Energy-Aware Dynamic Planning Algorithm for Autonomous UAVs

Adam Seewald[1], Hector Garcia de Marina[2], and Ulrik Pagh Schultz[1]

***Abstract*— abstract**
abstract
abstract
abstract
abstract
abstract
abstract
abstract
abstract
abstract
abstract
abstract
abstract

I. Introduction

Many scenarios involving unmanned aerial vehicles (UAVs), such as precision agriculture, search and rescue, and surveillance, require high autonomy but have limited energy budgets. A typical example of these scenarios is a UAV flying a path and performing some onboard computational tasks. For instance, the UAV might detect ground patterns and notify other ground-based actors with little human interaction. We refer to such computational tasks that can be dynamically replanned and adapted as *computations*. We are interested in the energy optimization of the path and computations under uncertainty (atmospheric interferences) and refer to it as energy-aware dynamic planning. Such planning would find optimal tradeoffs between the path, computations, and energy requirements. Current generic planning solutions for outdoor UAVs do not plan the path and computations dynamically, nor are they energy-aware. They are often semi-autonomous: the path and computations are static and usually defined using planning software [1] (for instance [2] and [3]). Such a state of practice has prompted us to propose an *energy-aware dynamic planning algorithm* for UAVs. The algorithm combines and generalizes some of the past body of knowledge on mobile robot planning problems and addresses the increasing *computational demands* and their relation to energy consumption, path, and autonomy for the UAV planning problem.

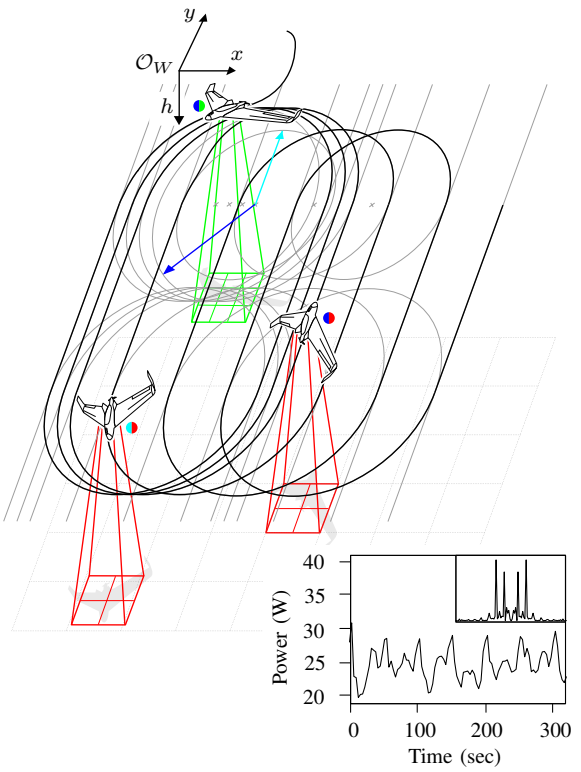


Fig. 1. An initial plan of an agricultural scenario replanned dynamically in terms of both the path (travels a more accurate survey than) and computations (elaborates more images per second than). In bottom right, the collected energy data of the UAV flying the scenario, along the power spectrum (small rectangle) where the first frequency has been filtered.

Planning algorithms literature for mobile robots includes topics such as trajectory generation and path planning. Generally, the algorithms select an energy-optimized trajectory [4], e.g., by maximizing the operational time [5]. However, they apply to a small number of robots [6] and focus exclusively on planning the trajectory [7], despite compelling evidence for the energy consumption also being significantly influenced by computations [8]. Given the availability of powerful GPU-equipped mobile hardware [9], the use of computations is expected to increase in the near future [10]–[12]. More complex planning, which includes a broader concept of the plan being a set of tasks and a path, all focus on

This work is supported and partly funded by the European Union's Horizon2020 research and innovation program under grant agreement No. 779882 (TeamPlay).

[1]Adam Seewald, Ulrik Pagh Schultz are with the SDU UAS Center, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, Denmark. Email: ads@mmmi.sdu.dk.

[2]Hector Garcia de Marina is with the Faculty of Physics, Department of Computer Architecture and Automatic Control, Universidad Computense de Madrid, Spain.

the trajectory [8], [13] and apply to a small number of robots [14], [15]. For UAVs specifically, rotorcrafts have also gained interest in terms of algorithms for energy-optimized trajectory generation [16], [17].

Unlike most of the past planning algorithms literature, our algorithm plans both the path and computations. We present here a brief summary of the algorithm. To model the path we use multiple trajectories and denote each of them with a mathematical function. In Figure 1, the path contains multiple circles and lines. To model the computations we use a profiling tool presented in previous work [18]. To guide the UAV we use a vector field [19] that converges smoothly to the planned trajectory. The use of vector fields for planning is widely discussed in the literature [19]–[24].

To achieve the energy-aware dynamic planning, we further introduce and formally proof a periodic energy model that accounts for the uncertainty. We use Fourier analysis to derive the model, and state estimation to address the uncertainty. Periodicity is often present due to repetitive patterns in the plan [25]. Indeed, UAV scenarios often iterate over a set of tasks and trajectories (e.g., monitoring or search and rescue). Given that the plan is periodic, we expect the energy consumption to approximately evolve periodically. The observation applies broadly; we show different UAV scenarios that can produce such periodic energy consumption evolution. Moreover, we briefly investigate periodic energy consumption evolution in some ground-based mobile robots.

In the spirit of reducing costs and resources, we showcase the algorithm using the problem of dynamic planning for a precision agriculture fixed-wing UAV. Precision agriculture is often put into practice [26] with ground mobile robots used for harvesting [27]–[32], and UAVs for preventing damage and ensuring better crop quality [1], [33]. The plan is structured as follows. Trajectory-wise, the UAV flies in circles and lines covering a polygon. Computationally-wise, it detects obstacles using a convolutional neural network (CNN) and notifies grounded mobile robots employed for harvesting. The algorithm alters the plan; it controls the processing rate and the radius of the circles (affecting the distance between the lines). Figure 1 shows a slice of such and the small plot shows the periodicity from the collected energy data. The small rectangle shows the spectrum power with the first frequency filtered, showing thus that three frequencies are mostly necessary to model the energy. Computations significantly impact the battery, with a potential extension of up to 13 minutes over an hour by switching from the highest to the lowest level of computations (see Section V).

The remainder of the paper is organized as follows. The overview of dynamic planning is provided in Section II, and the problem is set with some preliminaries in Section II-A. A suitable model for the energy in Section III. The algorithm is proposed in Section IV. Section V presents the result and showcases the performances. We propose some conclusions in Section VI. Finally, we provide some additional information and examples in Appendixes I–III.

II. Planning Overview

The algorithm plans the path and computations from a user-specified initial plan that consists of different stages and some initial guesses that are necessary for the position and energy estimation. At each stage, the UAV must follow a trajectory within the trajectory constraints set and do the computations within the computations constraints sets. There is one trajectory constraints set but multiple computations constraints sets, one per each computation parameter. The user specifies the maximum, minimum, and current levels of some parameters relative to the path and computations in such plan with an energy budget. In Figure 1, there are two parameters. The radius of the trajectory circle is relative to the trajectory (and). The quality of image detection is relative to the computations (and). The algorithm outputs the parameters (the control) using output model predictive control (MPC) [34]. The UAV utilizes the control to alter its energy consumption to meet the energy budget. The control is data-driven. Energy sensor data estimates coefficients of the energy model used to predict future energy consumption and derive the control, replanning the initial plan in presence of uncertainty. The energy budget is the battery capacity and some other battery parameters. Our goal is to complete the plan with the highest possible parameters as the UAV flies and its batteries drain. The term parameters should not be confused with the battery parameters, which we always mention explicitly. These are fixed values that are not replanned by the algorithm.

A. Preliminaries and Problem Definition

Let us assume that the trajectory at stage i can be altered with ρ trajectory parameters and the computations with σ computations parameters. We then express the trajectory as a continuous twice differentiable function $\varphi_i : \mathbb{R}^2 \times \mathbb{R}^\rho \to \mathbb{R}$ of a point and the trajectory parameters. The function returns a metric of the distance between the point and the nominal trajectory. We express the computations as the value of the computations parameters. We discuss the concrete meaning of the value of trajectory parameters in Subsection III-A, and computations parameters in Subsection III-B.

Let us adopt the following mathematical notation. Given an integer a, $[a]$ is the set $\{0, 1, \ldots, a\}$, $[a]^+$ the

set $[a]/\{0\}$. $c_{i,j}$ specifies the j-th parameter of the i-th parameters set c_i. $c_{i,j}^0$ is the optimal value of the parameter $c_{i,j}$ w.r.t. a given cost function l, and $\underline{c}_{i,j}, \overline{c}_{i,j}$ its lower and upper bounds. The set $\langle c_{i,1}, c_{i,2}, ..., c_{i,n}\rangle$ is an ordered list of n parameters.

Definition II.1 (Stage, plan, triggering, and final point)**.** A *stage* Γ_i at time instant k of a plan Γ is defined as the ordered list

$$\begin{aligned}\Gamma_i := \{&\langle \varphi_i(\mathbf{p}_k, c_{i,1}, \ldots, c_{i,\rho}), c_{i,\rho+1}, \ldots, c_{i,\rho+\sigma}\rangle \\ &| \exists\ \mathbf{p}_k,\ \varphi_i(\mathbf{p}_k, c_{i,1}, \ldots, c_{i,\rho}) \in \mathbb{C}_i, \\ &\forall j \in [\sigma]^+,\ c_{i,\rho+j} \in \mathbb{S}_{i,j}\ \},\end{aligned} \tag{1}$$

where $\mathbb{C}_i := [\underline{c}_i, \overline{c}_i] \subseteq \mathbb{R}$ is the trajectory constraints set, and $\mathbb{S}_{i,j} := [\underline{c}_{i,\rho+j}, \overline{c}_{i,\rho+j}] \subseteq \mathbb{Z}_{\geq 0}$ the j-th computation constraints set. $\mathbf{p}_k$ is a point of a UAV flying at an assigned altitude $h \in \mathbb{R}_{>0}$ w.r.t. some inertial navigation frame $\mathcal{O}_W$.

The *plan* is a finite state machine Γ where the state-transition function $\delta : \bigcup_i \Gamma_i \times \mathbb{R}^2 \to \bigcup_i \Gamma_i$ maps a stage and a point to the next stage

$$\delta(\Gamma_i, \mathbf{p}_k) := \begin{cases} \Gamma_{i+1} & \text{if } \mathbf{p}_k = \mathbf{p}_{\Gamma_i} \\ \Gamma_i & \text{otherwise} \end{cases}.$$

The point $\mathbf{p}_{\Gamma_i}$ that allows the transition between Γ_i and Γ_{i+1} is called *triggering point*. The last triggering point $\mathbf{p}_{\Gamma_l}$ relative to last stage Γ_l is called *final point*.

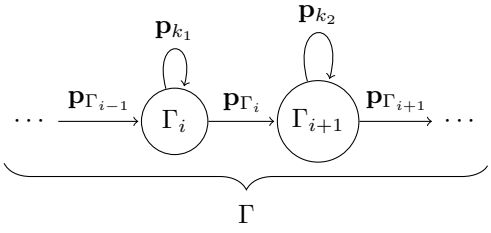


Fig. 2. The plan defined as a finite state machine

A slice of the plan in Figure 2 shows the transition between the stages. The triggering point $\mathbf{p}_{\Gamma_{i-1}}$ allows the transition to stage Γ_i. The UAV remains in the stage with any generic point $\mathbf{p}_{k_1}$ and $\mathbf{p}_{k_2}$. It eventually enters the stage Γ_{i+1} with the triggering point $\mathbf{p}_{\Gamma_i}$ and so on, until it reaches the last triggering point.

Generally, one can express the triggering points in function of the i-th trajectory parameters $c_{i,1}, \ldots, c_{i,\rho}$, or any previous trajectory parameters, propagating the information therein if necessary. See Figure 3 where, for simplicity, we neglected the $\sigma = 0$ computations parameters, so $c_i = c_{i,1}, \ldots, c_{i,\rho}$.

We refer the reader to an example in Appendix II-B for a detailed implementation. In the example, the first trajectory parameter is propagated to all the following trajectories and triggering points.

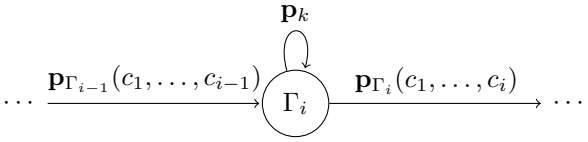


Fig. 3. Detail of the stage Γ_i in the finite state machine

We store the plan in the plan specification, the format is described in Appendix II-A.

In order to formulate the problem, we considerate some assumptions.

Assumption II.1 (Plan periodicity)**.** Given an initial plan Γ consisting of l stages, a generic starting point $\mathbf{p}$ and current levels of parameters, there exists a constant $n \in \mathbb{Z}_{>0}, n < l$ such that

$$\varphi_i - \varphi_{n+i} = d_i,\ \ \forall i \in [l]^+$$

where $d_1, \ldots, d_l$ are arbitrary constant differences.

Physically, this means that the plan is periodic.

Definition II.2 (Period)**.** The period $T \in \mathbb{R}_{>0}$ is defined as the time between φ_i and φ_{n+i} for an arbitrary $i \in [l]^+$.

The algorithm measures the time between the stages Γ_i and Γ_{n+i} and assumes the initial period is one.

From Assumption II.1, follows the next proposition.

Proposition II.2 (Rank of the trajectory functions)**.** Suppose Assumption II.1 holds. Then

$$\operatorname{rank} \begin{bmatrix} \varphi_1 & \varphi_{n+1} & \cdots & \varphi_{(l-n)+1} \\ \varphi_2 & \varphi_{n+2} & \cdots & \varphi_{(l-n)+2} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_n & \varphi_{2n} & \cdots & \varphi_l \end{bmatrix} = 2.$$

Proof. The proof follows directly from Assumption II.1. The column rank of the matrix in the lemma is the dimension of its column space. It denotes the size of the set of all the possible linear combination of the column vectors. We note that there is a limited number (two) of linear combinations of the column vectors being these linearly dependent (the values of the constant differences $d_1, \ldots, d_l$ are the same for each i). ■

The lemma shows an easy way to verify that a plan is periodic.

Problem II.1 (UAV planning problem)**.** Consider an initial plan Γ defined in Definition II.1 satisfying Assumption II.1 and Proposition II.2.

We are interested in the planning of the path parameters $c_{i,1}, \ldots, c_{i,\rho}$ and computations parameters $c_{i,\rho+1}, \ldots, c_{i,\rho+\sigma}$ and to derive the path resulting from such plan for the UAV planning problem.

III. Periodic Energy Model

We model the energy using some energy coefficients $\mathbf{q} \in \mathbb{R}^m$ that characterize the energy signal. We refer to the instantaneous energy consumption evolution simply as the energy signal. The coefficients are derived from Fourier analysis (the size of the energy coefficients vector m is related to the order of a Fourier series) and estimated using a state estimator (Kalman filter).

We proof a relation between the energy signal and the energy coefficients in Theorem IV.1. We show after the main results how this approach allows us variability in terms of the systems behaving periodically, piece-wise periodically, or merely linearly with sporadic periodicity.

Once we illustrate the energy model, we enhance it with the energy contribution of the computations in Subsection III-B.

Let us consider a periodic signal of period T, and a Fourier series of an arbitrary order $r \in \mathbb{Z}_{\geq 0}$ for the purpose of modeling of the signal

$$h(t) = a_0/T + (2/T)\sum_{j=1}^{r}(a_j \cos \omega j t + b_j \sin \omega j t), \quad (2)$$

where $h : \mathbb{R}_{\geq 0} \to \mathbb{R}$ maps time to the instantaneous energy consumption, $\omega := 2\pi/T$ is the angular frequency, and $a, b \in \mathbb{R}$ the Fourier series coefficients.

The energy signal can be modeled by Equation 2 and the output of a linear model

$$\begin{aligned} \dot{\mathbf{q}} &= A\mathbf{q} + B\mathbf{u}, \\ y &= C\mathbf{q}, \end{aligned} \quad (3)$$

where $y \in \mathbb{R}$ is the instantaneous energy consumption.

The state $\mathbf{q}$ are the energy coefficients

$$\begin{aligned} \mathbf{q} &= \begin{bmatrix} \alpha_0 & \alpha_1 & \beta_1 & \cdots & \alpha_r & \beta_r \end{bmatrix}^T, \\ A &= \begin{bmatrix} 0 & & & & \\ & A_1 & & & \\ & & \ddots & & \\ & & & A_r \end{bmatrix}, A_j \begin{bmatrix} 0 & \omega j \\ -\omega j & 0 \end{bmatrix}, \\ C &= (1/T) \begin{bmatrix} 1 & 1 & 0 & \cdots & 1 & 0 \end{bmatrix}, \end{aligned} \quad (4)$$

where $\mathbf{q} \in \mathbb{R}^m$ with $m = 2r+1$, $A \in \mathbb{R}^{m \times m}$ is the state transmission matrix, and $C \in \mathbb{R}^m$ is the output matrix. In matrix A, the top left entry is zero, the diagonal entries are $A_1, \ldots, A_r$, the remaining entries are zeros.

Linear model in Equation 3 allows us to include the control in the model of Equation 2. The algorithm uses the model to find the control sequence that results in the optimal parameters configuration (using MPC).

Lemma III.1 (Signals equality)**.** Suppose control is zero, matrices A, C are described by Equation (4), and the initial guess $\mathbf{q}_0$ is

$$\mathbf{q}_0 = \begin{bmatrix} a_0 & a_1/2 & b_1/2 & \cdots & a_r/2 & b_r/2 \end{bmatrix}^T.$$

Then, the signal h in Equation (2) is equal to the signal y in Equation (3).

Proof. The equivalence of the models is trivial and the equality of the two signals is achieved by a proper choice of items of matrices A, C and the initial guess $\mathbf{q}_0$. We refer the reader to Appendix I for a formal proof of the equality where we justify the choices of the items of the matrices and of the initial guess. ■

Let us consider the discretized version $A_d := A\Delta t + I$ of the matrix A. We denote the continuous and discrete time with the standard notation t, k. We use the forward Euler approximation with a small enough value of the time step Δt. We note that one can guarantee to have the same outputs rather than an approximation with $A_d := e^{At}$.

Let us suppose that at time instant k the plan reached the i-th stage Γ_i.

The control along with the input matrix

$$\begin{aligned} \mathbf{u} &= \begin{bmatrix} c_{i,1} & \cdots & c_{i,\rho} & c_{i,\rho+1} & \cdots & c_{i,\rho+\sigma} \end{bmatrix}^T, \\ B\mathbf{u} &:= B_i(\mathbf{u}_k - \mathbf{u}_{k-1}), \ B_i = \begin{bmatrix} \nu_i \\ \mathbf{0} \end{bmatrix}, \end{aligned} \quad (5)$$

where $\mathbf{u} \in \mathbb{R}^n$ is the control with $n = \rho + \sigma$, $B \in \mathbb{R}^{m \times n}$, and $\mathbf{u}_{-1} = \mathbf{0}$. Moreover, the top row of B contains gain factors $\nu_i = \begin{bmatrix} \nu_{i,1} & \cdots & \nu_{i,\rho} & \nu_{i,\rho+1} & \cdots & \nu_{i,\rho+\sigma} \end{bmatrix}$, quantifying the contribution of a given parameter to the instantaneous energy consumption. The other entries of B are zeros.

The first ρ gain factors quantifies the contribution of the i-th trajectory parameters. A first guess of these parameters is derived empirically. They are later refined by the algorithm. The following σ gain factors quantifies the contribution of the computations parameters. They are derived using the modeling tool. We use the gain factors to select the optimal control $\mathbf{u}^0$.

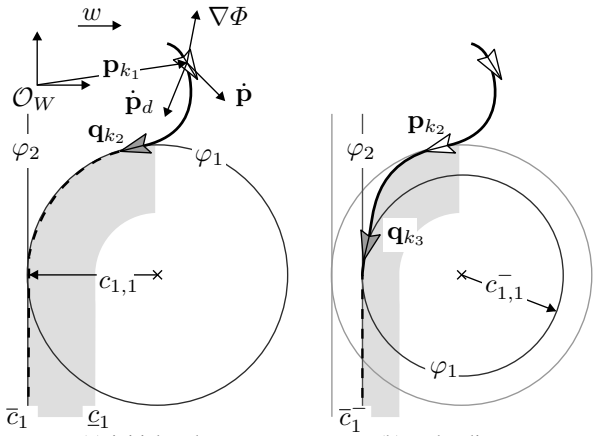
Equation (5) accounts for the energy due to the change of parameters $\mathbf{u}_k - \mathbf{u}_{k-1}$. For instance, when the trajectory φ_1 is a circle (see Figure 4), a decrement in the trajectory parameter $c_{1,1}$–the radius of the circle–adds a negative contribution. It thus simulates the lowering of instantaneous energy consumption $\nu_{1,1}(c_{1,1} - c_{1,1}^-) < 0$, for a given $\nu_{1,1} \in \mathbb{R}_{<0}$.

A. Trajectory parameters energy contribution

Let us assume that the set

$$\mathcal{P}_i := \{\mathbf{p}_k \mid \varphi_i(\mathbf{p}_k, c_{i,1}, \ldots, c_{i,\rho}) \in \mathbb{C}_i\}, \quad (6)$$

delimits the area where the i-th trajectory φ_i is free to evolve using the trajectory parameters $c_{i,1}, ..., c_{i,\rho}$ (the gray area in Figure 4). φ_i is a function of the two coordinates and is equal to zero when a point $\mathbf{p}_k$ is on the trajectory. Physically, this means the UAV is flying exactly over the nominal trajectory. The trajectory



(a) initial path (b) path adjustment

Fig. 4. Illustrative example of the alteration of a trajectory parameter

parameters allows to change the trajectory. They are a way to alter the nominal trajectory in the initial plan and thus alter the energy signal. In fact, the algorithm uses the set from Equation (6) to alter the optimal trajectory parameters c_i^0 (here we again neglect for simplicity the $\sigma = 0$ computations parameters), such that $\varphi_i(\mathbf{p}_k, c_i^0)$ has the highest instantaneous energy consumption (while still respecting the energy budget).

We derive the new position $\mathbf{p}_{k+1}$ using the function $\Phi := \varphi_i(\mathbf{p}_k, \mathbf{c}_i^0)$, computing its vector field $\nabla\Phi := \begin{bmatrix}\partial\Phi/\partial x & \partial\Phi/\partial y\end{bmatrix}^T$, and the direction to follow in the form of velocity vector [19]

$$\dot{\mathbf{p}}_d(\mathbf{p}_k) := E\nabla\Phi - k_e\Phi\nabla\Phi, \;\; E = \begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}, \quad (7)$$

where E specifies the rotation (which influence the tracking direction), and $k_e \in \mathbb{R}_{\geq 0}$ the gain to adjusts the speed of convergence. The direction the velocity vector $\dot{\mathbf{p}}_d$ is pointing at is generally different from the course heading due to the uncertainty, such as wind ($w \in \mathbb{R}$ in Figure 4).

B. Computations parameters energy contribution

Let us recall from Definition II.1 that the i-th stage Γ_i of the plan Γ contains the computations parameters which characterize the computations. We assess the energy cost of these computations using `powprofiler`, the open-source modeling tool adapted from earlier work on computational energy analysis [18], [35], and energy estimation of a fixed-wing UAV [25].

We assume the UAV carries an embedded board that runs the computations. The tool measures the instantaneous energy consumption of a subset of possible computations parameters within the computations constraints sets and builds an energy model: a linear interpolation, one per each computation.

Specifically, the computations are implemented by software components, such as ROS nodes in a ROS based system. The user implements these nodes s.t. they change the computational load by node-specific ROS parameters–the computations parameters. In a generic software component system, the user maps the computational load to the arguments. In both cases, with ROS [36] or with generic software components system [35], the tool performs automatic modeling. For instance, if the computation is a CNN object detector, the computation parameter $c_{i,\rho+1}$ corresponds frames-per-second (fps) rate and the tool changes the detection frequency to build a model. The model is then a linear multivariate interpolation for all the parameters.

We note that while the trajectory can differ for each stage, the tasks remain the same. However, the user can inhibit or enable a computation varying its computation set.

Let us further define $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as the instantaneous computational energy consumption value obtained using the tool.

The gain factors that quantifies the contribution of the computations parameters

$$\begin{bmatrix}\nu_{i,\rho+1}\\ \vdots \\ \nu_{i,\rho+\sigma}\end{bmatrix}^T = \begin{bmatrix}g(c_{i,\rho+1})\\ \vdots \\ g(c_{i,\rho+\sigma})\end{bmatrix} / \begin{bmatrix}c_{i,\rho+1} & \cdots & c_{i,\rho+\sigma}\end{bmatrix}. \quad (8)$$

Moreover, let $g(\{\emptyset\})$ be zero.

IV. Algorithm

The main purpose of the algorithm is to output a valid control sequence $\mathbf{u} := \{\mathbf{u}_0, \mathbf{u}_1, \dots\}$ at each time step given an initial plan Γ and to direct the UAV on a valid path: to solve Problem II.1.

Definition IV.1 (Valid control sequence and path)**.** A *control sequence* $\mathbf{u}$ is *valid* if for every stage Γ_{i-1}, $i \in [l]^+$ there exist a control $\mathbf{u}_k$ that produce the next stage Γ_i.

The path $\mathcal{P}$ resulting from such control sequence is a *valid path*

$$\begin{aligned}\mathcal{P} := \{\mathbf{p}_k \mid \exists \mathbf{u}_k, n \in \mathbb{Z}_{>0},\\ \langle \varphi_{i-1}(\mathbf{p}_{k-n}), \mathbf{u}_{k-n}\rangle \in \Gamma_{i-1}\\ \implies \langle \varphi_i(\mathbf{p}_k), \mathbf{u}_k\rangle \in \Gamma_i\},\end{aligned} \quad (9)$$

where for simplicity $\langle \varphi_i(\mathbf{p}_k), \mathbf{u}_k\rangle := \langle \varphi_i(\mathbf{p}_k, c_{k,1}, \dots, c_{k,\rho}), c_{k,\rho+1}, \dots, c_{k,\rho+\sigma}\rangle$.

Let us proof that the plan is periodic and one period is valid, then the plan is valid.

Theorem IV.1 (Valid periodic plan)**.** ?

—-

Consider the mission from Definition II.1, the valid mission from IV.1. Assume Assumption II.1 holds, the model of Equation (3) behaves ideally ($\mathbf{w} = \mathbf{0}, v = 0$), the initial energy coefficients state $\mathbf{q}_0$ is y_0^a/m for the first coefficient where $y_0^a \in \mathbb{R}_{>0}$ is an initial measurement[1], $(1/2)y_0^a/m$ for all the others, and the mission is valid. Then, the instantaneous energy consumption y_k is a linear combination of the state $\mathbf{q}_k$.

Proof. The proof is based on mathematical induction. Base case: we proof that $y_0 = y_0^a$. Recall the definition of the state in Equation (4). The output is $y_0 = \alpha_{0,0} + \alpha_{0,1} + \cdots + \alpha_{0,r} = y_0^a/m + (1/2)y_0^a/m + \cdots + (1/2)y_0^a/m = y_0^a$.

Induction step: by inspection of Equation (3), the output at instant k can be expressed $y_k = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_{k-1}) + p_1(k)\alpha_{0,1} + \cdots + p_r(k)\alpha_{0,r}$, where $\forall t \in \mathbb{Z}_{\geq 2}$

$$p_r(t) := \begin{cases} \prod_{i=1}^{t/2} r^3/\xi^3 & \text{for even } t \\ (r/\xi)\prod_{i=1}^{(t-1)/2} r^3/\xi^3 & \text{for odd } t \end{cases}. \quad (10)$$

Suppose k is even and the theorem holds up to k. Initial energy coefficients state $\mathbf{q}_0$ leads to $y_k = (y_0^a/m + B\mathbf{u}_0 + \cdots + B\mathbf{u}_{k-1}) + p_1(k)(1/2)y_0^a/m + \cdots + p_r(k)(1/2)y_0^a/m = y_k^a$.

We prove now that the instantaneous energy consumption at $k+1$ is still a linear combination of the state. We express the output in function of the previous state $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + (1/\xi)\beta_{k,1} + \cdots + (r/\xi)\beta_{k,r}$. Notice that the coefficients α, β have an equivalent evolution (indeed this allows to simulate the periodicity) and $\beta_{k,r} = p_r(k)\beta_{0,r}$. Thus, the output can be expressed $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + (1/\xi)p_1(k)\beta_{0,1} + \cdots + (r/\xi)p_r(k)\beta_{0,r}$. The expression is equivalent to $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + p_r(k+1)\beta_{0,r} + \cdots + p_r(k+1)\beta_{0,r}$ using the definition of p_r in Equation (10). Again, the state $\mathbf{q}_0$ leads to $y_{k+1} = (y_0^a/m + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + p_r(k+1)(1/2)y_0^a/m + \cdots + p_r(k+1)(1/2)y_0^a/m = y_{k+1}^a$, alike the previous statement, but at instant $k+1$. The proof for odd k is equivalent. ■

A. Output constraints set

We stated earlier the output y_k–the instantaneous energy consumption–evolves in $\mathbb{R}_{\geq 0}$. This is generally untrue. Physical UAVs are bounded by strict energy budgets due to battery limitations.

Let us hence consider the state of charge (SoC) of such battery with a simplistic difference equation [25]

$$\mathrm{SoC}_k = -\left(V - \sqrt{V^2 - 4R_r\tilde{V}y_k V^{-1}}\right)/2R_rQ_c, \quad (11)$$

where $V \in \mathbb{R}$ is the internal battery and $\tilde{V} \in \mathbb{R}$ the stabilized voltage, $R_r \in \mathbb{R}$ the resistance, and $Q_c \in \mathbb{R}$ the constant nominal capacity. We define the output constraints set

$$\mathbb{Y}_k := \{y_k \mid y_k \in [0, \mathrm{SoC}_k Q_c V] \subseteq \mathbb{R}_{\geq 0}\}, \quad (12)$$

and $\max \mathbb{Y}_k$ is the maximum discharge capacity by the internal battery voltage–the maximum instantaneous energy consumption.

B. Deployment algorithm

1: **procedure** STEP($\mathbf{q}_{k-1}$, $\mathbf{u}^a_{k-1}$, $\mathbf{u}^a_{k-2}$, P_{k-1})
2: $\quad \mathbf{u}_{k-1} \leftarrow \mathbf{u}_{k-1}(\max \mathbf{u}^a_{k-1}, \mathbf{u}^a_{k-2})$
3: $\quad \mathbf{u}^0_{k-1} \leftarrow \arg\max_{\mathbf{u}} \sum_{i=k-1}^{k+N-2} l(\mathbf{q}_i, \mathbf{u}_i) + V_f(\mathbf{q}_{k+N-1})$
4: $\quad \hat{\mathbf{q}}^-_k \leftarrow A\hat{\mathbf{q}}_{k-1} + B\mathbf{u}^0_{k-1}$
5: $\quad$ **if** $C\hat{\mathbf{q}}^-_k \notin \mathbb{Y}_k$ **then**
6: $\quad\quad \mathbf{u}^a_{k-1} \leftarrow \mathbf{u}^a_{k-1}/\{\max \mathbf{u}^a_{k-1}\}$
7: $\quad\quad$ **return** STEP($\mathbf{q}_{k-1}$, $\mathbf{u}^a_{k-1}$, $\mathbf{u}^a_{k-2}$,P_{k-1})
8: $\quad$ **else**
9: $\quad\quad$ **if** $|y^a_k - C\hat{\mathbf{q}}^-_k| \leq \varepsilon$ **then**
10: $\quad\quad\quad \hat{\mathbf{q}}_k \leftarrow \hat{\mathbf{q}}^-_k$
11: $\quad\quad\quad P_k \leftarrow P^-_k$
12: $\quad\quad$ **else**
13: $\quad\quad\quad P^-_k \leftarrow AP_{k-1}A^T + Q$
14: $\quad\quad\quad K \leftarrow P^-_k C^T/(CP^-_k C^T + R)$
15: $\quad\quad\quad \hat{\mathbf{q}}_k \leftarrow \hat{\mathbf{q}}^-_k + K(y^a_k - C\hat{\mathbf{q}}^-_k)$
16: $\quad\quad\quad P_k \leftarrow (I + KC)P^-_k$
17: $\quad\quad$ **end if**
18: $\quad\quad \mathbf{u}^a_k \leftarrow \mathbf{u}^a_{k-1}$
19: $\quad\quad$ **return** $(\hat{\mathbf{q}}_k, P_k, \mathbf{u}^a_k)$
20: $\quad$ **end if**
21: **end procedure**

22: **procedure** EADMPA($\mathcal{M}$, $\mathbf{p}_0$, $\mathbf{q}_0$)
23: $\quad k \leftarrow 0$
24: $\quad \mathbf{u}^a_{k-1} \leftarrow \{\emptyset\}$
25: $\quad \mathbf{p}_k \leftarrow \mathbf{p}_0$
26: $\quad \mathbf{q}_k \leftarrow \mathbf{q}_0$
27: $\quad$ **while** $k \leq t_f$ **do**
28: $\quad\quad \mathbf{u}^a_k \leftarrow \{\mathbf{u}^a_k \mid (\varphi_i(\mathbf{p}_k, \mathbf{c}_i), \Psi(\mathbf{s_i})) \in \lambda(\mathbf{p}_k)\}$
29: $\quad\quad (\mathbf{q}_k, P_k, \mathbf{u}^a_k) \leftarrow$STEP($\mathbf{q}_k$, $\mathbf{u}^a_k$, $\mathbf{u}^a_{k-1}$, P_k)
30: $\quad\quad \mathbf{p}_k \leftarrow \mathbf{p}_k\dot{\mathbf{p}}_d(\mathbf{p}_k)/v$
31: $\quad\quad \mathbf{u}^a_{k-1} \leftarrow \mathbf{u}^a_k$
32: $\quad\quad k \leftarrow k + 1$
33: $\quad$ **end while**
34: **end procedure**

Per each time step k (the final time t_f is unknown), the algorithm updates the state–the position at line 30 and the energy coefficients at line 29–and the control input. Note that the position can be computed directly from Equation (7). If the velocity is $v \in \mathbb{R}_{\geq 0}$, and the starting point $\mathbf{p}_0$, $\mathbf{p}_{k+1} = \mathbf{p}_k\dot{\mathbf{p}}_d(\mathbf{p}_k)/v$.

In detail, initial guess for $P_0 \in \mathbb{R}^{j\times j}$ is positive definite and derived empirically, for $\mathbf{q}_0$ the initial mea-

[1] y_0^a can be the initial measurement or the measurement from a previous instance of the algorithm

surement is distributed to the coefficients (see Theorem IV.1). Line 2 selects the maximum possible control from the current control input. Line 3 uses robust output feedback model predictive control (MPC) [34] to select the optimal control $\mathbf{u}^0$ for a given horizon $N \in \mathbb{Z}_{>0}$ from the cost function

$$\begin{aligned} l(\mathbf{q}_k, \mathbf{u}_k) &:= (1/2)(\mathbf{q}_k^T Q \mathbf{q}_k + \mathbf{u}_k^T R \mathbf{u}_k), \\ V_f(\mathbf{q}_k) &:= (1/2)(\mathbf{q}_k^T P_f \mathbf{q}_k), \end{aligned} \tag{13}$$

where matrices $Q \in \mathbb{R}^{j \times j}, R \in \mathbb{R}^{l \times l}$ are positive definite.

Follows a check if the mission can finish without the eventuality of battery discharge (output constraints satisfaction) at line 5, with the control input being eventually updated and the process reiterated at line 7.

Before the next step, state estimator–the discrete-time Kalman filter [37] at lines 13–16–predicts the state $\mathbf{q}$ if the modeled instantaneous energy consumption diverges from the sensor's value y_k^a more than a given $\varepsilon \in \mathbb{R}_{\geq 0}$, or sensor measurements are unavailable ($y_k^a = 0$).

V. Results

Some preliminary results are discussed. Further analysis and re-implementation of the functionality of the algorithm is needed.

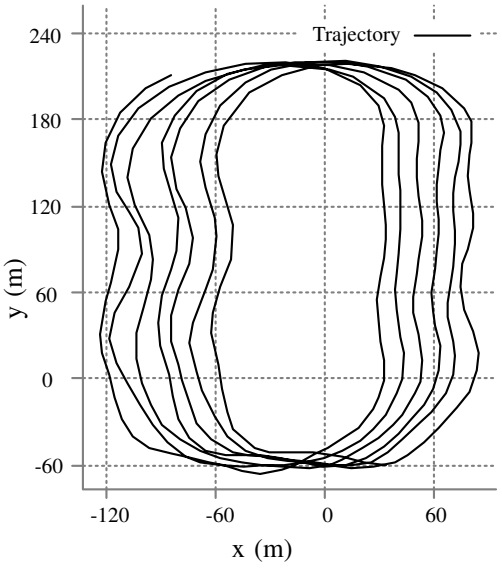


Fig. 5. The true trajectory. The trajectory is retrieved from the flight log.

VI. Conclusion and Future Work

*

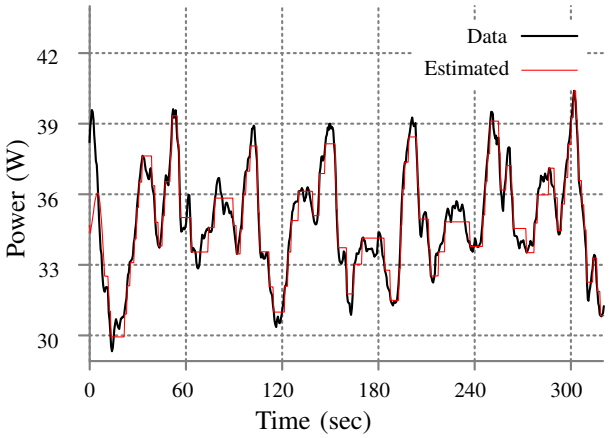


Fig. 6. Derived energy consumption from the sensors (throttle). Red line is the estimated value. We use KF for state estimation when the difference between the sensor and model value is $> \varepsilon$. Otherwise we evolve the state without noise.

References

[1] P. Daponte, L. De Vito, L. Glielmo, L. Iannelli, D. Liuzza, F. Picariello, and G. Silano, "A review on the use of drones for precision agriculture," in *IOP Conference Series: Earth and Environmental Science*, vol. 275, no. 1. IOP Publishing, 2019, p. 012022.

[2] Paparazzi. UAV open-source project. [Online]. Available: http://wiki.paparazziuav.org/

[3] PX4. PX4 open-source autopilot. [Online]. Available: https://px4.io/

[4] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Energy-efficient motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4344–4349.

[5] M. Wahab, F. Rios-Gutierrez, and A. El Shahat, *Energy modeling of differential drive robots*. IEEE, 2015.

[6] C. H. Kim and B. K. Kim, "Energy-saving 3-step velocity control algorithm for battery-powered wheeled mobile robots," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2375–2380.

[7] H. Kim and B.-K. Kim, "Minimum-energy translational trajectory planning for battery-powered three-wheeled omnidirectional mobile robots," in *2008 10th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2008, pp. 1730–1735.

[8] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, 2005, pp. 492–497.

[9] S. T. H. Rizvi, G. Cabodi, D. Patti, and M. M. Gulzar, "A general-purpose graphics processing unit (gpgpu)-accelerated robotic controller using a low power mobile platform," *Journal of Low Power Electronics and Applications*, vol. 7, no. 2, p. 10, 2017.

[10] A. Abramov, K. Pauwels, J. Papon, F. Worgotter, and B. Dellen, "Real-time segmentation of stereo videos on a portable system with a mobile gpu," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1292–1305, 2012.

[11] M. T. Satria, S. Gurumani, W. Zheng, K. P. Tee, A. Koh, P. Yu, K. Rupnow, and D. Chen, "Real-time system-level implementation of a telepresence robot using an embedded gpu platform," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1445–1448.

[12] U. Jaramillo-Avila, J. M. Aitken, and S. R. Anderson, "Visual saliency with foveated images for fast object detection and recognition in mobile robots using low-power embedded gpus,"

in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 773–778.

[13] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[14] A. Sadrpour, J. Jin, and A. G. Ulsoy, "Mission energy prediction for unmanned ground vehicles using real-time measurements and prior knowledge," *Journal of Field Robotics*, vol. 30, no. 3, pp. 399–414, 2013.

[15] ——, "Experimental validation of mission energy prediction model for unmanned ground vehicles," in *2013 American Control Conference*. IEEE, 2013, pp. 5960–5965.

[16] F. Morbidi, R. Cano, and D. Lara, "Minimum-energy path generation for a quadrotor uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1492–1498.

[17] N. Kreciglowa, K. Karydis, and V. Kumar, "Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 656–662.

[18] A. Seewald, U. P. Schultz, E. Ebeid, and H. S. Midtiby, "Coarse-grained computation-oriented energy modeling for heterogeneous parallel embedded systems," *International Journal of Parallel Programming*, pp. 1–22, 2019. [Online]. Available: https://adamseew.bitbucket.io/short/coarse2019

[19] H. G. De Marina, Y. A. Kapitanyuk, M. Bronz, G. Hattenberger, and M. Cao, "Guidance algorithm for smooth trajectory tracking of a fixed wing uav flying in wind flows," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 5740–5745.

[20] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 3553–3559.

[21] V. M. Gonçalves, L. C. Pimenta, C. A. Maia, B. C. Dutra, and G. A. Pereira, "Vector fields for robot navigation along time-varying curves in n-dimensions," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 647–659, 2010.

[22] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2513–2518.

[23] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6567–6572.

[24] Y. A. Kapitanyuk, A. V. Proskurnikov, and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1372–1385, 2017.

[25] A. Seewald, H. Garcia de Marina, H. S. Midtiby, and U. P. Schultz, "Mechanical and computational energy estimation of a fixed-wing drone," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2020, pp. 135–142. [Online]. Available: https://adamseew.bitbucket.io/short/mechanical2020

[26] S. S. H. Hajjaj and K. S. M. Sahari, "Review of research in the area of agriculture mobile robots," in *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications*. Springer, 2014, pp. 107–117.

[27] F. Qingchun, Z. Wengang, Q. Quan, J. Kai, and G. Rui, "Study on strawberry robotic harvesting system," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 1. IEEE, 2012, pp. 320–324.

[28] F. Dong, W. Heinemann, and R. Kasper, "Development of a row guidance system for an autonomous robot for white asparagus harvesting," *Computers and Electronics in Agriculture*, vol. 79, no. 2, pp. 216–225, 2011.

[29] Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu, "Design and control of an apple harvesting robot," *Biosystems engineering*, vol. 110, no. 2, pp. 112–122, 2011.

[30] A. Aljanobi, S. Al-Hamed, and S. Al-Suhaibani, "A setup of mobile robotic unit for fruit harvesting," in *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*. IEEE, 2010, pp. 105–108.

[31] Z. Li, J. Liu, P. Li, and W. Li, "Analysis of workspace and kinematics for a tomato harvesting robot," in *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 1. IEEE, 2008, pp. 823–827.

[32] Y. Edan, D. Rogozin, T. Flash, and G. E. Miles, "Robotic melon harvesting," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 831–835, 2000.

[33] V. Puri, A. Nayyar, and L. Raja, "Agriculture drones: A modern breakthrough in precision agriculture," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 507–518, 2017.

[34] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[35] A. Seewald, U. P. Schultz, J. Roeder, B. Rouxel, and C. Grelck, "Component-based computation-energy modeling for embedded systems," in *Proceedings Companion of the 2019 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. ACM, 2019, pp. 5–6. [Online]. Available: https://adamseew.bitbucket.io/short/component2019

[36] G. Zamanakos, A. Seewald, H. S. Midtiby, and U. P. Schultz, "Energy-aware design of vision-based autonomous tracking and landing of a uav," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2020, pp. 294–297. [Online]. Available: https://adamseew.bitbucket.io/short/energy2020

[37] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[38] B. Kuo, *Automatic Control Systems*, ser. Electrical engineering series. Prentice-Hall, 1967.

[39] K. Ogata, *Modern Control Engineering*. Prentice Hall, 2002.

[40] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM review*, vol. 45, no. 1, pp. 3–49, 2003.

APPENDIX I
PROOF OF LEMMA III.1

We propose a formal proof of Lemma III.1. The proof justifies the choice of the items of the matrices A, C and of the initial guess $\mathbf{q}_0$ in Equation (4). We write these elements such that the coefficients of the series $a_0, \ldots, b_r$ are the same as the coefficients of the state $\alpha_0, \ldots, \beta_r$.

Let us re-write the Fourier series expression in Equation (2) in its complex form with the well-known Euler's formula $e^{it} = \cos t + i \sin t$. With $t = \omega j t$, we find the expression for $\cos \omega j t = (e^{i\omega j t} + e^{-i\omega j t})/2$ and $\sin \omega j t = (e^{i\omega j t} - e^{-i\omega j t})/(2i)$ by substitution of $\sin \omega j t$ and $\cos \omega j t$ respectively. This leads [38]

$$\begin{aligned} h(t) = a_0/T + (1/T) \sum_{j=1}^{r} e^{i\omega j t}(a_j - i b_j) + \\ (1/T) \sum_{j=1}^{r} e^{-i\omega j t}(a_j + i b_j), \end{aligned} \tag{14}$$

where i is the imaginary unit.

The solution at time t can be expressed $\mathbf{q} = e^{At}\mathbf{q}_0$. Both the solution and the system in Equation (3) are well established expressions derived using standard textbooks [38], [39]. To solve the matrix exponential e^{At}, we use the eigenvectors matrix decomposition method [40].

The method works on the similarity transformation $A = VDV^{-1}$. The power series definition of e^{At} implies $e^{At} = Ve^{Dt}V^{-1}$ [40]. We consider the non-singular matrix V, whose columns are eigenvectors of A; $V := \begin{bmatrix} v_0 & v_1^0 & v_1^1 & \ldots & v_r^0 & v_r^1 \end{bmatrix}$. We then consider the diagonal matrix of eigenvalues $D = \operatorname{diag}(\lambda_0, \lambda_1^0, \lambda_1^1, \ldots, \lambda_r^0, \lambda_r^1)$. λ_0 is the eigenvalue associated to the first item of A. λ_j^0, λ_j^1 are the two eigenvalues associated with the block A_j. We can write $Av_j = \lambda_j v_j \ \ \forall j = \{1, \ldots, m\}$, and $AV = VD$.

We apply the approach in terms of Equation (3), under the assumptions; $\dot{\mathbf{q}} = A\mathbf{q}$. The linear combination of the initial guess and the generic solution

$$F\mathbf{q}(0) = \gamma_0 v_0 + \sum_{k=0}^{1} \sum_{j=1}^{r} \gamma_j v_j^k$$

$$F\mathbf{q}(t) = \gamma_0 e^{\lambda_0 t} v_0 + \sum_{k=0}^{1} \sum_{j=1}^{r} \gamma_j e^{\lambda_j t} v_j^k$$

where $F = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$ is a properly sized vector of ones.

Let us consider the second expression. It represents the linear combination of all the coefficients of the state at time t. It can also be expressed in the following form

$$\begin{aligned} F\mathbf{q}(t)/T = \gamma_0 e^{\lambda_0 t} v_0/T + (1/T) \sum_{j=1}^{r} \gamma_j e^{\lambda_j^0 t} v_j^0 + \\ (1/T) \sum_{j=1}^{r} \gamma_j e^{\lambda_j^1 t} v_j^1. \end{aligned} \tag{15}$$

We proof that the eigenvalues λ and eigenvectors V are such that Equation (15) is equivalent to Equation (14).

The matrix A is a block diagonal matrix, so we can express its determinant as the multiplication of the determinants of its blocks $\det(A) = \det(0) \times \det(A_1) \times \cdots \times \det(A_r)$. We proof the first determinant and the others separately.

We proof that the first term of both the Equation (14) and (15) matches. We find the eigenvalue from $\det(0) = 0$, which is $\lambda_0 = 0$. The corresponding eigenvector can be chosen arbitrarily $(0 - \lambda_0)v_0 = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} \ \forall v_0$, thus we choose $v_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$. We find the value γ_0 of the vector γ so that the terms are equal $\gamma_0 = \begin{bmatrix} a_0 & 0 & \cdots & 0 \end{bmatrix}$.

Then, we proof that all the terms in the sum of both the Equations (14) and (15) match.

For the first block A_1, we find the eigenvalues from $\det(A_1 - \lambda I) = 0$. The polynomial $\lambda^2 + \omega^2$, gives two complex roots–the two eigenvalues $\lambda_1^0 = i\omega$ and $\lambda_1^1 = -i\omega$. The eigenvector associated with the eigenvalue λ_1^0 is $v_1^0 = \begin{bmatrix} 0 & -i & 1 & 0 & \cdots & 0 \end{bmatrix}^T$. The eigenvector associated with the eigenvalue λ_1^1 is $v_1^1 = \begin{bmatrix} 0 & i & 1 & 0 & \cdots & 0 \end{bmatrix}^T$. Again, we find the values γ_1 of the vector γ such that the equivalences

$$\begin{cases} e^{i\omega t}(a_1 - i b_1) & = \gamma_1 e^{i\omega t} v_1^0 \\ e^{-i\omega t}(a_1 + i b_1) & = \gamma_1 e^{i\omega t} v_1^1 \end{cases}$$

hold. They hold for $\gamma_1 = \begin{bmatrix} b_1 & a_1 \end{bmatrix}$. The proof for the remaining $r - 1$ blocks is equivalent.

The initial guess is build such that the sum of the coefficients is the same in both the signals. The output matrix is built as follows. The coefficient $1/T$ accounts for the period in Equations (14–15) and (2). At time instant zero, the coefficients b_j are not present (and the coefficients a_j are doubled for each $j = 1, 2, \ldots, r$). To match the outputs $h(t) = y(t)$–or equivalently $F\mathbf{q}(t)/T = C\mathbf{q}(t)$–we define $C = 1/T \begin{bmatrix} 1 & 1 & 0 & \cdots & 1 & 0 \end{bmatrix}$. We thus conclude that the lemma holds.

We note that the signal would still be periodic with another linear combination of coefficients (for instance, $C = d \begin{bmatrix} 1 & 0 & 1 & \cdots & 0 & 1 \end{bmatrix}$, or $d \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$ for a constant value $d \in \mathbb{R}$). Even if both the signals h and y would be still periodic, they would not be equal.

■

APPENDIX II
PLAN SPECIFICATION

A. Format

The algorithm reads the plan Γ from a file–the plan specification–with a specific file format. We refer the reader to Appendix II-B for an example.

The first stage Γ_1 is expressed in the first line of the file

1: k_e**;** r**;** $\varphi_1(\mathbf{p}_k, c_{1,1}, \ldots, c_{1,\rho})$

where k_e is a convergence value, r a rotation, and φ_1 the trajectory. In the next subsection where we discuss the physical meaning of the convergence value and rotation. The rotation is expressed in binary format with zero being the counter-clockwise rotation, one the clockwise rotation.

The next line specifies the triggering point $\mathbf{p}_{\Gamma_1}$

2: x**,** y

where x and y are the two coordinates of the point. The UAV reaches the point within a given radius (a parameter of the algorithm) and enters the next stage Γ_2. The coordinates can be expressed in function of the trajectory parameters $c_{1,1}, \ldots, c_{1,\rho}$.

The line

3: **[**$\underline{c}_1$**,** $\overline{c}_1$**]**

specifies the lower $\underline{c}_1$ and upper $\overline{c}_1$ bounds of the trajectory φ_1.

The following σ lines

4: **[**$\underline{c}_{1,j}$**,** $\overline{c}_{1,j}$**]**

specifies the lower $\underline{c}_{1,j}$ and upper $\overline{c}_{1,j}$ bounds of the j-th computation parameter set. They are in ascending order.

This means that for the first stage, the algorithm can alter the trajectory satisfying $\varphi_1 \in \mathbb{C}_1$. It can alter the computations individually. In fact the j-th computation parameter has to be in $\mathbb{S}_{1,j}$.

Follows the entries for stage Γ_2 (equivalent to lines 1–4), Γ_3 and so on, until the last stage. We assume that the last triggering point is the final UAV position.

B. Example

We discuss how to specify an initial mission plan with an example of a survey mission. The reader can later execute the initial mission plan using the simulator in Appendix III. The plan is user-defined. It specifies the trajectory and tasks of a mission along the QoS and TEEs sets.

Recall that the mission is composed of a set of stages (Section II). The plan has a variable number of entries per each stage. We plan the mission in Figures 1, 5. The first line

1: **0.0003;90;(x+45)^2+(y-146)^2-4900+c1**

corresponds to TEE $\varphi_0(\mathbf{p}_k, c_{0,1}) := (x+45)^2 + (y-146)^2 - 4900 + c_{0,1}$ in the initial stage $\mathcal{M}_0$. The TEE is a circle. The gain is $k_e = 3 \cdot 10^{-4}$, and the rotation is of ninety degrees–we use the rotation matrix E from Equation (7). Note that line 1 further specifies the adjustment **c1** of the radius of the circle. It corresponds to $c_{0,1}$ in Definition II.1 (zero numerates the stage, one the adjustment).

The following line

2: **-sqrt(4900+c1)-45,146**

specifies the triggering point. **sqrt** is the square root. The UAV reaches the triggering point within a given ε (the tolerance), and the algorithm switches to the next stage $\mathcal{M}_1$. The algorithm ensures that the overall mission is still valid–Equation (9). It ensure that the TEEs and QoS constraints sets are satisfied with the generated control input. Note that the triggering point is in function of the adjustment **c1**. This is necessary; an adjustment in one stage affects the overall flight.

Let us assume the mission has two tasks. One task is the CNN object detection algorithm. Its computation $s_{0,1}$ is the fps rate. Another task is the variable key-size encryption algorithm. Its computation $s_{0,2}$ is the key-size. Then in mission plan

3: **[-3*10^3,0]**
4: **[2,10]**
5: **[32,448]**

line 3 corresponds to TEE φ_0's set $\mathbb{C}_0 = [-3 \cdot 10^3, 0]$. The algorithm selects $c_{0,1} \in \mathbb{C}_0$. Line 4 corresponds to the QoS set $\mathbb{S}_{0,1}$ for the computation $s_{0,1}$. Line 4 corresponds to the QoS set $\mathbb{S}_{0,2}$ for the computation $s_{0,2}$. Lines 3–5 must follow the ascending order of adjustments and computations. Lines 1–5 all describe the stage $\mathcal{M}_0$.

Once the UAV reaches the triggering point (line 2) and there are no further lines, the algorithm terminates. If there are further lines, the stage switches to $\mathcal{M}_1$

6: **0.05;270;x+sqrt(4900+c1)+45**
7: **-sqrt(4900+c1)-45,11**
8: **[-3*10^3,0]**
9: **[2,10]**
10: **[32,448]**

line 6 corresponds to TEE $\varphi_1(\mathbf{p}_k, c_{0,1}) := x + \sqrt{4900 + c_{0,1}} + 45$, a linear equation that intersects the triggering point (line 2). The gain is $k_e = 5 \cdot 10^{-2}$, and the rotation is opposite to line 1. The algorithm computes $-E$ from the rotation matrix in Equation 7.

The gain is different from the one used in φ_0 as different equations require different convergence rate. When we follow a circle it is useful to turn the rotation direction in advance. When we follow a straight line, it is preferable to turn the rotation direction closer to the line. We refer to the simulator in Appendix III for more details on how to tune the gain k_e. The reader can simulate different rates to see how they affect the flight.

The rotation is likewise different. In fact, the rotation matrix E points upwards in the space. It guides the UAV in the counter-clockwise direction (see Figure 4; the UAV is traveling counter-clockwise). The rotation matrix $-E$ points downwards and guides the UAV in the clockwise direction.

The triggering point of the stage $\mathcal{M}_1$ at line 7 also depends on the adjustment **c1**. Lines 8–10 specifies the constraints sets. They are expressed the same way as lines 3–5. They have to be specified explicitly for each stage, although they don't change. The current implementation of the algorithm has no other means to distinguish different constraints sets.

Again, if there are further lines and UAV reaches the triggering point (line 7), the algorithm switches the stage to $\mathcal{M}_2$

```
11: 0.0003;90;(x+sqrt(4900+c1)-30)^2
                +(y-11)^2-5625
12: -sqrt(4900+c1)+105,11
13: [-3*10^3,0]
14: [2,10]
15: [32,448]
```

line 11 corresponds to TEE $\varphi_2(\mathbf{p}_k, c_{0,1}) := (x + \sqrt{4900 + c_{0,1}} - 30)^2 + (y - 11)^2 - 5625$, a circle of fixed size that changes the coordinates in function of the adjustment **c1**. The rotation is counter-clockwise, alike φ_0.

The lines

```
16: 0.05;90;x+sqrt(4900+c1)-105
17: -sqrt(4900+c1)+105,147
18: [-3*10^3,0]
19: [2,10]
20: [32,448]
21: 0.0003;90;(x-sqrt(4900+c1)+105)^2
                +(y-147)^2-4900+c1
22: -3*sqrt(4900+c1)+105,147
23: [-3*10^3,0]
```

If there are any following stages in the survey, they defines the stages $\mathcal{M}_3$ and $\mathcal{M}_4$. Line 16 corresponds to TEE $\varphi_3(\mathbf{p}_k, c_{0,1}) := x + \sqrt{4900 + c_{0,1}} - 105$. Line 21 corresponds to TEE $\varphi_4(\mathbf{p}_k, c_{0,1}) := (x - \sqrt{4900 + c_{0,1}} + 105)^2 + (y - 147)^2 - 4900 + c_{0,1}$.

Note that the survey mission example changes the radius of the circular TEE φ_0 through the parameter $c_{0,1}$. It also changes the radius of φ_4 in the same way. The other TEEs $(\varphi_1, \varphi_2, \varphi_3)$ displacement alters the distance between the lines in the survey. A familiar pattern from Figure 4. For simplicity, we suppose line 22 describes the last point of the mission. We further suppose the user does not desire to process any task in the last stage $\mathcal{M}_4$. To inhibit the computations

```
24: [0,0]
25: [0,0]
```

are constructed the same way. The stages $\mathcal{M}_i$ for $i := \{0, 4, 8, \dots\}$ contain TEE circle with variable radius. For $i := \{1, 3, 5, \dots\}$ TEE line with variable displacement. For $i := \{2, 6, 10, \dots\}$ TEE circle with fixed radius and variable displacement.

We need a further abstraction before we can simulate a mission in Appendix III. The algorithm needs to know how to model the tasks–the software components.

APPENDIX III
SIMULATOR

We discuss how to execute the initial mission plan and the component specification in the simulator. The reader can use the simulator to simulate different environmental interference and their effects on the mission.

The simulator is written in MATLAB and released under MIT license. It implements the algorithm (Subsection IV) providing a position $\mathbf{p}_k$ at each time step k and an energy sensor's value y_k^a. It is derived from empirical data. It is an abstraction of the physical observations.

The user is prompted to initialize the trajectory and the algortihm.