J Intell Robot Syst (2016) 83:445–462
DOI 10.1007/s10846-016-0348-x

Coverage Path Planning for UAVs Photogrammetry with Energy and Resolution Constraints

Carmelo Di Franco · Giorgio Buttazzo

Received: 8 July 2015 / Accepted: 27 January 2016 / Published online: 5 February 2016

Abstract Unmanned Aerial Vehicles (UAVs) are starting to be used for photogrammetric sensing of large areas in several application domains, such as agriculture, rescuing, and surveillance. In this context, the problem of finding a path that covers the entire area of interest is known as Coverage Path Planning (CPP). Although this problem has been addressed by several authors from a geometrical point of view, other issues such as energy, speed, acceleration, and image resolution are not often taken into account. To fill this gap, this paper first proposes an energy model derived from real measurements, and then uses this model to implement a coverage path planning algorithm for reducing energy consumption, as well as guaranteeing a desired image resolution. In addition, two safety mechanisms are presented: the first, executed off-line, checks whether the energy stored in the battery is sufficient to perform the planned path; the second, performed online, triggers a safe return-to-launch (RTL) operation when the actual available energy is equal to the energy required by the UAV to go back to the starting point.

Keywords Energy-aware trajectories · Coverage path planning · Unmanned aerial vehicles

C. Di Franco (✉) · G. Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
e-mail: c.difranco@sssup.it

G. Buttazzo
e-mail: g.buttazzo@sssup.it

1 Introduction

Unmanned Aerial Vehicles (UAVs), also referred to as drones, are used in several application domains that require the inspection of large areas for the reconstruction of territorial maps or the identification of specific details. Device miniaturization reduces dimensions and costs and thus enables the use of UAVs in new application fields. Agriculture is one of the emerging sectors in which UAVs are currently being adopted for detecting the state of vegetation and the growth of weeds, in order to plan for a timely and proper intervention [1–3]. In other applications, UAVs are used to inspect industrial plants [4], survey wooded area for fire prevention [5], or quickly reach and monitor hazardous environments to better support disasters interventions and people rescuing.

The sensors needed onboard to carry out a mission depend on the specific application. Typical examples include temperature and humidity sensors, barometric pressure sensors for altitude detection, video cameras, infrared or multispectral cameras, ultrasound sensors for precise distance measurements during landing, and so on. From the actuation perspective, UAVs can be distinguished in two main categories: fixed-wing and multi-rotor. Fixed-wing UAVs are more appropriate for covering large distances and have the advantage of carrying higher payloads. However, they cannot focus on the same scene for long period of time, and thus cannot be used as a remote flying camera controller. Multi-rotor UAVs are more suitable whenever there is

a need of controlling the position and orientation of a camera on a desired detail of the scene, although they have the disadvantage of a lower payload and a lower battery lifetime.

The problem of monitoring geographical zones by UAVs has been mainly addressed with the objective of finding the optimal path able to fully cover a given area of interest. This problem is referred to as coverage path planning (CPP) and most of the work done on CPP only considers geometrical constraints, without taking into account other peculiar features of the drone, such as the available energy, the weight, the maximum speed, and other mission requirements, (e.g., the spatial resolution of the acquired images). This paper aims to bridge this gap by proposing an energy-aware path planning algorithm that minimizes energy consumption while satisfying a set of other requirements, such as area coverage and image resolution.

Contributions The main contributions of the paper can be summarized as follows:

1. An energy model is derived from real measurements to estimate the power consumption of the drone as a function of its speed in different operating conditions.
2. An energy-aware algorithm is proposed to determine a path that reduces energy consumption while satisfying coverage and resolution constraints. The optimal number of stripes in a back-and-forth path is determined for convex areas.
3. An online energy checking mechanism is presented to guarantee a safe return-to-launch in case the residual battery charge is not sufficient to complete the planned path.

Paper organization The remainder of this paper is organized as follows: Section 2 illustrates the related work; Section 3 presents the system model and derives the relation among the states variables used to formalize the problem; Section 4 explains the method used to identify the speed that minimizes the energy of a given straight trajectory; Section 5 describes the energy-aware path planning algorithm; Section 6 presents two different mechanisms for providing additional safety guarantees during the UAV mission; Section 7 reports a set of experiments performed on a real system to validate the proposed approach; and finally, Section 8 concludes the paper and presents some future research directions.

2 Related Work

Coverage path planning is a problem that has been addressed extensively in literature. Galceran and Carreras [6] reviewed and evaluated several (CPP) algorithms based on different methods, such as cellular networks, grids, graphs, neural networks, with online and off-line computation, and for known or unknown areas. Maza et al. [7] presented a CPP algorithm for multiple UAVs that partitions the area of interest based on the number of drones, their capabilities and their initial locations. In particular, the capability of a drone is measured by the maximum distance it can cover with a given energy. After decomposing the area, each UAV computes the sweep direction that minimizes the number of turns required in a trajectory following a back-and-forth pattern. Barrientos et al. [8] presented a task manager that automatically partitions a search area for a team of UAVs based on negotiation among the vehicles, considering their state and capabilities. Öst [9] analyzed two different types of trajectories for covering convex and concave areas, the back-and-forth and the spiral patterns, combining them with some area decomposition algorithms. Santamaria et al. [10] presented a path planning algorithm for multiple heterogeneous UAVs that also considers different sensor footprints.

Bast and Hert [11] investigated the problem of partitioning an arbitrary polygon into the minimum number of convex areas, considering that the resulting subspaces must be suited to the robotic applications. They proved that partitioning an area with minimal cut length is NP-hard and proposed a polynomial-time algorithm capable of producing a non-optimal but reasonable partitioning. Huang [12] presented an optimal line-sweep-based decomposition algorithm that minimizes the amount of time needed to cover an area including obstacles. Dynamic programming is used to find the optimal decomposition by assuming the knowledge of the area boundaries and the position of the obstacles present inside it.

A different approach was proposed by Lawrance and Sukkarieh [13], who formulated the path planning

problem by taking into account the energy gain produced by the wind. The proposed algorithm is capable of generating energy-gain trajectories using both static and dynamic soaring. Al-Sabban et al. [14] proposed a method for exploiting the wind energy to extend the flight duration of a UAV during the route from a starting point to another.

Roberts et al. [15] derived an energy model of a hover-capable flying robot and proposed an algorithm that mitigates the energy consumption in an indoor aerial exploration by using ceiling attachment as a means for preserving energy while maintaining the camera contact with the target.

Mei et al. [16] investigated the problem of deploying mobile wheel robots with energy and timing constraints. A speed management policy is used to reduce energy consumption and area constraints, maximum mission time, and obstacles are taken into account for computing the minimum number of robots to be deployed.

One of the main limitations of the energy-aware approaches mentioned above is that energy consumption is modeled and accounted for trajectories executed at a constant speed, neglecting more complex maneuvers performed under acceleration. Also, no algorithms considered image resolution as an additional constraint for the solution.

A preliminary study of a CPP algorithm including energy consumption and image resolution constraints has been presented by Di Franco and Buttazzo [17]. This paper extends such a study reducing the path length and by adding an online energy failsafe mechanism that allows a safe (RTL) in case the residual battery charge is not sufficient to complete the planned path. Also, additional experiments are presented for validating the energy model and setting the fail-safe threshold on the battery discharge.

3 System model

3.1 Camera Model

The UAV considered in this paper is an IRIS quadrotor with a GoPro camera mounted on a Gimbal stabilizer for compensating small rotation errors experienced during flight (Fig. 1). The goal of the mission is to scan a given area and reconstruct its map with a spatial resolution R greater than or equal to R_d, expressed in pixels/cm.

Fig. 1 The IRIS quadcopter with a GoPro camera mounted on a Gimbal stabilizer

The video camera used to take pictures has the following parameters:

- Angle of view (AOV), α, expressed in radians, represents the angular extent of a scene captured by the camera;
- Image Resolution (I_x, I_y), expressed in pixels for both sides of the image;
- Aspect ratio $\rho = I_x/I_y$ between the width and height of the image;
- Minimum sampling period T_s^{min} between two consecutive shots.
- Exposure time T_e (or shutter speed) represents the time interval in which the shutter is open and the camera sensor is exposed to light.

When a drone flying at height h acquires a picture with the camera pointing down, the corresponding portion of the area is called *projected area*. The size of the projected area depends on the height h and the angle of view α, as illustrated in Fig. 2.

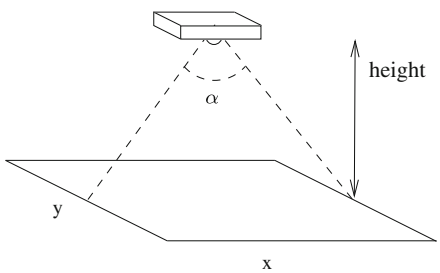

Fig. 2 Projected area of a camera with AOV α placed at height h

If the camera is parallel to the ground (this assumption is realistic considering that the camera is mounted on a stabilizer), the size (L_x, L_y) of the projected area can be computed as

$$\begin{cases} L_x = 2h \cdot \tan\left(\frac{\alpha}{2}\right) \\ L_y = L_x/\rho \end{cases} \tag{1}$$

Hence, the spatial resolution R obtained by taking a picture at height h is

$$R = \frac{I_x}{L_x} = \frac{I_x}{2h \cdot \tan\left(\frac{\alpha}{2}\right)}. \tag{2}$$

By substituting Eq. (2) in the inequality imposed by the mission requirement ($R \geq R_d$) we have that

$$h \leq \frac{I_x}{2R_d \cdot \tan\left(\frac{\alpha}{2}\right)}. \tag{3}$$

Hence, the mission requirement imposes a constraint on the maximum height the UAV can fly, which is

$$h_{max} = \frac{I_x}{2R_d \cdot \tan\left(\frac{\alpha}{2}\right)}. \tag{4}$$

To reconstruct the map using photogrammetry, the area of interests needs to be decomposed into a sequence of rectangles of size (L_x, L_y) and the drone trajectory must be programmed to pass through their centers. The complete path is then stored as a list of coordinates, called waypoints, and the drone moves from one waypoint to the next until it reaches the last element in the list. Other parameters are included in the path, such as the GPS coordinates, the height, the delays between waypoints, the speed, etc.

Note that, to correctly integrate the acquired images, projected areas must overlap as illustrated in Fig. 3. The amount of overlap can be chosen by the user and can vary on each side. The horizontal and vertical overlaps are denoted as ov_x and ov_y, respectively. The distance between the centers of two adjacent areas along the wider direction (also equal to the distance between two adjacent stripes) is then $d_s = L_x - ov_x$, whereas the distance between adjacent areas along the path direction (also equal to the distance between consecutive waypoints) is $d_w = L_y - ov_y$.

Note that the sampling period T_s of the camera (i.e., the interval between two consecutive shots) imposes another constraint on the maximum speed of the UAV. In particular, the space covered by the UAV between two consecutive image acquisitions must satisfy the minimum specified overlap, that is $v \cdot T_s \leq L_y - ov_y$. Therefore,

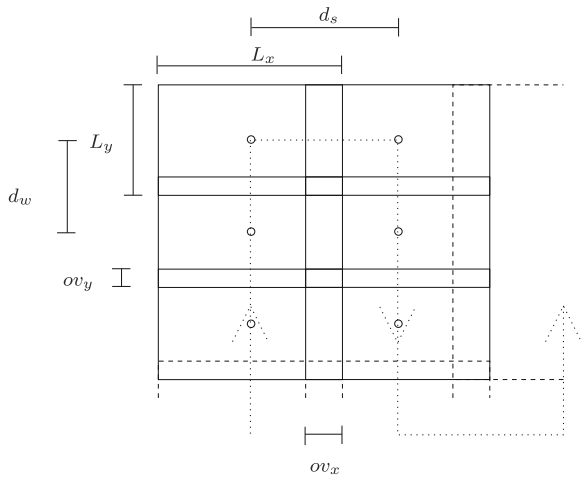

Fig. 3 Projected areas with overlaps. Centers of rectangles are the path waypoints

$$v \leq \frac{L_y - ov_y}{T_s}. \tag{5}$$

Similarly, the exposure time T_e imposes a constraint on the speed. This parameter varies from $1/16000\ s$ to $30\ s$ depending on light conditions. Photos taken during the day usually have a really short exposure time (e.g., $1/580\ s$) while night photos have values between $1/10\ s$ to $30\ s$ depending on how much light is present in the scene. If the camera moves while the shutter is open it will capture light emitted from a wider area producing a blurred image. This leads to another constraint on the speed, that is $v \cdot T_e \leq \delta/R$, where δ is the allowed amount of blur in pixels. Therefore,

$$v \leq \frac{\delta}{R \cdot T_e} = \frac{\delta \cdot 2h \cdot \tan\left(\frac{\alpha}{2}\right)}{I_x \cdot T_e}. \tag{6}$$

For applications that make use of specific sensors with long exposure time this constraint becomes relevant.

Considering the constraints imposed by Eq. (5) and (6), the maximum speed of the UAV can be expressed as:

$$v_{max} = \min\left(\frac{L_y - ov_y}{T_s}, \frac{\delta}{R \cdot T_e}\right). \tag{7}$$

For the sake of clarity, Table 1 summarizes the main parameters used in this section.

Table 1 List of the main system parameters

Variable	Name
α	angle of view (rad)
f	focal length (mm)
I_x	image width (pixels)
I_y	image height (pixels)
ρ	aspect ratio
T_s	camera sampling period (s)
T_e	camera exposure time (s)
R	spatial resolution (pixels/cm)
δ	maximum allowed amount of blur (pixels)
h	height (m)
L_x	horizontal length of the projected area (m)
L_y	vertical length of the projected area (m)
ov_x	horizontal overlap (m)
ov_y	vertical overlap (m)
d_s	distance between parallel paths (m)
d_w	distance between waypoints (m)

3.2 LiPo Battery characteristics

LiPo batteries are gaining favor in the world of radio-controlled aircraft thanks to the lower weight, increased capacity, and power delivery. The specific characteristics of a LiPo battery may affect the performances and lifetime of an UAV. A LiPo battery may have in its pack one or more cells connected in series or parallel. The voltage of a LiPo cell varies from about $2.7 - 3.0$ V (discharged) to about $4.20 - 4.35$ V (fully charged) depending on its chemistry. Each battery pack must specify the nominal voltage, the number of cells (in series or in parallel), the maximum capacity, and the discharge rate. The number before the S is the number of cells in series to give your pack voltage. The number before the P is the number of parallel batteries in your pack. The nominal voltage is expressed in Volt and can be computed by multiplying the number of cells in series with their voltage. The capacity is expressed in mAh and specify the theoretical current I_H draw under which the battery would deliver its nominal rated capacity in one hour. The C-rate is a measure of the rate at which a battery is being discharged. It is defined as the discharge current divided by I_H. The C-rate also indicates that the maximum current that it is possible to supply can be computed as C times I_H. For example, a LiPo battery expressed as 11.V 3S1P 5500 mAh 30C means that the battery has 3 cells in series with a global nominal voltage of 11.1 V, a capacity of 5500 mAh and a C-rate of 30.

If there is no load the voltage level is slightly higher. At the beginning of the discharging process, the voltage quickly drops to the nominal voltage value and decreases very slowly during the normal activity until a final phase, where the voltage falls down rapidly up to a complete discharge. An example of discharging voltage curve for a LiPo cell is shown in Fig. 19. The final phase is the most critical because the battery voltage rapidly decreases and due to this, batteries must have always a remaining capacity greater than 30–40 % at the end of the flight. This final phase needs to be controlled carefully to avoid damage to the batteries or to the UAVs that may not have enough energy to finish their goal.

3.3 Energy model

Due to the wide variety of drones available today on the market, each with peculiar characteristics, i.e. different weights, dimensions, propellers, and types of motors, deriving a parametric model that can be used for predicting the energy consumption in different flying conditions is not a trivial task. Therefore, this paper presents a method for modelling and analyzing the energy consumption of a specific drone as a function of its speed, acceleration, and flying maneuvers. Although the derived energy model is specific to the drone used, the proposed method can be applied to any UAV by performing the same measurements and without modifying any equation.

In order to derive a model that can be used effectively in the analysis, a set of experiments has been carried out to understand how energy consumption is affected by the different flying conditions, such as speed, horizontal and vertical accelerations. The drone considered in this paper and used in the experiments is an IRIS quadrotor controlled by a PX4 autopilot board and equipped with a GoPro camera mounted on a Gimbal stabilizer. It is driven by four 850 KV brushless motors and powered with a 3S LiPo battery (11.1 V 5.5 Ah), and weighs approxximately 1.3 Kg in total.

The Gimbal stabilizer uses two brushless motors to compensate for rotation errors and keep the camera always aligned toward a specified direction. The GoPro camera can take pictures at three different resolutions (5, 7, and 12 Megapixels) and two different

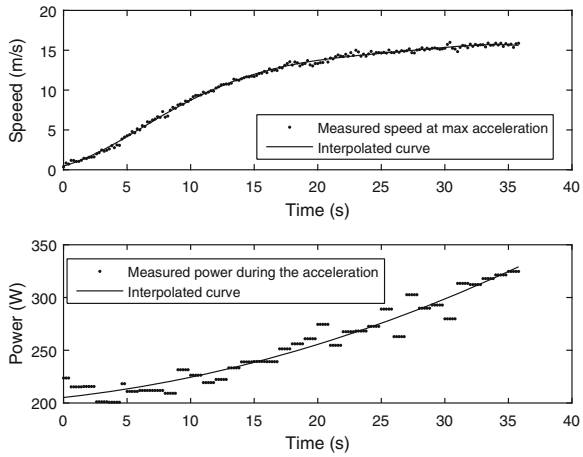

Fig. 4 Speed and power consumption acquired during maximum acceleration

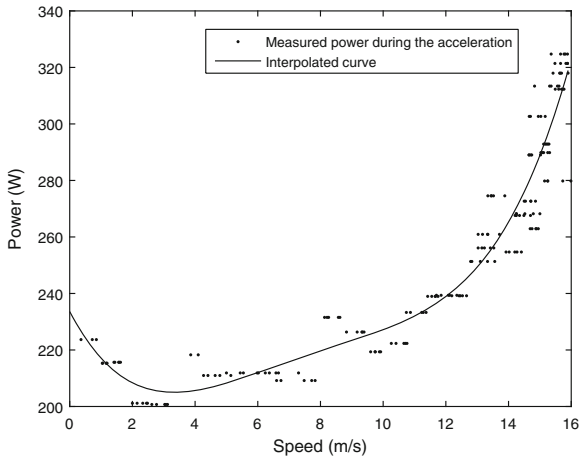

Fig. 6 Power consumption during maximum acceleration as a function of the speed

AOV (94.4 and 122.6 deg). The PX4 board runs the open-source APM Ardupilot.

The first experiment was carried out to monitor the power consumption of the drone as a function of the flying speed, reached with maximum acceleration. The speed was monitored from the onboard GPS and the absorbed current was acquired from the control board. The consumed power was then derived, for each speed, by multiplying the absorbed current by the supply voltage. Figure 4 shows the speed and the absorbed power as a function of time, acquired under the maximum acceleration, along with the corresponding fitted curves. Similarly, Fig. 5 reports the same variables acquired under the maximum deceleration. Figures 6 and 7 also show the power absorbed during the maximum acceleration/deceleration but as a function of the speed.

Given the power curve $P_a(t)$, the energy consumed to vary the speed from v_1 to v_2 with a given acceleration a can be computed as

$$E_a(v_1, v_2) = \int_{t_1:v=v_1}^{t_2:v=v_2} P_a(t)\, dt. \tag{8}$$

A second experiment was performed to derive the power consumption as a function of the speed in different flight conditions, such as horizontal flight, climbing, descending, and hovering. The results of

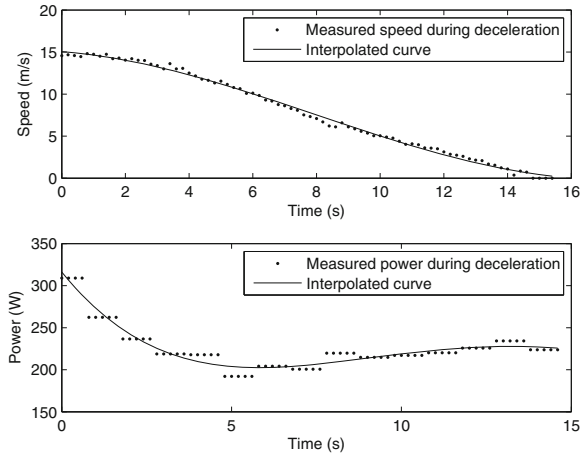

Fig. 5 Speed and power consumption acquired during maximum deceleration

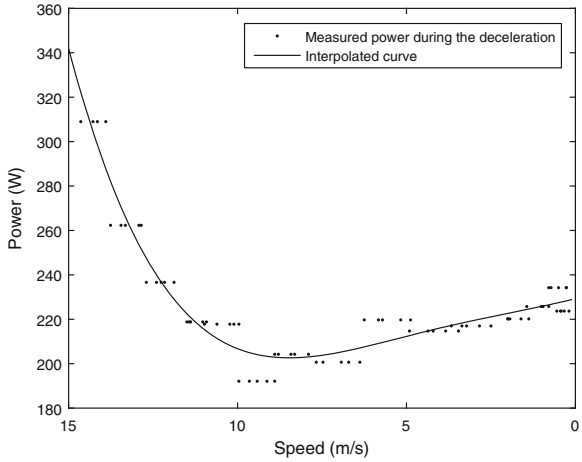

Fig. 7 Power consumption during maximum deceleration as a function of the speed

this experiment, along with the fitted curve, are reported in Fig. 8. Each point in the figure corresponds to the average power consumption in a specific flight condition (e.g., constant speed), which has been recorded for at least 10 seconds. Observe that, since climbing and descending operations are always performed at a constant speed (v_{climb} and v_{desc}, respectively), the corresponding power consumptions, P_{climb} and P_{desc}, are plotted in the graph as two points (denoted by '+' and '*', respectively). Also observe that the power P_{hover} consumed during hovering corresponds to the the point in the graph for $v = 0$.

Once the function $P(v)$ is derived, the energy consumed by the drone to cover a distance d in a straight flight at a constant speed v can be computed as

$$E_0(v, d) = \int_0^{d/v} P(v)\, dt = P(v)\frac{d}{v}. \tag{9}$$

The energy consumed during climbing and descending to cover a height displacement Δh can be computed as

$$E_{climb}(\Delta h) = \int_{h_1/\hat{v}_{climb}}^{h_2/\hat{v}_{climb}} P_{climb}\, dt = P_{climb}\frac{\Delta h}{v_{climb}} \tag{10}$$

$$E_{desc}(\Delta h) = \int_{h_2/\hat{v}_{desc}}^{h_1/\hat{v}_{desc}} P_{desc}\, dt = P_{desc}\frac{\Delta h}{v_{desc}} \tag{11}$$

$$\tag{12}$$

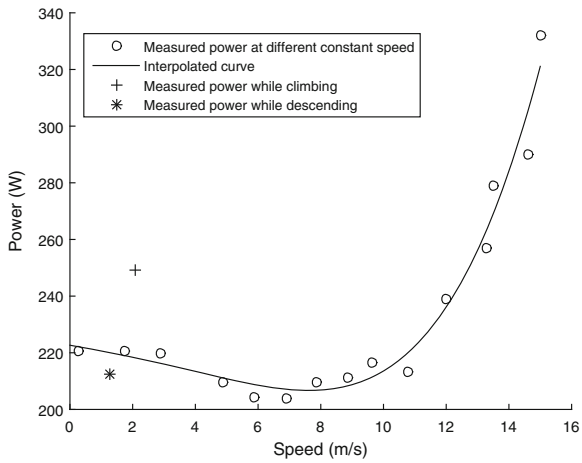

Fig. 8 Fitted curve of the consumed power as a function of the speed. Each point corresponds to the average power consumption when flying at a constant speed, which has been recorded for at least 10 s. Power consumption during climbing, descending, and hovering are also reported as single points

Finally, the energy consumed during hovering in an interval $[t_1, t_2]$ can be computed as

$$E_{hover} = \int_{t_1}^{t_2} P_{hover}\, dt = P_{hover}(t_2 - t_1). \tag{13}$$

A third experiment was carried out to measure the time and the power needed during rotations. Assuming that both the angular speed $\omega_{turn}(2.1\ rad/s)$ and the corresponding power $P_{turn}(225\ W/s)$ can be considered to be constant during rotations, the energy required to cover an angle $\Delta\theta$ can be computed as

$$E_{turn} = P_{turn}\frac{\Delta\theta}{\omega_{turn}}. \tag{14}$$

4 Finding the optimal speed

The goal of this section is to compute the speed that minimizes the energy needed to cover a given distance d in a straight flight. The solution is first derived in the simple case of constant speed, and then extended to the more general case of variable speed, taking into account the acceleration profile recorded during the experiments.

4.1 Constant speed

The energy E_0 consumed by the drone for covering a distance d at a constant speed v during a straight flight can be computed by Eq. 9. The optimal speed can then be found by minimizing the energy per unit distance, $E_0^u(v)$, defined as

$$E_0^u(v) = \frac{E_0(v, d)}{d} = \frac{P(v)}{v}. \tag{15}$$

The function $E_0^u(v)$ is illustrated in Fig. 9, which clearly shows a minimum for speed $v^* \simeq 12$ m/s.

4.2 Variable speed

In the presence of accelerations, the energy minimization problem is solved for trajectories of length d consisting of three distinct phases: an initial phase executed at maximum acceleration, an intermediate phase at constant speed v, and a final phase at maxi-

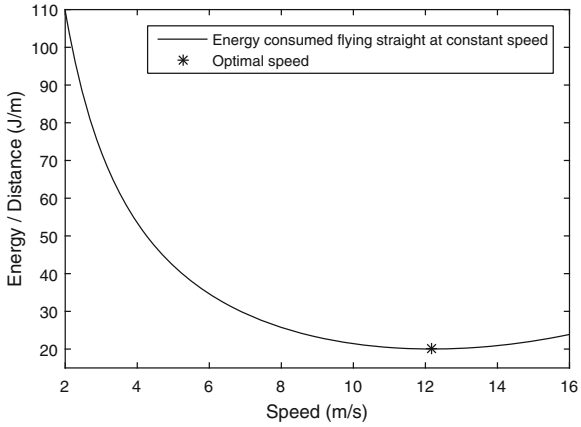

Fig. 9 Energy per unit distance as a function of the (constant) speed

mum deceleration. Then, the optimal speed v^* can be derived by minimizing the following function:

$$E_a(v,d) = \int_0^v P_{acc}(v)\,dv + \int_0^{t(v)} P(v)\,dt + \int_0^v P_{dec}(v)\,dv \qquad (16)$$

where $P_{acc}(v)$, $P(v)$ and $P_{dec}(v)$ denote the functions derived in the experiments reported in Section 3.3 by interpolating data with fifth order polynomials, and

$$t(v) = \frac{d - \int_0^{t:v_{acc}(t)=v} v_{acc}(t)\,dt - \int_0^{t:v_{dec}(t)=v} v_{dec}(t)\,dt}{v} \qquad (17)$$

where $v_{acc}(t)$ and $v_{dec}(t)$ are the polynomial fitted functions of the speed during the acceleration and deceleration phase illustrated in Figs. 4 and 5. Using Eq. (17), Eq. (16) can be expressed as:

$$E_a(v,d) = \int_0^v P_{acc}(v)\,dv + P(v)\cdot t(v) + \int_0^v P_{dec}(v)\,dv. \qquad (18)$$

Note that Fig. 10 shows the energy $E_a(v,d)$ consumed during flight as a function of the speed for different given distances d. For each curve, the figure also indicates the optimal speed $v^*(d)$. Note that for increasing distances, the contribution of the intermediate phase becomes dominant with respect to the other two phases, thus for long distances ($d > 300$ m) the values of $v^*(d)$ tend to the optimal value

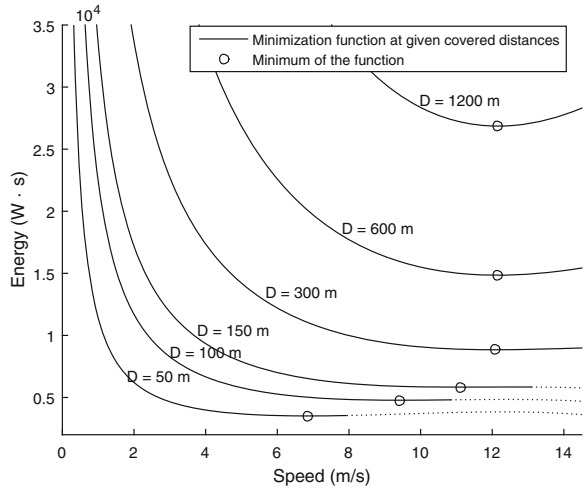

Fig. 10 Energy consumed as a function of the speed for different given distances d

($v^* \simeq 12$ m/s) computed in the case of constant speed. Function $v^*(d)$, which indicates how the optimal speed varies as a function of the covered distance, is illustrated in Fig. 11. This result is used in this paper to set the optimal speed for each straight line of length d in the back-and-forth path of the scanned area.

5 Path planning

The scanned area is modeled as a polygon described by an ordered set of p vertices $\{v_1, \ldots, v_p\}$. Each vertex v_i is characterized by a pair of coordinates

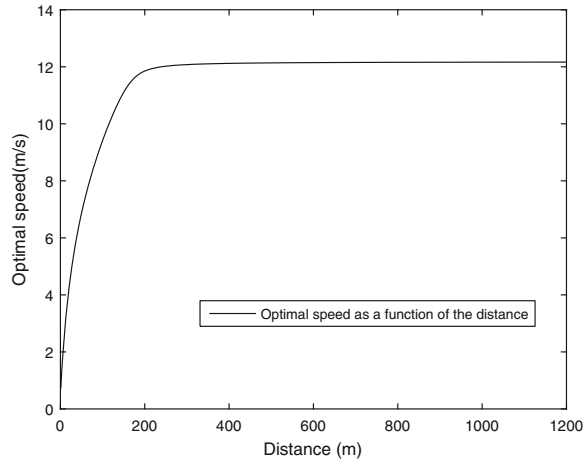

Fig. 11 Optimal speed as a function of the covered distance d

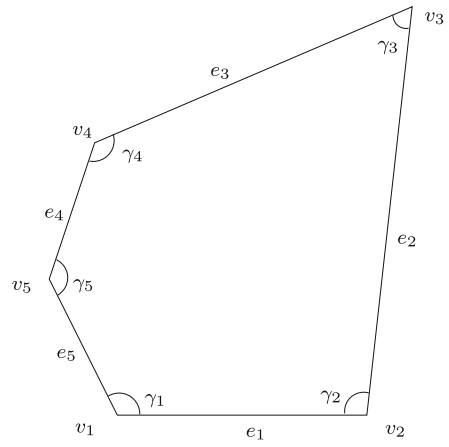

Fig. 12 Example of an area with its vertices and edges

$(v_x(i), v_y(i))$ and its inner angle, denoted by γ_i. For each vertex v_i, the next vertex of the polygon in the considered order is denoted by $v_{next(i)}$, where $next(i) = i \pmod p + 1$. The edge between a pair of consecutive vertices v_i, $v_{next(i)}$ is denoted by e_i, and its length by $l_i = ||v_i - v_{next(i)}||$, as illustrated in Fig. 12. We assume that the area of interest is convex (that is, $\forall i, \gamma_i < \pi$), and bounding lines do not intersect to each other.

To minimize the number of back and forth paths while guaranteeing a desired image resolution R_d, the drone is programmed to fly at the maximum admissible height h_{max} computed by Eq. 4. The rest of this section describes an algorithm for scanning the area with a back-and-forth path taking energy consumption into account.

5.1 Back-and-forth path

The area of interest is scanned using a back-and-forth trajectory specified by a set of waypoints. For a given maximum speed, the overall scanning time is highly dependent on the number of turns, since on each turn the drone has to decelerate, rotate, and accelerate again. Figure 13 illustrates two examples of scanning paths, for a given area, characterized by a different number of turns.

A lot of work has been done in the literature to determine the scan direction that minimizes the number of turns [6, 12]. The major problem of these algorithms is that, in same cases, the resulting trajectories may contain crossing lines necessary to make

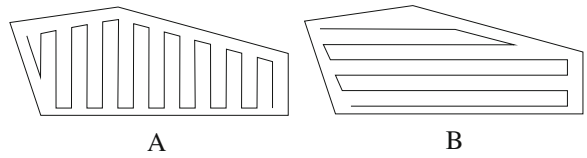

Fig. 13 The number of turns in path A is higher than in path B

a close path. In the proposed solution this problem is avoided by considering an initial phase and a final phase that reduce the travel distance from and to the starting point, also avoiding crossing paths. If the area has a high number of concavities, finding the optimal direction can be quite difficult. However, in UAV applications the shape of the area is not a real concern because in most situations: a) the survey area is typically described by a small number vertices, and b) above a certain height there are no obstacles. Under these two assumptions, this section presents a simple but effective algorithm that contains the number of turns by setting the scanning direction parallel to the longest bounding line. The algorithm computes the number of stripes to reduce the overall path length and improve the overlap between images.

Given a polygonal area $A = \{v_1, \ldots, v_p\}$, the following special vertices are defined to generate the path:

v_{start}	is the vertex closest to the starting point of the scanning path;
v_{scan}	is the vertex corresponding to the longest edge (e_{scan});
v_{far}	is the vertex with the largest distance from e_{scan}.

Without loss of generality, we set $v_1 = v_{start}$ and chose the ordering that makes the index $scan$ smaller. For instance, considering the area illustrated in Fig. 12, we have $v_{start} = v_1$, $v_{scan} = v_2$, and $v_{far} = v_5$. The scanning path consists of three parts:

1. An initial path going from v_{start} to v_{scan} along the borders;
2. A back-and-forth path starting from v_{scan} scanning the area along the scan direction until reaching the vertex v_{far};
3. A final return path going from v_{far} to v_{start}.

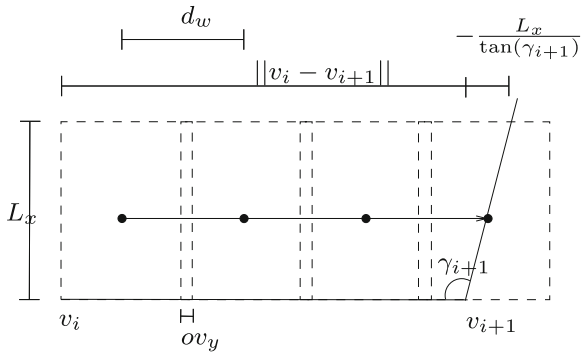

Fig. 14 Case in which the projected area on the border contains a region out of interest

To reduce the total energy consumption, for every straight path of length d the speed is set to the optimal value computed in Section 4.

The distance d_w between consecutive waypoints in a straight path is set as $d_w = L_y - ov_y$. The number n_w of waypoints along a straight path of length d is computed as

$$n_w = \left\lceil \frac{d - ov_y}{d_w} \right\rceil. \tag{19}$$

Note that, since n_w is rounded up, the projected area of the last waypoint may include a region that is out of interest, as illustrated in Fig. 14. This problem can be mitigated by increasing the overlap at the value $\hat{ov}_y$ such that $n_w(L_y - \hat{ov}_y) + \hat{ov}_y$ becomes exactly equal to d, as shown in Fig. 15. That is,

$$\hat{ov}_y = \frac{n_w L_y - d}{n_w - 1}. \tag{20}$$

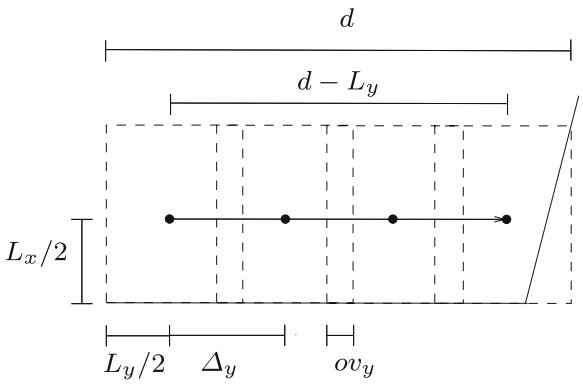

Fig. 15 Projected areas when the overlap is increased to reduce the region out of interest

In this way, the distance between two waypoints becomes

$$\hat{d}_w = \frac{d - L_y}{n_w - 1}. \tag{21}$$

Similarly, the distance d_s between stripes is computed as $d_s = L_x - ov_x$ and the number n_s of stripes is computed as

$$n_s = \left\lceil \frac{d_{fs} - ov_x}{L_x - ov_x} \right\rceil, \tag{22}$$

where d_{fs} is the distance of the vertex v_{far} from the longest edge e_{scan}. To prevent projected areas of the last strip from including regions out of interest, ov_x and d_s are recomputed as follows:

$$\hat{ov}_x = \frac{n_s L_x - d_{fs}}{n_s - 1}, \tag{23}$$

$$\hat{d}_s = \frac{d_{fs} - L_x}{n_s - 1}. \tag{24}$$

At each step of the back-and-forth path, the number of waypoints is computed by Eq. (19), where d is computed by intersecting a line parallel to e_{scan} with the left and right border of the area, respectively. If the angle between the right(left) border with the respect to the sweep direction is lower(greater) than $\pi/2$, an additional part must be included, similarly to Fig. 14. The distance d can be computed as follows:

$$d = ||v_{left} - v_{right}|| + max\left(0, -\frac{L_x}{\tan(\gamma_{left})}\right) + max\left(0, -\frac{L_x}{\tan(\gamma_{right})}\right). \tag{25}$$

where v_{left} and v_{right} are the two points on the borders and γ_{left} and γ_{right} are the angles of the borders with respect to the sweep direction.

The path generation procedure is summarized in the following algorithm:

Algorithm A:
Input: A set of vertices $\{v_1, \ldots, v_p\}$.
Output: A set of waypoints $\{w_1, \ldots, w_n\}$.

Phase 1:

1. Find the first vertex v_{scan} of the longest edge;

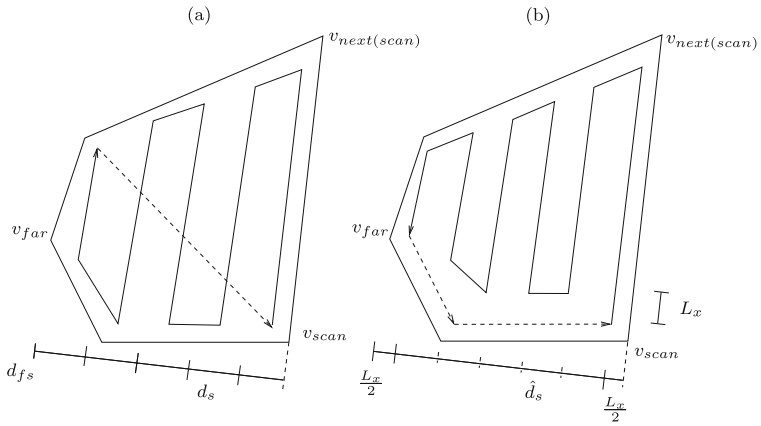

Fig. 16 A path with an odd number of strips (**a**) and a path with an even number of strips, where the return trajectory is exploited for scanning (**b**)

2. Compute the scan direction α_{scan} parallel to the longest edge;
3. for $(i = 1; i < scan; i{+}{+})$
   (a) Compute the distance d from vertex v_i to $v_{next(i)}$;
   (b) Compute n_w by Eq. (19);
   (c) Compute $\hat{ov}_y$ and $\hat{d}_w$ by Eqs. (20) and (21), respectively;
   (d) Place the first waypoint at a distance $(L_y/2, L_x/2)$ from the border and all the other $n_w - 1$ waypoints at a distance $\hat{d}_w$ from each other, and at a distance $L_x/2$ from edge e_i.

Phase 2:

1. Compute the distance d_{fs} of v_{far} from e_{scan};
2. Compute n_s by Eq. (22);
3. Compute $\hat{ov}_x$ and $\hat{d}_s$ by Eqs. (23) and (24), respectively;
4. for $(i = 1; i < n_s; i{+}{+})$
   (a) find v_{left} and v_{right} by intersecting the left and right borders with a line of inclination α_{scan} and distant $(i-1)\cdot\hat{d}_s$ from e_{scan};
   (b) Compute the distance d by Eq. (25);
   (c) Compute n_w and $\hat{d}_w$;
   (d) Place the first waypoint at a distance $(L_y/2, L_x/2)$ from v_{left} (v_{right}) and then all the $n_w - 1$ waypoints at a distance $L_x/2 + (i-1)\cdot\hat{d}_s$ from e_{scan} and a distance $\hat{d}_w$ from each other.

Phase 3:

1. Go from v_{far} to v_{start} through a straight line.

5.2 Improving the path

This section presents an improvements of the path generation algorithm based on the following observation: when the number of stripes is odd, the return path has to pass over a region that has already been scanned, thus it is wasted; whereas when the number of stripes is even, the overall trajectory can be planned so that the return path is also a scanning path. This idea is better illustrated in Fig. 16, which shows a path with an odd number of strips generated by the algorithm presented above (a) and an alternative path containing an even number of strips, where the return path is exploited for scanning (b).

To generate a path like the one shown in Fig. 16b, Algorithm A is modified into Algorithm B as follows:

1. Phase 1 remains unchanged.

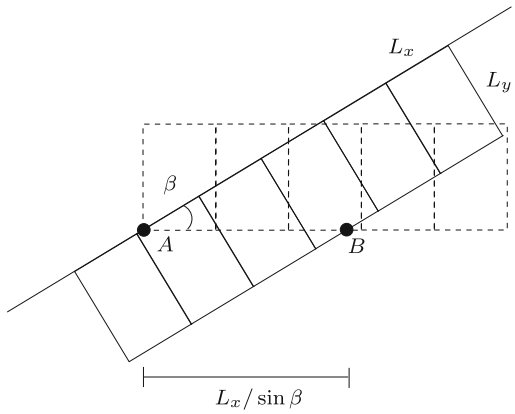

Fig. 17 Example showing that the segment between points A and B can be subtracted from the strip length since covered by the return path

2. In phase 2, a portion of the area is preserved in advance for a return path that goes from v_{far} to v_{start} along the borders. Each strip of the back-and-forth pattern is then reduced by a proper quantity d_r computed as

$$d_r = \frac{L_x}{\sin\beta} \tag{26}$$

where β is the inclination of the boundary side with respect to the sweep direction. This is clarified in Fig. 17 which includes an example of this quantity.
3. In phase 3, instead of going back to v_{start} through a straight line, a set of waypoints is computed along the border from v_{far} to v_{start}.

Algorithm B is is summarized as follows:

Algorithm B:
Input: A set of vertices $\{v_1, \ldots, v_p\}$.
Output: A set of waypoints $\{w_1, \ldots, w_n\}$.

Phase 1: Same as Phase 1 of Algorithm A.
Phase 2:

1. Compute the distance d_{fs} of v_{far} from e_{scan};
2. Compute n_s by Eq. 22;
3. if n_s is odd, then increment it by one;
4. Compute $o\hat{v}_x$ and $\hat{d}_s$ by Eqs. (23) and (24), respectively;
5. for $(i = 1; i < n_s; i{+}{+})$
   (a) find v_{left} and v_{right};
   (b) Compute the distance d and d_r by Eqs. (25) and (26), respectively;
   (c) Compute n_w and $\hat{d}_w$ as a function of $\hat{d} = d - d_r$;
   (d) Place the first waypoint at a distance $(L_y/2, L_x/2)$ from v_{left} (v_{right}) and the other $n_w - 1$ waypoints at a distance $L_x/2 + (i - 1)\cdot\hat{d}_s$ from e_{scan} and a distance $\hat{d}_w$ from each other.

Phase 3:

1. For $(i = far; i < start; i{+}{+})$
   (a) Compute the distance d from v_i to $v_{next(i)}$;
   (b) Compute n_w and $\hat{d}_w$;
   (c) Place the first waypoint at a distance $(L_y/2, L_x/2)$ from v_i and the other $m - 1$ waypoints at a distance $L_x/2$ from e_i and $\hat{d}_w$ from each other.

The following lemma analytically proves that, for certain trapezoidal areas, a path with an even number of stripes is always shorter than a path with an odd number of stripes, in spite of the increased overlap between stripes.

Lemma 1 *Let $\{v_1, v_2, v_3, v_4\}$ be a trapezoidal area where v_1 is start vertex and e_2 is the longest base. If $\gamma_1 \leq \gamma_2$, then the path length generated by Algorithm B is always less than or equal to the path length generated by Algorithm A.*

Proof For the sake of simplicity, the proof is provided assuming $ov_x = ov_y = 0$. If v_1 is the start vertex and e_2 is the longest edge, then we have $v_{start} = v_{scan} = v_1$. The two paths generated by algorithms A and B are illustrated in Fig. 18.

Let λ_A and λ_B be the two path lengths, respectively, and let $n = \lceil l_2/L_x \rceil$, computed by Eq. 22, be an odd number. The two path lengths can be expressed as follows:

$$\begin{cases} \lambda_A = \lambda_A^s + \lambda_A^{left} + \lambda_A^{right} + \lambda_A^{ret} \\ \lambda_B = \lambda_B^s + \lambda_B^{left} + \lambda_B^{right} + \lambda_B^{ret} \end{cases}$$

where λ_X^s denotes the total length of the parallel stripes of the back-and-forth paths generated by Algorithm X, λ_X^{left} is the length of the left part, λ_X^{right} is the length of the right part, and λ_X^{ret} is the length of the return path.

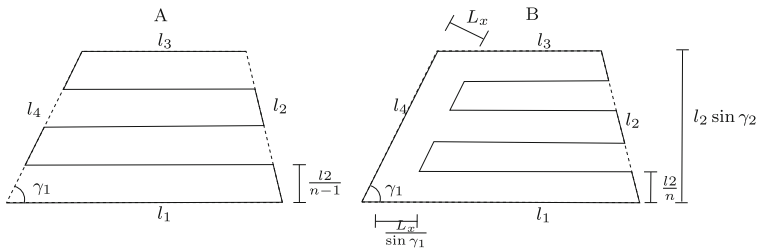

Fig. 18 Paths generated by algorithms A and B for a trapezoidal area

From Fig. 18 it is easy to see that

$$\begin{cases} \lambda_A^s &= (l_1+l_3)\frac{n}{2} \\ \lambda_A^{left} &= \frac{l_4}{2} \\ \lambda_A^{right} &= \frac{l_2}{2} \\ \lambda_A^{ret} &= \sqrt{(l_1-l_2\cos\gamma_2)^2+(l_2\sin\gamma_2)^2} \quad l_{31} \end{cases}$$

$$\begin{cases} \lambda_B^s &= (l_1+l_3)\frac{n+1}{2}-(n-1)\frac{L_x}{\sin\gamma_1} \\ \lambda_B^{left} &= \frac{n-1}{2n}l_4 \\ \lambda_B^{right} &= \frac{n+1}{2n}l_2 \\ \lambda_B^{ret} &= l_4 \end{cases}$$

The lemma holds if $\lambda_B \leq \lambda_A$. Subtracting λ_A from λ_B we get

$$\lambda_B-\lambda_A = \frac{l_1+l_3}{2}-(n-1)\frac{L_x}{\sin\gamma_1}+\frac{l_2}{2n}-\frac{l_4}{2n}+l_4-l_{31}.$$

If $a = l_2\sin\gamma_2 = l_4\sin\gamma_1$ is the altitude of the trapezoid, we can observe that $(n-1)L_x \geq a$, because after the correction done by Eq. (24), a is divided into $n-1$ pieces smaller that L_x. Hence we have that

$$(n-1)\frac{L_x}{\sin\gamma_1} \geq \frac{a}{\sin\gamma_1} = l_4.$$

Hence, we can write

$$\lambda_B-\lambda_A \leq \frac{l_1+l_3}{2}+\frac{l_2-l_4}{2n}-l_{31}.$$

Being $\gamma_1 \leq \gamma_2$, we have that $l_2 \leq l_4$, thus $\frac{l_2-l_4}{2n} \leq 0$, hence we can write

$$\lambda_B-\lambda_A \leq \frac{l_1+l_3}{2}-l_{31}.$$

And since

$$l_3 = l_1-l_2\cos\gamma_2-l_4\cos\gamma_1 \leq l_1-2l_2\cos\gamma_2$$

we can write

$$\lambda_B-\lambda_A \leq \frac{l_1+l_3}{2}-l_{31} \leq l_1-l_2\cos\gamma_2-l_{31},$$

that is,

$$\lambda_B-\lambda_A \leq (l_1-l_2\cos\gamma_2) -\sqrt{(l_1-l_2\cos\gamma_2)^2+(l_2\sin\gamma_2)^2} \leq 0.$$

Hence the lemma follows. □

Note that, provided that $\gamma_1 \leq \gamma_2$, Lemma 1 is also valid for $\gamma_2 > \pi/2$, since the quantity l_{31} increases for $\gamma_2 > \pi/2$. The following lemma shows that, when $\gamma_2 < \gamma_1$, there is a critical angle γ_2^* which determines whether path B is shorter than path A, or viceversa.

6 Providing safety guarantees

This section presents two different mechanisms that are helpful for providing safety guarantees during the UAV flight. First, an off-line feasibility test is proposed to check that the energy stored in the battery is sufficient to perform the planned path. Second, an online fail-safe mechanism is implemented on the ground station to continuously check that the actual available energy is always greater than the energy required to bring the drone back to the starting point. This mechanism prevents the drone to go to waypoints that are too far away from the starting point, always ensuring a safe return path.

6.1 Offline feasibility test

The total energy needed to cover the entire path can be computed as follows:

$$\begin{aligned} E_{path} &= E_{climb}(0,h_{max}) + E_{desc}(h_{max},0) \\ &+ n_t E_{turn} + \sum_i (E_{acc}(0,v_i^*) + E_v(d_i,v_i^*) \\ &+ E_{dec}(0,v_i^*)) \end{aligned} \tag{27}$$

where d_i is the distance between the extreme waypoints of segment i, v_i^* is the optimal speed computed for segment i, and n_t is the total number of turns in the path. Given the characteristics of a LiPo battery, the total available energy can be computed as:

$$E_{tot} = V_n \cdot I_H \cdot 3600 \cdot P\% \tag{28}$$

where $P\%$ is the maximum percentage of energy that can be used (typically 70%) to avoid damaging the UAV or the battery.

Then, the feasibility test can be simply done by checking if $E_{path} < E_{tot}$. If the feasibility test is passed, the remaining energy ($E_{tot} - E_{path}$) can be used to increase the spatial resolution of the acquired images. This can be done by iteratively reducing the flight altitude, recomputing the path, and re-running the feasibility test, until an altitude h is found such that $E_{path} = E_{tot} - \epsilon$, where ϵ is a given tolerance. If the feasibility test is not passed, the path has to be redesigned considering multiple flights or multiple UAVs. Note that energy can be traded with the spatial resolution, and the same iterative procedure can be applied to reduce the total required energy by increasing the altitude. However, finding the optimal altitude that minimizes E_{path} is highly complex due to

the non linearity of the problem: increasing/decreasing the height will change the energy consumption not monotonically.

6.2 Online battery failsafe mechanism

There are several cases in which a UAV is not able to complete the entire scheduled path, even when the off-line feasibility test is passed. This is due to the fact that the battery could drains more than expected, or that it was not completely charged. A set of failsafe operations exists to handle such exceptions, not only for issues related to the battery, but also for the radio, GPS, and Ground Station. The current existing battery failsafe mechanism is triggered when the battery voltage crosses a particular threshold. When the failsafe is triggered is it possible to execute specific recovery actions, such as a (RTL) or LAND. In the current mechanism, however, the position of the UAV in proximity of the failsafe is not taken into account, hence it may be too far from the starting point to come back safely. Figure 19 shows a typical discharging voltage curve of a 3S battery, highlighting two standard voltage failsafe thresholds. The first failsafe triggers an (RTL) operation when the voltage battery goes below a threshold $V_{rtl} = 10.4$ V for more than 10 seconds. A second failsafe may force the drone to land after a second threshold $V_{land} = 9V$ is crossed, even if it is still far from the starting point. A more efficient failsafe could estimate the energy required to come back to the starting point at every instant and trigger a safe return-to-launch action, thus preventing the drone to land in areas difficult to reach. It is worth noting that, an accurate failsafe is needed to prevent accidental fall that may cause damage to the UAV and the objects underneath. Moreover, even if the UAV lands safely (in a wrong location) an exhausting overuse of the battery may damage it completely.

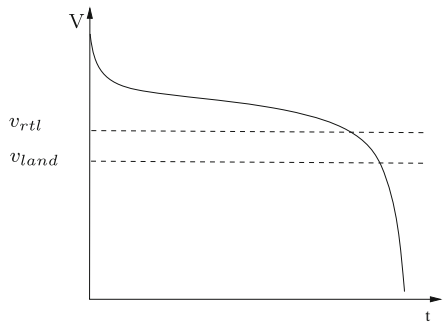

Fig. 19 Example of discharging voltage curve of a 3S liPo battery during a flight

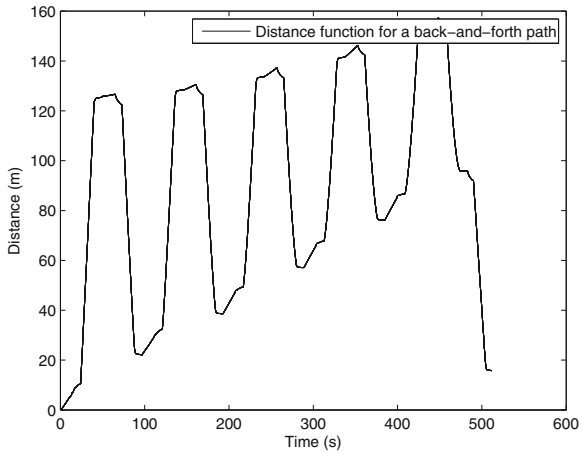

Fig. 20 Distance function for the path shown in Fig. 21

In this section, we introduce two functions, called Distance and Energy function (D-Function and E-Function) that are needed to implement the behavior of the online-failsafe mechanism. The D-function represents the distance of the UAV from the starting point at every time instant. This function can be computed off-line by knowing the UAV dynamics, or online as

$$D(t) = ||x(t) - x(0)|| \tag{29}$$

where $x(t)$ is the current drone position and $x(0)$ the starting location. Figure 20 shows the D-Function corresponding to the path showed in Fig. 21.

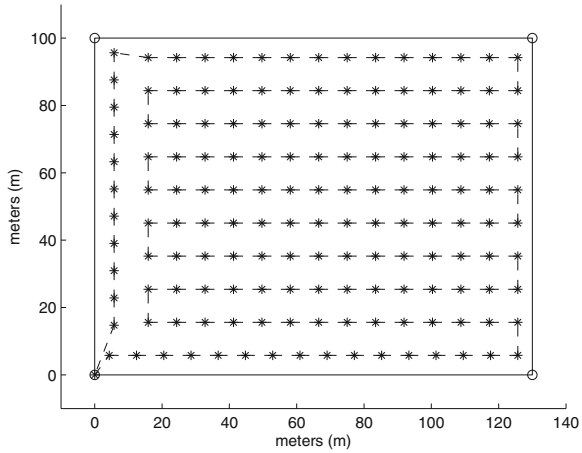

Fig. 21 Sample path used for computing the D-function reported in Fig. 20

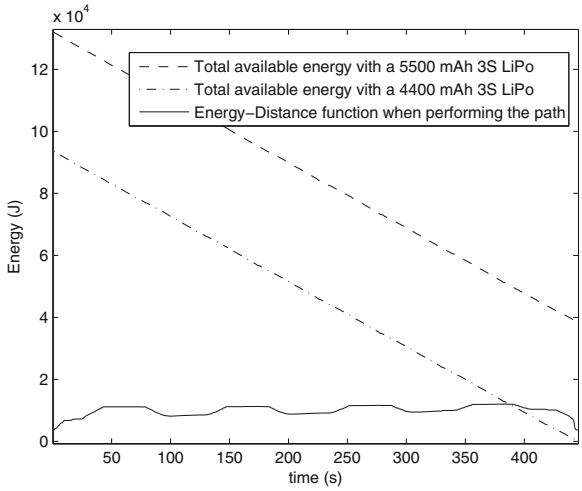

Fig. 22 E-function (*solid line*) of the D-function showed in Fig. 20. The two descending curves represent the estimated consumed energy during the path with two different batteries

The E-fucntion represents the energy required by the UAV to come back from its actual position to the starting point at every instant. The E-function is computed by substituting the D-Function $D(t)$ in Eq. 16 and summing the energy required to descend to the ground:

$$E^D(v^*(D(t)), D(t)) = E_a(v^*(D(t)), D(t)) + E_{desc}(h) \tag{30}$$

where $v^*(D(t))$ is the optimal speed to travel $D(t)$ and h is the actual height at which the drone is flying. The actual energy $E_{curr}(t)$ stored in the battery during a flight can be expressed as a descending curve where $E_{curr}(0) = E_{tot}$. $E_{curr}(t)$ can be estimated at every instant t by using the energy model and the planned path. Then, it is possible to compare the E-function (along the entire path) with $E_{curr}(t)$. If $E_{curr}(t)$ is greater than the E^D at every instant, then the entire path can be traveled safely. Figure 22 shows the E-function for the path represented in Fig. 21 and the estimated available energy $E_{curr}(t)$ at every instant for two batteries with different maximum capacity. It is worth noting that this is a graphical representation of the off-line feasibility test.

As stated above the feasibility test is not enough to provide safe guarantees. If, at a particular instant, $E_{curr}(t)$ intersects E^D the UAV has to return immediately at the starting point with a return-to-launch operation. We implemented this energy failsafe mechanism on the ground station: the UAV sends every second status messages about its current state, including GPS data and a measure of the voltage and the current. $E_{curr}(t)$ is estimated by integrating these two measurements. The distance is computed with the GPS information and E^D by a look-up table that stores the required energy and the optimal speed for every distance d. At every second, condition $E_{curr}(t) \leq E^D$ is checked to trigger a return-to-launch action.

7 Experimental Validation

The proposed algorithm has been used to reconstruct a geographical area of agricultural interest to study the growth of vegetation over time. The survey area with the planned path is showed in Fig. 23, while the

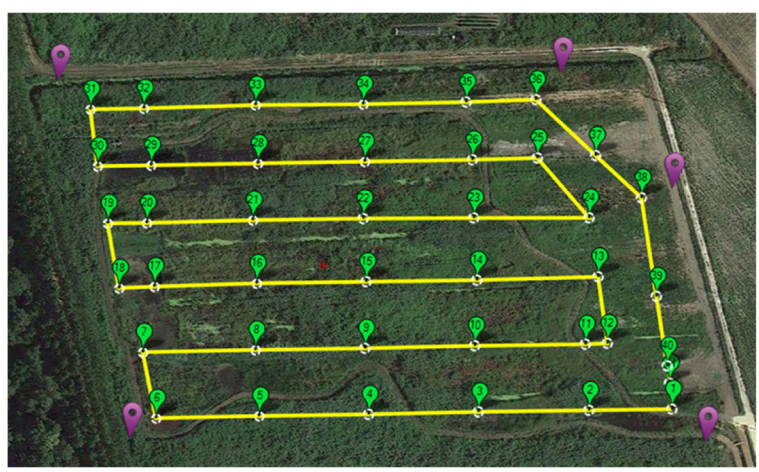

Fig. 23 The picture shows a set of waypoints that cover an area of interest

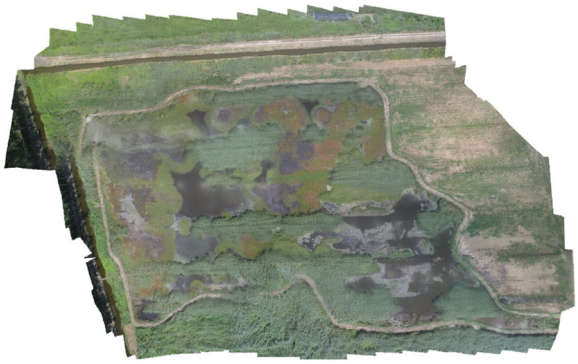

Fig. 24 The picture shows the resulting ortho-mosaic of the area covered with the waypoints of Fig. 23

corresponding ortho-mosaic generated image is shown in Fig. 24. This section presents two specific experiments aimed at validating the proposed approach. The first experiment was performed to compare Algorithm A and B, whereas the second one was carried out to test the proposed online failsafe mechanism. Both experiments were performed using an IRIS quadcopter equipped with a 5500 mAh 3S LiPo battery and all the measurements were logged in an SD card.

7.1 Comparing Algorithm A and B

In the first experiment, two flights were performed to compare the two algorithms. The area of interest is a rectangular area of dimension 150 m x 100 m., the camera has an AOV $\alpha = 94.4$ deg, $T_s = 1\ s$, $T_e = 1/580\ s$, and the required resolution is $R_d = 1.6\ pixel/cm$. The maximum altitude computed by Eq. 4 results to be $h_{max} = 11.6$ m, hence we set the $h = 11$ m, and the overlap is $ov_y = ov_y = 0\ m$. The fastest speed of the quadcopter is $v = 15.5\ m/s$, however Eq. (7) impose a bound on the maximum speed, that is $v_{max} = \min(38.1, 14.73) = 14.37\ m/s$. In this experiment v_{max} resulted greater than the optimal speed v^*, hence the constraint expressed by Eq. (7) is not effective.

A first flight was done by creating the set of waypoints using Algorithm A, choosing an area resulting in a odd number of stripes ($n_s = 5$), while a second flight was done on the path produced by Algorithm B on an even number of stripes ($n_s = 6$). Table 2 shows the total traveled distance, the total amount of consumed energy, the amount of time taken to complete the paths, and the number of turns for the two flights.

Table 2 Parameters related to the first experiment

Variable	n_s odd	n_s even
Total distance	926.25 m	896.1 m
Horizontal Overlap	11.3%	29.1%
Total energy	$7.83 \cdot 10^4$ J	$8.81 \cdot 10^4$ J
Total time	284 s	316 s
Number of stripes	5	6
Number of turns	9	11

As predicted by Lemma 1, the path generated by Algorithm B resulted in a shorter traveled distance, increasing the image overlap by 30%. However, since Algorithm B always produces two more turns than Algorithm A, the total time and energy required by Algorithm B resulted to be higher than the corresponding time and energy required by Algorithm A. Note that the energy increase is more significant when the flight has a short duration and the number of stripes is low, because the energy needed to perform two more turns is comparable or even greater than the energy saved on the total distance. Also note that such an energy increase is a phenomenon that is specifically related to quadcopters, where the energy consumed during turns and decelerations is quite high. In other vehicles characterised by a different energy models, such as wheeled robots, the observed phenomenon does not appear, since the energy consumed to perform a turn is much lower than the energy used to travel a distance.

In general, since none of the two algorithms are able to minimize the energy consumption in all possible situations, the user could run both of them and select the one that leads to the least energy consumption.

7.2 Experiment on the Energy failsafe

A second experiment was performed on the path showed in Fig. 21 to test the online failsafe mechanism described in Section 6.2. First, a feasibility test was successfully performed on a 3S 5500 mAh LiPo battery. However, in order to test the energy-failsafe mechanism, a battery with less capacity (4400 mAh) was used in the experiment. A previous estimation of $E_{curr}(t)$ (Fig. 22) showed that with this battery the energy expired before concluding the path. Figure 25

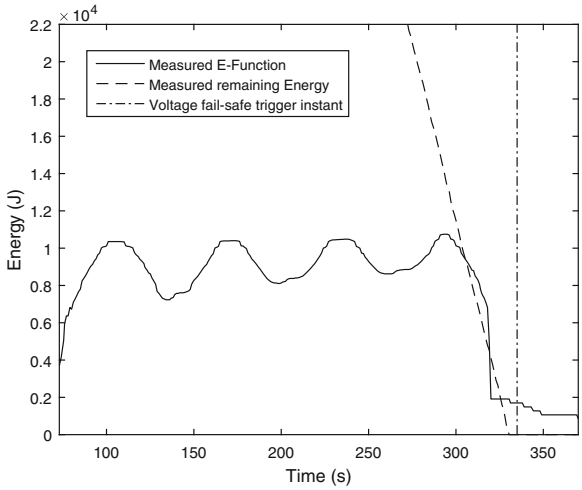

Fig. 25 The E-function (*solid line*) measured during the flight of the second experiment. The remaining Energy (*dashed line*) intersects the E-Function triggering a Return-to-Launch. The vertical dashed and dotted line represents the instant in which the voltage fail-safe triggers an RTL

shows the E-function and the remaining energy measured by the ground station. During the actual flight, $E_{curr}(t)$ (dashed line) intersects the E-function (solid line) when the UAV was finishing the 7^{th} strip (as estimated by the model) and a return-to-launch operation was triggered.

We observed that the classical voltage fail-safe threshold (vertical dashed-and-dotted line in Fig. 25) was passed when the UAV was already in the descending phase, confirming the timely activation of the proposed energy failsafe online mechanism.

In summary, this experiment showed two important results: first, it further validated the energy model, since the intersection between the E-function and the remaining energy was correctly estimated; second, the energy failsafe mechanism was triggered at the correct time instant, permitting a safe return to home without landing in unreachable areas and also preventing the battery to over discharge.

8 Conclusions

This paper presented a coverage path planning algorithm able to account for energy and resolution constraints specified on the mission. The paper described a method for deriving an energy model of a specific UAV starting from real measurements and using it to compute the speed that minimizes energy consumption along straight paths of a given length. Once the full path is generated, the proposed algorithm was used to derive the speed that minimizes the energy consumption for each segment of the path. Then, a feasibility test has been proposed to verify whether the energy available on the UAV is sufficient to scan the entire area. Finally, the proposed energy model has been also used to introduce an additional fail-safe mechanism triggered when the residual energy is just sufficient to bring the UAV back to the initial point.

As a future work we plan to derive the results proved for a trapezoidal area to generic convex areas and extend the proposed approach to multiple drones that have to be coordinated to fulfill the mission on larger areas or in shorter times.

References

1. Grenzdörffer, G., Engel, A., Teichert, B.: The photogrammetric potential of low-cost UAVs in forestry and agriculture. Int. Arch. Photogram. Rem. Sens. Spatial Inform. Sci. **31**(B3), 1207–1214 (2008)
2. Zarco-Tejada, P.J., Berni, J.A., Suárez, L., Fereres, E.: A new era in remote sensing of crops with unmanned robots. SPIE Newsroom, pp. 2–4 (2008)
3. Kazmi, W., Bisgaard, M., Garcia-Ruiz, F., Hansen, K.D., la Cour-Harbo, A.: Adaptive surveying and early treatment of crops with a team of autonomous vehicles. In: European Conference on Mobile Robots, pp. 253–258 (2011)
4. Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., Siegwart, R.: A UAV system for inspection of industrial facilities. In: Aerospace Conference 2013 IEEE, pp. 1–8. IEEE (2013)
5. Casbeer, D.W., Kingston, D.B., Beard, R.W., McLain, T.W.: Cooperative forest fire surveillance using a team of small unmanned air vehicles. Int. J. Syst. Sci. **37**(6), 351–360 (2006)
6. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
7. Maza, I., Ollero, A.: Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In: Distributed Autonomous Robotic Systems 6, pp. 221–230. Springer (2007)
8. Barrientos, A., Colorado, J., Cerro, J.d., Martinez, A., Rossi, C., Sanz, D., Valente, J.: Aerial remote sensing in agriculture: a practical approach to area coverage and path planning for fleets of mini aerial robots. J. Field Robot. **28**(5), 667–689 (2011)
9. Öst, G.: Search Path Generation with UAV Applications Using Approximate Convex Decomposition, Masters Thesis. Linköpings universitet, Sweden (2012)

10. Santamaria, E., Segor, F., Tchouchenkov, I., Schoenbein, R.: Rapid aerial mapping with multiple heterogeneous unmanned vehicles. International Journal On Advances in Systems and Measurements **6**(3 and 4), 384–393 (2013)
11. Bast, H., Hert, S.: The area partitioning problem. In: 12th Canadian Conference on Computational Geometry (2000)
12. Huang, W.H.: Optimal line-sweep-based decompositions for coverage algorithms. In: International Conference on Robotics and Automation (ICRA), vol. 1, pp. 27–32. IEEE (2001)
13. Lawrance, N., Sukkarieh, S.: Wind Energy Based Path Planning for a Small Gliding Unmanned Aerial Vehicle. In: AIAA Guidance Navigation and Controls Conference, American Institute of Aeronautics and Astronautics (2009)
14. Al-Sabban, H.W., Gonzalez, L.F., Smith, R.N.: Wind-energy based path planning for unmanned aerial vehicles using markov decision processes. In: International Conference on Robotics and Automation (ICRA), pp. 784–789. IEEE (2013)
15. Roberts, J.F., Zufferey, J.-C., Floreano, D.: Energy management for indoor hovering robots. In: International conference on Intelligent Robots and Systems (IROS), pp. 1242–1247. IEEE (2008)
16. Mei, Y., Lu, Y.-H., Hu, Y.C., Lee, C.G.: Deployment of mobile robots with energy and timing constraints. IEEE Trans. Robot. **22**(3), 507–522 (2006)
17. Di Franco, C., Buttazzo, G.: Energy-aware coverage path planning of UAVs. In: Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC 2015), (Vila Real, Portugal). IEEE, April 8–10 (2015)

Carmelo Di Franco is currently a Ph.D. student in embedded computing systems at the ReTiS Lab of the Scuola Superiore Sant'Anna, Pisa. He graduated cum Laude in 2013 in computer engineering at the University of Pisa. His research focuses on localization and navigation algorithms for mobile robots. In particular, he works on multi-sensor data fusion, distance-based relative localization, and energy-efficient coverage path planning of UAVs. He is involved in a project in which fleets of drones are used to monitor the growth of plants in farms to find plant infections and prevent bad crops.

Giorgio Buttazzo is full professor of computer engineering at the Scuola Superiore Sant'Anna of Pisa. He graduated in electronic engineering at the University of Pisa in 1985, received a M.S. degree in computer science at the University of Pennsylvania in 1987, and a Ph.D. in computer engineering at the Scuola Superiore Sant'Anna of Pisa in 1991. From 1987 to 1988, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory of the University of Pennsylvania, Philadelphia. He has been Program Chair and General Chair of the major international conferences on real-time systems and Chair of the IEEE Technical Committee on Real-Time Systems. He is Editor-in-Chief of Real-Time Systems, Associate Editor of the IEEE Transactions on Industrial Informatics, and IEEE Fellow since 2012. He has authored 7 books on real-time systems and over 200 papers in the field of real-time systems, robotics, and neural networks.