated Genetic Algorithm which refines the solution to a highly satisfactory level.

The guidance system is completed with the introduction of an any-angle path planning algorithm, Theta*, well described in [16], [17]. Deriving from the better-known A*, it overcomes some of its flaws, chiefly the suboptimality of the solution, while maintaining its intuitiveness and speed by introducing a simple line of sight (LOS) verification to cut through the corners that characterise the typical jagged A* solution. An alternative philosophy was investigated envisioning Dubins curves [18] to generate the path after the TSP solution has been obtained. Some papers [19], [20] even propose to solve the TSP directly while keeping the Dubins constraints into consideration, guaranteeing that all manoeuvers between waypoints are feasible by the selected vehicle. While been a very interesting approach and surely worth further consideration, many disadvantages have been outlined. First, the need for obstacle-avoidance required a LOS-checking routine anyway, leaving two options: either including such a routine in the Dubins solver, quite complicated, or performing obstacle-free course planning after the Dubins solver and then running the latter again to ensure feasibility of the new plan, with a disastrous impact on computational requirements and running times for the overall planning system. Then, the three-dimensional adaptation of the Dubins is more complicated than that of A*-like planners. For these reasons, the Dubins approach has been rejected quite early in the decision process.

Since operations require excellent precision, the control system must guarantee the best performance and stability. In general, the autopilot system is based on various layers of Proportional, Integral and Derivative (PID) controllers to regulate position, attitude and angular velocities [21].

In summary, the main goal of this research is to determine an optimal and innovative UAS path planning to guarantee precision and autonomous spray applications in vineyards, minimizing flight application time. First of all an introduction of the overall project is presented, summarizing the agricultural UAS main characteristics, the spray operational modes and the automation levels. In particular, we proposed a solution to introduce autonomous spray application in vast areas by reducing time and cost, improving precision. Our original contributions compared to previous papers are the combination of TSP with Theta* algorithm to generate a sequence of waypoints to feed the UAS autopilot in a global framework.

The remainder of this paper is organized as follows. Section 2 explains the definition of the operational scenario, the agricultural UAS dynamics and design, and the instrument to realize autonomous missions. In Section 3, the most important part of the research is described with regards to routing and robust guidance algorithms. Then, in Section 4, the control system and relative dynamics constraints are described. The simulation results are discussed in Section 5. Finally, our conclusions are exposed in Section 6.

II. Problem statement

The proposed coupling of a TSP solver with a robust path planner guarantees high efficiency in spraying missions with multirotor UASs, where high accuracy is requested in visiting diseased plants. Vineyards are a reference scenario for this project due to their typical sloped and irregular terrain and the challenges of meticulously mapping of vine rows.

First of all, activities can be divided into three phases, as shown in Figure 1. During the *vineyard mapping* phase, small UASs are tasked with acquiring images by means of multispectral sensors in order to obtain high resolution 3D maps. The second phase, with the help of an expert agronomist, concerns *crop monitoring* for the purpose of detecting plant problems related to water shortages or the presence of pathogens and consequently creating prescription maps. In this way, it will be possible to identify plants that require prompt intervention, which are classified as waypoints on which to plan trajectories, taking into account desired flight speed and external disturbances. Once off-line path planning is performed, the *crop spraying* phase can start. The objective of this larger research effort is to achieve an intelligent and autonomous plant treatment system powered by UASs. One of the most critical elements, due to the lack of a dedicated human pilot, is the communication system. Thanks to the newest mobile Internet technologies, with 5G as the latest mobile network, Cloud technologies

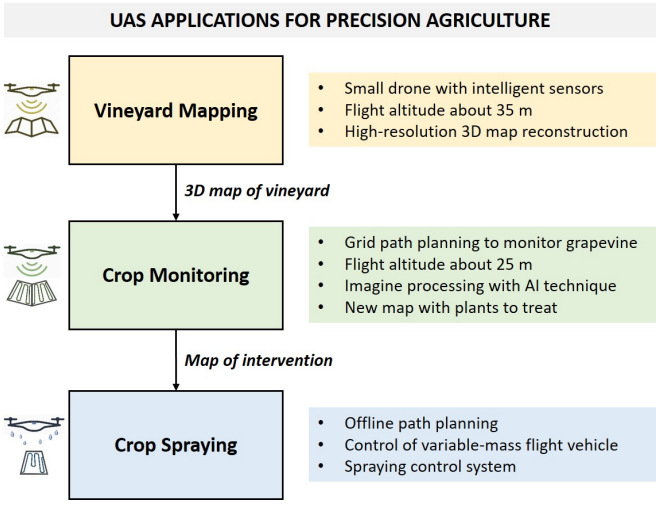

Figure 1. Spray applications phases with UAS

offer extraordinary opportunities, discussed in [22], to solve on-board resource limitation: the connection of devices to high-capacity wireless networks enables the management of massive amounts of data off-board, conferring a high computational power to a small and agile UAS. This allows AI methods to assist in the identification of critical issues on the field and to prevent serious diseases timely and autonomously.

The development of autonomous missions has a number of benefits, such as the reduced risk of accidents, human factors, operative flight time and cost. Any UAS mission can be classified based on the level of automation of the system, as in [23], for the aircraft system. In [7], authors resume this topic and evaluate the automation control of UAS in terms of human involvement. Moreover, they also explain how a high level of autonomy, combined with the BVLOS flight authorization, represents a huge opportunity for this type of drone operations in terms of distance, safety and flexibility in poorly accessible area.

A. UAS design

The activities listed above enforce several UAS design and architecture requirements. In particular, for the first two phases, we simply need a small drone with a full HD camera. The same is used to simulate and to evaluate path planning algorithm at the beginning. The reference model is the MAVTech Q4T Drone, a light UAS with a MTOW equal to 5 kg, reported in Figure 2. It is a commercial quadrotor, with good maneuverability and a programmable guidance, control and navigation system. For more details, refer to [24].

Whereas for spraying applications, a bigger drone equipped with a dedicated spray system is required. To comply with current regulations, the maximum weight must be less than 25 kg including payload. A preliminary design of a concept rotary-wing UAS for aerial precision spraying application is proposed in [10]. Moreover, the authors propose two interesting application modes to minimize drift problems exploiting the downwash of rotors. The first one

Figure 2. MAVTech - Q4T model

involves flying in a *cross configuration* over the vineyard, while in the second one, the UAS flies in a *X configuration* over the centre line section of two neighbouring rows of the vineyard.

B. Quadrotor dynamics model

A simplified six-Degrees of Freedom model of a quadrotor UAS has been put in place in order to test our guidance and control algorithms. An invaluable source on system modeling has been [25].

A simplified relation between the i^{th} rotor speed and its actions on the quadrotor frame is employed as

$$T_i = K_T\ \omega_i^2$$
$$\tau_i = (-1)^{i+1} K_q\ \omega_i^2$$

These relations exclude more complex effects such as blade flapping and aerodynamics, but are sufficient to guarantee a satisfactory model.

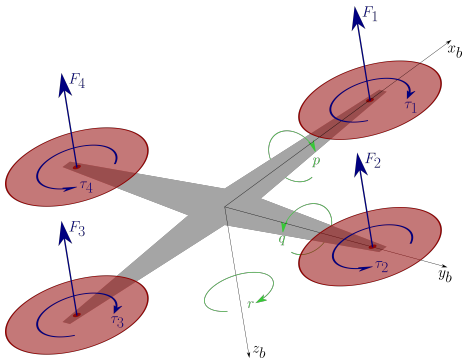

Figure 3. Rotor forces and moments

The general control strategy provides four control inputs depending on the rotor speeds, defined from the combination of rotor forces and moments illustrated in figure 3 as follows:

- **total thrust**: u_1 is the sum of all rotor forces. Referring to (II-B),

$$u_1 = T_b = \sum_{i=1}^{4} K_T \omega_i^2 \tag{1}$$

- **rolling moment**: u_2 is the torque generated by the even-numbered rotors, that is the ones placed along y_b. They

generate a moment along x_b as follows

$$u_2 = \tau_\varphi = l(-F_2 + F_4) \tag{2}$$

- **pitching moment**: u_3 is described by a similar equation to the previous one

$$u_3 = \tau_\theta = l(F_1 - F_3) \tag{3}$$

Due to the symmetry of the vehicle, the pitch dynamics is the same as the roll dynamics.

- **yawing moment**: u_4 is generated by taking advantage of the directions of rotation of the rotors, by using (II-B)

$$u_4 = \tau_\psi = \sum_{i=1}^{4} (-1)^{i+1} K_q \omega_i^2 \tag{4}$$

Knowing the maximum and minimum rotation speeds for the rotors, the limit control actions can be easily evaluated by simply substituting the maximum or minimum rotational speeds in (1), (2), (3) and (4).

In order to simplify the equations, some assumptions have been made:

- the flexibilty of the small structure is not taken into consideration
- the inertia matrix J_b is symmetrical, that is $J_b = diag[J_{xx}, J_{yy}, J_{zz}]$
- the Earth is considered flat and non rotating, as is often done when such phenomena contribute largely insignificant accelerations on small and relatively slow aircraft
- the ground effect is negligible, as is the wind
- the rotors are not flexible, so no flapping effect takes place
- the motors are not modeled, and therefore do not introduce delays in the dynamics

The translational dynamics is based on the classical North-East-Down (NED) inertial reference frame, noted by a "G" superscript for "ground" in the following. Therefore, $\mathbf{X}^G = \{N, E, D\}^\intercal$ will indicate the coordinates.

The total force in the ground frame is given by the sum of drag, gravity, and total rotor thrust (u_1). The complex shape of the drone makes the estimations of aerodynamic effects an excessively hard task, so drag is simplified as a force proportional to the velocity components in each direction:

$$\mathbf{F_d} = \begin{bmatrix} k_{dx} & 0 & 0 \\ 0 & k_{dy} & 0 \\ 0 & 0 & k_{dz} \end{bmatrix} \begin{Bmatrix} \dot{X^G} \\ \dot{Y^G} \\ \dot{Z^G} \end{Bmatrix}$$

u_1 is rotated from the body frame, to which it is fixed, by means of the fed back states from the rotational dynamics.

$$\mathbf{F_T}^G = R_b^G(\varphi, \theta, \psi) \begin{Bmatrix} 0 \\ 0 \\ -u_1 \end{Bmatrix}$$

In a similar manner, the equations for the rotational dynamics can be found as a sum of three contributions:

$$J_b \boldsymbol{\omega} = \boldsymbol{\tau_m} - \boldsymbol{\tau_g} - \boldsymbol{\omega} \times J_b \boldsymbol{\omega} \tag{5}$$

The first term are motor torques due to the controls:

$$\boldsymbol{\tau_m} = \begin{Bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{Bmatrix} = \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

For the gyroscopic torques due to propellers, let J_r be the rotational inertia of the single rotor, G_z the global z-axis versor (i.e. $G_z = \{0, 0, 1\}^\intercal$) and ω_i the angular velocity of the i^{th} rotor. Then

$$\boldsymbol{\tau_g} = \boldsymbol{\omega} \times G_z \sum_{i=1}^{4} J_r (-1)^{i+1} \omega_i$$

represents the gyroscopic torques generated by the propellers rotation when coupled with the rotation of the frame. Keep in mind that the boldface $\boldsymbol{\omega}$ represents the angular velocities with respect to the frame $p,\ q,\ r$ as the ones picked up by the gyroscopes on the frame, and that the signs in the sum have to be alternated to account for the different spin directions of the propellers as described above. The development of the cross product yields

$$\boldsymbol{\tau_g} = \begin{Bmatrix} \dot{\theta} \\ -\dot{\varphi} \\ 0 \end{Bmatrix} J_r \sum_{i=1}^{4} (-1)^{i+1} \omega_i$$

Finally, gyroscopic torques due to rigid body rotation are described by classical, rigid body dynamics:

$$\boldsymbol{\omega} \times J_b \boldsymbol{\omega} = \begin{Bmatrix} \dot{\theta}\dot{\psi}(J_z - J_y) \\ \dot{\psi}\dot{\varphi}(J_x - J_z) \\ \dot{\theta}\dot{\varphi}(J_y - J_x) \end{Bmatrix}$$

III. Path Planning

Given a number of waypoints representing the positions of each plant to treat, we are interested in the shortest possible tour visiting each of them. This is a common iteration of the famous Travelling Salesman Problem (TSP). The solution approaches of our interest are two: the Nearest Neighbor (NN) and the Genetic Algorithms family (GA); the former skips from city to city by simply choosing the closest. While being very intuitive, when the number of cities grows larger than a handful, this approach does not lead to optimal or even acceptable results, but due to its low computational effort can be used to work out a starting approximation to be further refined. The inner workings of the latter family are detailed in section III-A.

After the optimal visiting sequence has been obtained, a more precise path planning effort is necessary due to the variable nature of vineyards: slopes and other unforeseen obstacles may render the flight domain very hazardous, to the point where simple straight-line planning is not sufficient. The availability of a map of the yard, provided by the preliminary survey, proves very helpful in regard to this problem: after being converted to a Digital Elevation Model (DEM), it can be used for offline path planning around the orographic features of the terrain and other large obstacles such as plant rows and buildings. We decided to rasterize this model and feed it to a suitable path planning algorithm as

discussed in section III-B. The work started on a basic 3D A* algorithm, extended to enable any-angle path planning as suggested by [16]. Further collision avoidance has to be enforced through the use of on-board sensors, but lies outside of the scope of the present work.

A. Traveling Salesman Problem

Due to the complex nature of the problem and the large abundance of solvers openly available online, we chose to pick one from the MATLAB Central portal. A comprehensive toolbox built by Joseph Kirk [26], also available on GitHub, was retrieved. The toolbox contains numerous solvers tailored for several iterations of the problem, including multiple salesmen and constrained capacity vehicle routing problems, hence proving invaluable in future extensions of the application, such as the case where a single yard is too large to be covered in a single pass (i.e. checking whether the sprayer tank runs dry before completion) or where more sprayer UASs are to be employed simultaneously on a single yard. These, however, are possible future developments still to be investigated, but they reveal the flexibility provided by this package.

In order to improve on the computational times, a sequence of a Nearest Neighbor solver and a Genetic Algorithm has been put together as suggested by the package author. The NN provides a first approximation to be fed into the GA, which can significantly improve on it by generating a number of random permutations of the starting path, that is called the *initial population*, which is then *evolved* through several iterations until some condition is satisfied, such as the lack of further improvement through consecutive iterations or a fixed number of steps. The evolution process can be detailed as follows: from an initial population of randomly generated paths, based on an appropriate metric, a small number of the best ones (four in this case) is picked. The rest is discarded (go *extinct*, to maintain the biological terminology from which GAs take inspiration), while the best are mutated enough times so as to rebuild a new population of the original size. The cities are contained in the rows of a matrix, so a path is represented by a vector containing the indices of said rows in a determined order. The possible permutations that can operate on each path are:

- **flip**: a portion of the path, chosen by a random pair of indexes (start and end of the segment) is reversed.
- **swap**: the positions of two randomly chosen cities in the path are swapped.
- **slide**: the last city of a random segment is pulled out and reinserted at the beginning of the segment, which has now slid forward one position.

A brief comparison between two methods has been put in place: the two-stage computation running NN+GA and the GA alone, both on sets of 20, 100 and 500 cities randomly generated in a cubic, 100-metres sided 3D space. The running times have been recorded using the well-known `tic/toc` MATLAB commands and are compared in Table I together with the best distances found. The direct run is respectively 1.4%, 6.7% and 158.8% worse than the combined run in terms of total length, even though having taken 92.9% and 13.8% longer and 62.3% shorter. In conclusion, the combined approach has been selected due to its more desirable results with respect to the direct run, even at the expense of increasingly longer computational times, especially since we do not expect the waypoints to be as dense or numerous as the 500-city case.

		NN+GA, overall	GA, direct
20 cities	time [s]	4.296	8.287
	length [m]	632.05	640.86
100 cities	time [s]	15.568	17.726
	length [m]	1775.6	1883.3
500 cities	time [s]	161.1	60.7
	length [m]	5157.0	8191.2

TABLE I

COMPARISON BETWEEN COMBINED APPROACH AND DIRECT RUN

B. Theta* algorithm

The Theta* algorithm is an extension of the famous A* search algorithm, on which a comprehensive explanation is available in [27] and whose limit lies in the strict adherence to the grid, which enables any-angle path planning. The A* search algorithm was introduced in [28]. The domain is discretised through a grid made of nodes, or vertices, for each of which three values are retained:

- **G-value**: $g(s)$ is the cost-to-come of the vertex s, that is the smallest cost found to go from the starting node s_I to s.
- **H-value**: $h(s)$ is an estimate of the cost-to-go, that is an underestimation of the cost to go from s to the goal node s_G. Usually the best estimate, easy to calculate and sufficient to ensure a satisfactory performance, is the distance between s and goal

$$h(s) = \|s_G - s\|$$

- **F-value**: $f(s)$ is of practical usefulness, namely the sum of G and H, which indicates the estimated cost of a shortest start-end path passing through s:

$$f(s) = g(s) + h(s)$$

  Often, and in this case, an α value is inserted which multiplies the heuristic H-value, called a "heuristic weight":

$$f(s) = g(s) + \alpha h(s)$$

  This weight influences the relevancy of the heuristic with respect to the cost-to-go. Another source [17] uses a slightly different method with two gains, one for $g(s)$ and one for $h(s)$, to gauge the impact of the two estimates.
- **parent identification**: $p(s)$ is needed to trace back the path from the goal to the start once A* has successfully reached the end of the search.

The above values are calculated for each node encountered by the algorithm and stored in one of two data structures:

- **open list**: it contains the nodes that are still available for expansion.
- **closed list**: it gathers all the nodes which have been expanded and for which no further examination is possible.

The idea is to move from node to node by choosing, among the neighbors of the current one, the one with the lowest F-value. A simple kinematic constraint can be inserted here: we can instruct the system to avoid moving from a cell to the one directly above or below it, thereby restricting vertical climb.

Introduced by [16] in 2010, Theta* foresees a Line of Sight (LOS) checking routine between each new move s' departing from s and its *grandparent* (or the parent of s, so $p(s)$). If the checking routine has positive outcome, then the path is updated so as to cut short through s and connecting $p(s)$ with s' directly. This reduces the length of the solution and creates a smoother path: A* is in fact characterized by jagged paths due to its constraint of moving by multiples of 45 degrees. The algorithm for A* and its modification into Theta* can be found in [16].

As done in the previous chapter with the TSP solver, a MATLAB function implementing the A* was retrieved and adapted for the occasion. The selected script was developed by A. Chrabieh and is available at MATLAB Central File Exchange [29]. This is a 2D solver that was chosen based on the clarity and readability of the code, due to the necessity for an intense modification. In particular, we had to extend it to three dimensions and introduce LOS checking. This last feature is implemented, as suggested in the appendix of [16], by means of a modified Bresenham line drawing algorithm. Published in 1965 to draw rectilinear segments on rasters, it can easily be modified to check if there is a straight, unobstructed rectilinear path between to points on a discretized map. It simplicity gives it very low computational requirements, hence its short execution times make it a suitable choice.

The 3D extension of this method is made of two instances of the 2D algorithm, working on the XY and XZ projections of the line and merging the coordinates of the blocks. The script we adopted was retrieved and adapted from [30]. Figure 4 shows the result of a illustrative test script for the LOS checking routine. Green lines represent a favourable outcome of the procedure, while red ones imply that one of the walls got in the way.

On a side note, multiple iterations of Bresenham's algorithm can be used to check the visibility for a cylindrical tube centered around the original line, in order to enable a safety clearance around the UAS in pursuit of safer paths.

Another advantage of having a LOS-checking routine is that we can verify the visibility between two consecutive waypoints and decide whether or not path planning is needed at all, potentially saving time and computational resources. We set up a small environment to demonstrate the advantages of the selected method: on a 20-by-50-by-30 cells map, voxel obstacles have been placed so as to disrupt the linear path between the two endpoints, including "trees" with random position and height. Figure 5 shows the results of the planning algorithms in the scenario, and Figure 6 contains their projections in the side and top planes. In particular, the legend of the former states the paths' lengths in cells: we can see that Theta* yields a 12% shorter route with respect to A* (that is, $D_{\Theta}* = 88\% D_{A}*$).

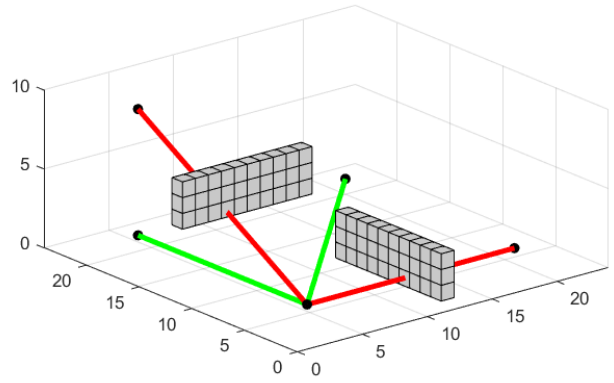

Figure 4. LOS checking test

IV. Control system

The control system relies, to date, solely on PID control and is organized by concentric control loops, each one providing the set point for the next or, eventually, the control inputs for the system itself as outlined in II-B:

- the outermost provides an attitude request based on the distance from the goal,
- the middle one provides an angular rate request based on the angular distance from said command,
- and the last loop listens to these rate commands in order to generate the actual control inputs (u_2, u_3, u_4, which are torques) to the plant.

A further controller regulates the altitude by commanding the cumulative rotor thrust u_1, without interacting with the others.

At the time of the actual hardware assembly, a fourth array of controllers will be required so that the propellers are spun at the exact speeds requested by the controller.

Inputs to the controller are the coordinates of the current waypoint and the 12-state vector of the system, composed of ground-referenced position (X,Y,Z), Euler angles (φ, θ, ψ), linear velocities $(\dot{X}, \dot{Y}, \dot{Z})$ and body frame angular rates (p, q, r) as output by the dynamic model. The sensors and their relative noises have not been modeled. The architecture is visualized in Figure 7.

Before simulation, a waypoint enrichment routine adds points to the original sequence by linear interpolation between consecutive waypoint pairs spaced by a predetermined amount. This ensures a smoother, more regular control action by driving the system by small steps instead of feeding it large position errors. More refined strategies are under investigation at the time of writing.

The setpoint for the first control loop is decided by a dedicated selection routine, which contains an indexed list of

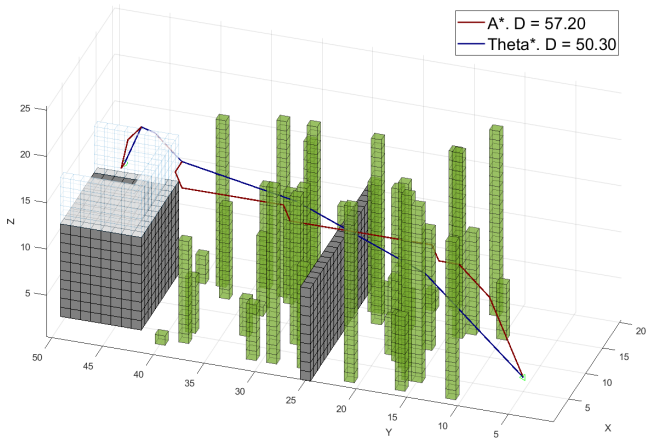

Figure 5. Path planning behaviour in a reference scenario

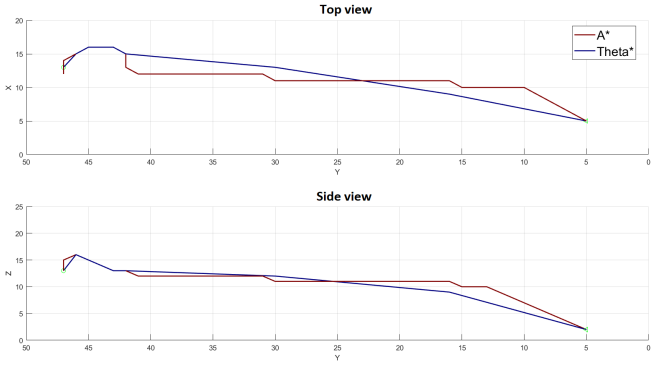

Figure 6. Projection of the resulting paths

coordinates (those provided by the path planning algorithm) and picks the next index based on the position of the system and the distance from the current waypoint. When the system is within a defined minimum distance from the current waypoint, this is considered visited and the index is increased, making the selector skip to the next point.

A. Position control loop

The force necessary to move a multi-rotor UAS comes from the attitude: by tilting the frame, to which rotors are fixed, the total force u_1 projects components in the horizontal plane and hence pushes the frame horizontally: the outermost loop requests the necessary attitude changes for the movement.

A rotation operation around the z-axis (ψ) is performed to project the X and Y position errors on the UAS body frame, so as to produce a pitch and roll command respectively. Two PID controllers are contained here, with a saturation to avoid excessive tilting of the frame with consequent decreases in the upward force component. This could lead to higher movement speeds, as the altitude controller (described below) would keep increasing the total force, leading to an increase of the horizontal component that would make the system less safe and precise. The simulations proposed in the following have this limit at 25°.

An attempt to build a control logic for yaw has not

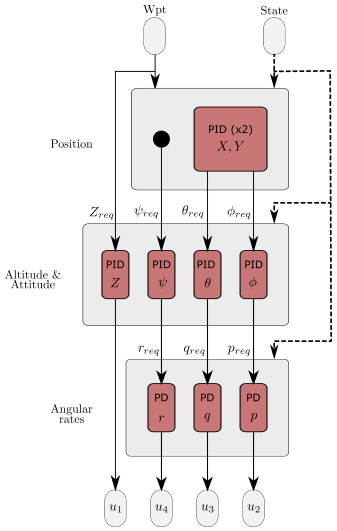

Figure 7. Controller schematic

produced any significant advantage at the time of writing and is left for future work. However, the idea is to calculate the goal ψ as the arctangent of the error components in the body frame

$$\psi_{req} = \arctan\left(\frac{e_Y^b}{e_X^b}\right)$$

so as to direct the system head-on toward the next waypoint: therefore, the position error in the body frame would lie almost entirely in the body-x axis, making the pitch controller work on reducing the distance on its own, while the roll controller concentrates on minor course-correction actions. The flight dynamics, therefore, would resemble more closely that of a fixed-wing system.

B. Attitude control

The middle loop controls the attitude based on the reference given by the position control. Three PID controllers gather the respective roll, pitch and yaw commands from the outer loop and compare them with their feedback homologous, generating body-framed angular rate commands (p, q, r).

C. Altitude control

The Z controller calculates the altitude error from the current goal in the ground frame and rotates it into the body frame:

$$e_Z^G(t) = Z_{req} - Z^G(t);$$
$$e_Z^b(t) = \frac{e_Z^G(t)}{\cos\varphi\cos\theta}$$

This signal is then fed to the altitude PID controller, to whose result the contribution of gravity is added. The feedforward of gravitational action rids the controller of an unnecessary burden (the gravitational action is constant with good approximation) and allows it to focus solely on altitude corrections.

Finally, the signal is saturated with the minimum and maximum commands available from the actuators, $u_{1,min}$ and $u_{1,max}$. For example, for u_1:

$$u_{1,max} = K_T \sum_{i=1}^{4} \omega_{i,max}^2 = 4K_T\omega_{max}^2$$

and similarly it goes for the minimum.

D. Angular rates loop

The innermost loop generates the three rotational control actions to drive the angular velocities along the body axes in order to achieve the requested Euler angles.

A Proportional and Derivative (PD) architecture has been preferred over the PID configuration in pursuit of a better behaviour: we found that the addition of the integral term would extend the settling time and, most importantly, cause an overshoot that could hinder the precision of the control in the wider picture. Again, saturations have been enforced to limit the control action to the ones actually available by the rotors, similarly to what done above for the Z-loop controller.

V. Results

To demonstrate the functionality of the planning and guidance algorithms, two typical missions have been simulated and analysed to gauge the UAS' ability to follow a predefined path in the presence of obstacles: a grid mission with obstacles and an optimal path obtained by the strategy described in the previous section with 30 waypoints. The simulation results are achieved using software code in MATLAB/Simulink. The vehicle under consideration is a small UAS, precisely MAVTech 4QT [24].

The versatile plotting capabilities of MATLAB have been exploited to visualise the results of the simulations by means of a voxel plotter found, again, online and substantially modified to fit our needs [31].

A. Grid Pattern

Our simulations start with a classic grid pattern, with a wall-like obstacle placed in the middle, interfering with the original trajectory so as to showcase the work of Theta*. This type of mission is useful in all situations where the coverage of the entire field is required, like photogrammetric surveys and monitoring operations. Figures 8 and 9 show the trajectories (ideal, planned and simulated) and some relevant simulation parameters (Euler angles and control inputs) respectively. The red dashed line is the response of the planner to the presence of the wall, detected in the mapping phase prior to planning and disrupting the ideal grid (blue, solid line). The solid yellow line is the trajectory flown by the system. For increased safety, the UAS will deploy a Sense and Avoid system to detect barriers in real time during the flight, and to modify the path to avoid them.

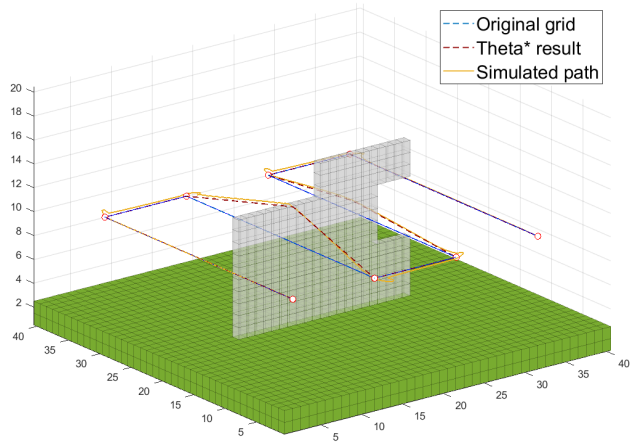

Figure 8. Grid path scenario

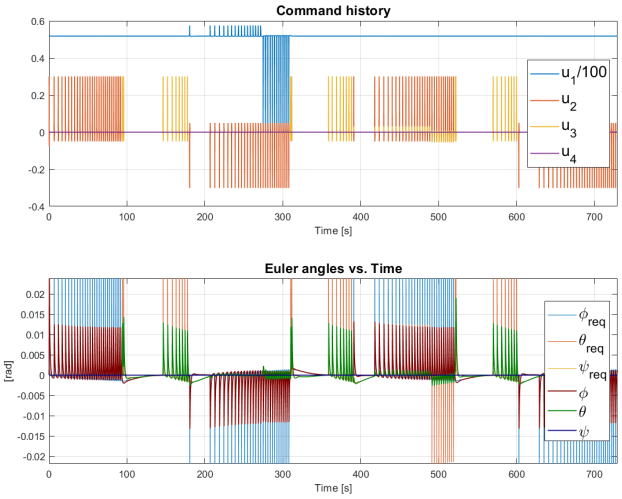

Figure 9. Control inputs and Euler angles

B. Random waypoint-based path

The visualisation of figure 10 is based on the likely morphology of typical Mediterranean vineyards on hillsides, pseudo-randomly generated with rows spaced by three metres and a plant each metre along the rows. The position of each plant is registered in an array while generating the rows, from which the desired number of plants to treat is randomly picked. These, represented by white circles in Figure 10, will constitute the waypoints with an overhead clearance of 4 metres. This height was added for testing purposes, but can be adjusted according to mission requirements. The map created is a square with 50-metres sides and a maximum height of circa 26 metres. In this space, as a demonstration, 30 waypoints are inserted to show the work of TSP together with Theta* and the control algorithm. Again, the red dashed line represents the result of Theta*, following the waypoints sequence obtained through TSP, and the yellow line shows the simulated trajectories.

In Figure 11, the history of controls and Euler angles

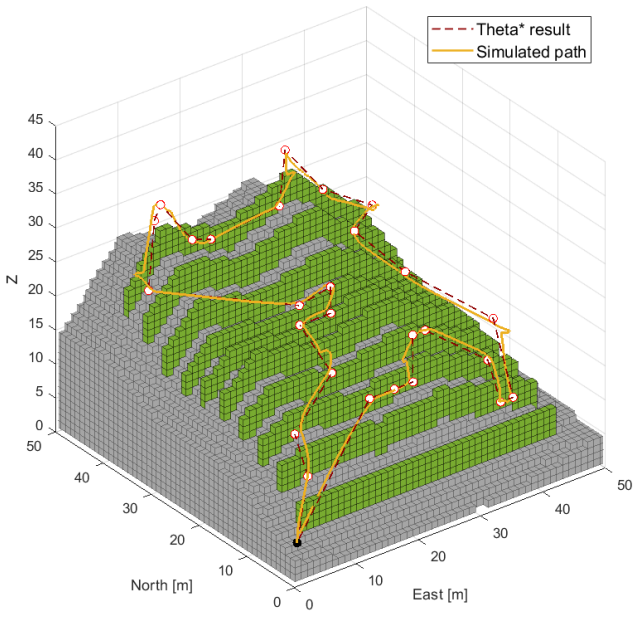

Figure 10. Trajectories for the 30-waypoints scenario

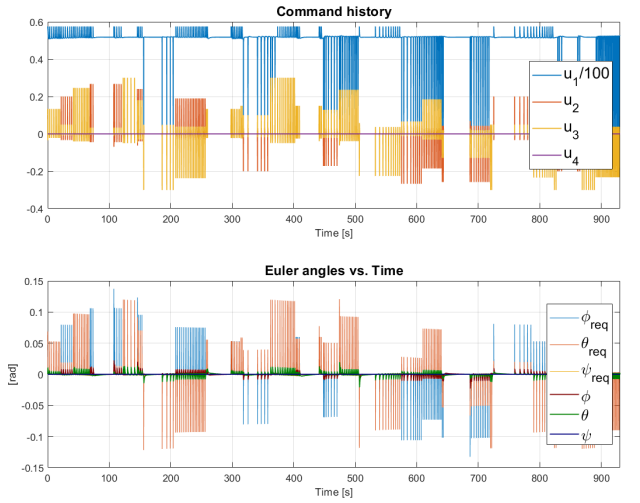

Figure 11. Control inputs and Euler angles

required and obtained is shown.

VI. Conclusions

In this paper, we have outlined a preliminary solution to plan optimal and safe trajectories in agricultural spraying applications.

Results are obtained for two cases proposed to highlight the work of Theta* combined with the TSP. The next step foresees the introduction of an energy consumption model and mass estimation to account for the delivery of PPP during the mission. These aspects will improve the algorithm in terms of energy minimization and accuracy of the control. The idea is to extend the TSP solver to produce lowest-energy trajectories, therefore shifting the optimised variable from the length of the tour to the energy expense, which are not necessarily bound by a univocal relation. The energy model will enable UAS endurance estimations to plan missions as well as recharge times.

As stated in section III-A, the selected solver can handle variations of the problem such as the Capacitated Vehicle Routing Problem (CVRP) and the Multiple Traveling Salesmen Problem (MTSP). The first allows to optimise the route accounting for the amount of product contained in the tank and the amount that must be delivered at each plant, in case warning the user that the tank will not be sufficient to cover the field in a single tour; the second one enables the use of multiple drones to cooperate in covering larger fields. The two algorithms can be combined with a little further effort, deploying a CVRP with multiple vehicles.

Another important aspect will be to introduce and simulate external disturbances, such as wind, to produce a more robust control system and to define a wind speed limit for our application. This study would also enable heading angle control, which besides improving on the trajectory will be of paramount importance in the spray accuracy over the target location.

As work progresses, experiments will be performed via laboratory testing at first, and then in actual vineyards to validate the proposed algorithm.

Acknowledgment

This research was funded by the project “New technical and operative solutions for the use of drones in Agriculture 4.0” (Italian Ministry of University and Research - Progetti di Ricerca di Rilevante Interesse Nazionale – PRIN 2017, Prot. 2017S559BB).

References

[1] M. De Clercq, A. Vats, and A. Biel, “Agriculture 4.0: The future of farming technology,” *Proceedings of the World Government Summit, Dubai, UAE*, pp. 11–13, 2018.

[2] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, “A review on uav-based applications for precision agriculture,” *Information*, vol. 10, no. 11, p. 349, 2019.

[3] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, “A compilation of uav applications for precision agriculture,” *Computer Networks*, vol. 172, p. 107148, 2020.

[4] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, and P. Gay, “Cooperative agricultural operations of aerial and ground unmanned vehicles,” in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2020, pp. 224–229.

[5] E. Bassi, N. Bloise, J. Dirutigliano, G. P. Fici, U. Pagallo, S. Primatesta, and F. Quagliotti, “The design of gdpr-abiding drones through flight operation maps: A win–win approach to data protection, aerospace engineering, and risk management,” *Minds and Machines*, vol. 29, no. 4, pp. 579–601, 2019.

[6] E. A. S. Agency, “Introduction of a regulatory framework for the operation of drones,” https://www.easa.europa.eu/sites/default/files/dfu/A-NPA%202015-10.pdf, accessed: 2021-01-13.

[7] N. Bloise, S. Primatesta, R. Antonini, G. P. Fici, M. Gaspardone, G. Guglieri, and A. Rizzo, “A survey of unmanned aircraft system technologies to enable safe operations in urban areas,” in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 433–442.

[8] M. Asim, D. Ehsan, and K. Rafique, “Probable causal factors in uav accidents based on human factor analysis and classification system,” *history*, vol. 1905, p. 5, 2005.

[9] Ministero dell'Ambiente e della Tutela del Territorio e del Mare, "Piano di azione nazionale per l'uso sostenibile dei prodotti fitosanitari," https://www.minambiente.it/sites/default/files/archivio/allegati/vari/pubbl_PAN.pdf, accessed: 2021-01-13.

[10] N. Bloise, M. C. Ruiz, D. D'Ambrosio, and G. Guglieri, "Preliminary design of a remotely piloted aircraft system for crop-spraying on vineyards," in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2020, pp. 1–6.

[11] L. Comba, A. Biglia, D. R. Aimonino, and P. Gay, "Unsupervised detection of vineyards by 3d point-cloud uav photogrammetry for precision agriculture," *Computers and Electronics in Agriculture*, vol. 155, pp. 84–95, 2018.

[12] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. Jacket, and H. S. Baird, "Handwritten zip code recognition with multilayer networks," in *[1990] Proceedings. 10th International Conference on Pattern Recognition*, vol. 2. IEEE, 1990, pp. 35–40.

[13] M. Kerkech, A. Hafiane, and R. Canals, "Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in uav images," *Computers and electronics in agriculture*, vol. 155, pp. 237–243, 2018.

[14] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search in combinatorial optimization*, vol. 1, no. 1, pp. 215–310, 1997.

[15] G. Gutin, A. Yeo, and A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp," *Discrete Applied Mathematics*, vol. 117, no. 1-3, pp. 81–86, 2002.

[16] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579.

[17] L. De Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for uavs in 3d environments," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 247–264, 2012.

[18] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[19] P. Váňa and J. Faigl, "The dubins traveling salesman problem with constrained collecting maneuvers," *Acta Polytechnica CTU Proceedings*, vol. 6, pp. 34–39, 2016.

[20] P. Váňa, J. Sláma, and J. Faigl, "The dubins traveling salesman problem with neighborhoods in the three-dimensional space," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 374–379.

[21] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[22] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[23] B. T. Clough, "Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway," Air Force Research Lab Wright-Patterson AFB OH, Tech. Rep., 2002.

[24] MAVTech s.r.l., "Drone q4t," https://www.mavtech.eu/it/prodotti/q4t-drone/, accessed: 2021-01-08.

[25] W. Selby, "Arducopter," https://www.wilselby.com/research/arducopter/, accessed: 2021-01-13.

[26] J. Kirk, "Traveling salesman problem (tsp) genetic algorithm toolbox," https://www.github.com/rubikscubeguy/matlab-tsp-ga, accessed: 2021-01-08.

[27] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[29] A. Chrabieh, "A star search algorithm," https://www.mathworks.com/matlabcentral/fileexchange/64978-a-star-search-algorithm, accessed: 2021-01-08.

[30] "Three dimensional line algorithm," https://web.archive.org/web/20110708171823/http://www.cobrabytes.com/index.php?topic=1150.0, accessed: 2021-01-08.

[31] I. B. Shabat, "Voxelplotter," https://www.mathworks.com/matlabcentral/fileexchange/50802-voxelplotter, accessed: 2021-01-05.