# 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 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.

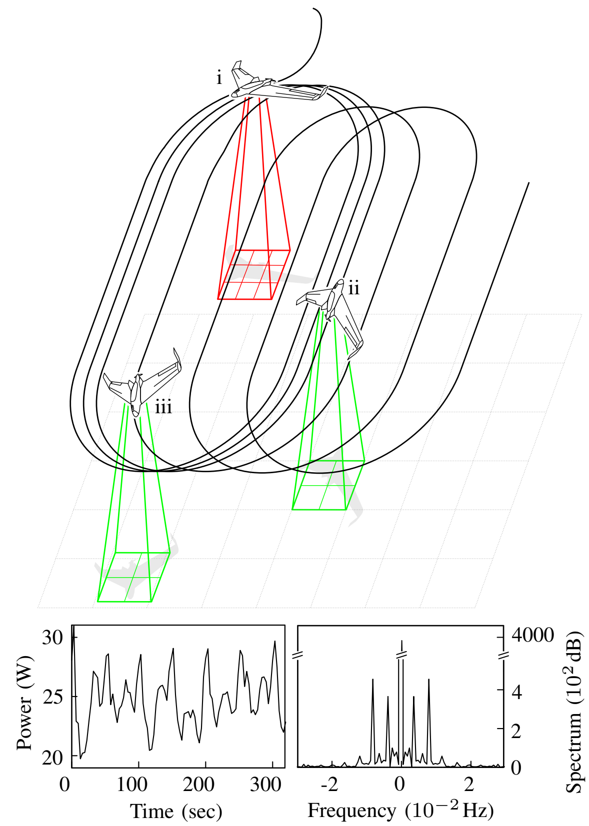



Fig. 1. An initial plan of an agricultural scenario replanned dynamically in terms of both the path (ii travels a more accurate survey than iii), and computations (i elaborates more images per second than ii). In bottom left, the collected energy data of the UAV flying the scenario, along the power spectrum (bottom right) 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

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 plan being a set of tasks and a path, all focus on 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. To model the path we use multiple trajectories and denote each 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 guidance 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. In Figure 1, some collected energy data from a UAV flying a survey scenario along its power spectrum motivates our choice.

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 plan. We observe that not only the path but also the computations significantly impact the energy, 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), in presence of a standard battery.

The remainder of the paper is organized as follows. The overview of dynamic planning, some preliminaries, and problem definition are provided in Section II. We show a suitable model for the energy in Section III, and propose the algorithm in Section IV. In Section V, we present the results and showcases the performances. We then derive some conclusions in Section VI. Finally, we provide some additional information and examples in Appendixes I–II.

## II. Planning Overview

The algorithm inputs a user-specified initial plan that consist of different stages. At each stage the UAV flies a trajectory, and does some computations. Per each stage the plan contains some parameters that allow to alter the path and computations along an energy budget. The alterations are bounded. There is one trajectory constraint set which bounds the path and multiple computation constraint sets, one per each computation parameter, that bound computations. In Figure 1, there are two parameters. One relative to the trajectory (ii has indeed a shorter distance between the survey lines then iii), and the other one to the computations (i processes more images per second then ii).

The algorithm outputs the control (the parameters) using output model predictive control (MPC) [34]. The UAV utilizes the control to influence its energy consumption to meet the energy budget. The control is data-driven. Energy sensor data estimates some coefficients of an energy model used to predict future energy consumption in presence of uncertainty. The energy budget is the battery capacity and other battery parameters. These are fixed values that are not replanned by the algorithm. Our goal is to complete the plan with the highest possible parameters as the UAV flies and its batteries drain.

### A. Plan definition

Let us adopt the following mathematical notation. Given an integer $a$, $[a]$ is the set $\{0, 1, \dots, a\}$, $[a]^+$ the set $[a]/\{0\}$. Bold lower-case letters indicates vectors. $c_{i,j}$ specifies the $j$-th parameter of the $i$-th parameters set $c_i$. $c^0_{i,j}$ 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 $\langle c_i\rangle$ of $n$ parameters.

Let us assume that the trajectory at stage $i$ can be altered with $\rho$ trajectory parameters $c^\rho_i := \{c_{i,1}, c_{i,2}, \dots, c_{i,\rho}\}$, and the computations with $\sigma$ computations parameters $c^\sigma_i := \{c_{i,\rho+1}, c_{i,\rho+2}, \dots, c_{i,\rho+\sigma}\}$. 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.

**Definition II.1** (Stage, plan, triggering, and final point)**.** The $i$-th *stage* $\Gamma_i$ at time instant $k$ of a plan $\Gamma$ is defined

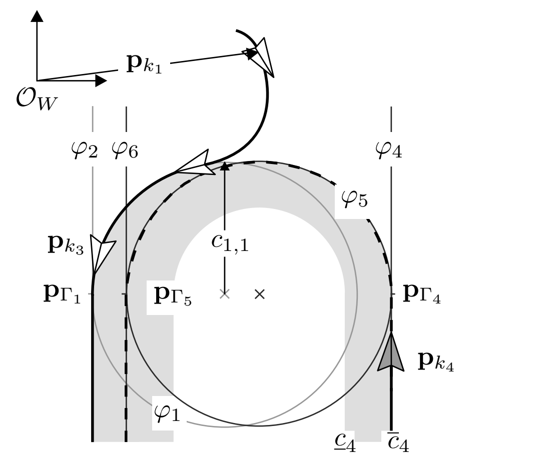



Fig. 2. The notation on a small initial slice of the plan of Figure 1 in 2D.

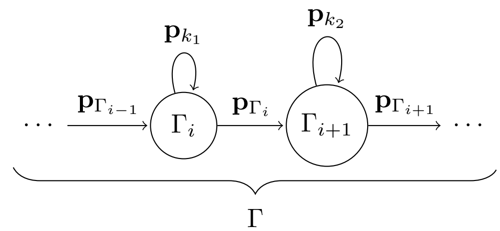



Fig. 3. The plan defined as a FSM

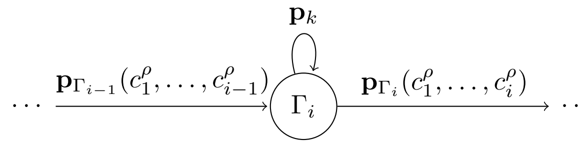



Fig. 4. Detail of the stage $\Gamma_i$ in the FSM

as the ordered list

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

where $\mathcal{C}_i := [\underline{c}_i, \overline{c}_i] \subseteq \mathbb{R}$ is the trajectory constraint set, and $\mathcal{S}_{i,j} := [\underline{c}_{i,\rho+j}, \overline{c}_{i,\rho+j}] \subseteq \mathbb{Z}_{\geq 0}$ the $j$-th computation constraint 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 $\mathcal{O}_W$. In Figure 2, $\varphi_1, \dots, \varphi_6$ are trajectories. $\varphi_1$ and $\varphi_5$ are circles, while $\varphi_2$, $\varphi_4$, and $\varphi_6$ are lines. They are all relative to different stages $\Gamma_1, \Gamma_2, \dots$. The constaints set $\mathcal{C}_1, \mathcal{C}_2, \dots$ forms the area where the trajectories $\varphi_1, \varphi_2, \dots$ can be altered with the parameters $c_{i,1}, \dots, c_{i,\rho}$ (gray area in the figure). This area is bounded by $\underline{c}_i, \overline{c}_i$, and can be different per each stage (in Figure 2, the area relative to $\Gamma_4$ is bounded by $\underline{c}_4, \overline{c}_4$).

The *plan* is a finite state machine (FSM) $\Gamma$ 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 $\Gamma_i$ and $\Gamma_{i+1}$ is called *triggering point*. In Figure 2, $\mathbf{p}_{\Gamma_1}$ allows the transition between $\Gamma_1$ and $\Gamma_2$, $\mathbf{p}_{\Gamma_4}$ between $\Gamma_4$ and $\Gamma_5$, and $\mathbf{p}_{\Gamma_5}$ between $\Gamma_5$ and $\Gamma_6$. The last triggering point $\mathbf{p}_{\Gamma_l}$ relative to the last stage $\Gamma_l$ is called *final point*.

A slice of the plan in Figure 3 shows the transition between the stages with the FSM. The triggering point $\mathbf{p}_{\Gamma_{i-1}}$ allows the transition to the stage $\Gamma_i$. The UAV remains in the stage with any generic point $\mathbf{p}_{k_1}$. It eventually enters the stage $\Gamma_{i+1}$ with the triggering point $\mathbf{p}_{\Gamma_i}$ and so on, until it reaches the final point.

Generally, one can express the triggering points in function of the $i$-th trajectory parameters $c_i^\rho$, or any previous trajectory parameters, propagating the information therein if necessary (see Figure 4).

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.

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

### *B. Problem formulation*

In order to formulate the problem, we considerate the next assumption.

**Assumption II.1** (Plan periodicity)**.** Given an initial plan $\Gamma$ consisting of $l$ stages, a generic starting point $\mathbf{p}$, the current levels of the trajectory parameters $c_1^\rho$, and a shift $\mathbf{d} := (x_d, y_d)$, there exists a constant $n \in \mathbb{Z}_{>0}$ ($n < l, l/n \in \mathbb{Z}$) such that

$$\begin{aligned}&\varphi_{(i-1)n+j}(\mathbf{p} + (i-1)\mathbf{d}, c_1^\rho) \approx \\ &\quad \varphi_{in+j}(\mathbf{p} + i\mathbf{d}, c_1^\rho),\ \ \forall i \in [l/n - 1]^+, j \in [n]^+.\end{aligned}$$

Physically, this means that the plan is approximately periodic. The justification for this assumptions originates from empirical data (we refer the reader back to the illustrative abstract in Figure 1). It indeed lays in the autonomous nature of the UAVs scenario we are interested into planning, with the UAV being expected to iterate over trajectories and computations attending some triggering events (such as detections). We will see this assumption being eased in practice for non-periodic plans in Section V.

**Definition II.2** (Period)**.** The period $T \in \mathbb{R}_{>0}$ is the time between $\varphi_{(i-1)n+j}$ and $\varphi_{in+j}$ in Assumption II.1.

From Assumption II.1 follows the next proposition.

**Proposition II.2** (Necessary and sufficient condition for periodicity)**.** The plan $\Gamma$ is periodic if

$$\varphi_{(i-1)n+j}(\mathbf{p}+(i-1)\mathbf{d},c_1^{\rho})-\varphi_{in+j}(\mathbf{p}+i\mathbf{d},c_1^{\rho})=e_j,\ \ \forall i\in[l/n-1]^+,j\in[n]^+.$$

Moreover, if the plan $\Gamma$ is periodic, then

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

where the trajectories in the first column are evaluated on $\mathbf{p},c_1^{\rho}$, in the second on $\mathbf{p}+\mathbf{d},c_1^{\rho}$, and in the last on $\mathbf{p}+(l/n-1)\mathbf{d},c_1^{\rho}$.

*Proof.* ($\Longrightarrow$) The expression implies that $\varphi_{(i-1)n+j}$ is the same but shifted function in space $\varphi_{(i-1)n+j}=\varphi_{in+j}+e_j$ and $e_j$ is the $j$-th constant difference. It is the same for trajectories selected at each period. This is equivalent to Assumption II.1.

($\Longleftarrow$) The column rank of the matrix in the proposition 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 (from $\Longrightarrow$). ■

The proposition is a necessary ($\Longleftarrow$) and sufficient ($\Longrightarrow$) condition for periodicity. The algorithm finds $n$ initializing it to one and finding the value which satisfies the proposition. It then measures the time between the stages and assumes the initial period is one. The periods might be different for different $j$s due to atmospheric interferences.

**Problem II.1** (UAV planning problem)**.** Consider an initial plan $\Gamma$ from Definition II.1 that satisfies Assumption II.1. We are interested in the planning of the parameters $c_i,\ \forall i\in[l]^+$ and in the guidance to the path resulting from such plan for the UAV planning problem.

## III. Periodic Energy Model

We refer to the instantaneous energy consumption evolution simply as the energy signal. We model the energy using energy coefficients $\mathbf{q}\in\mathbb{R}^m$ that characterize such 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.

We proof a relation between the energy signal and the energy coefficients in Lemma III.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 trajectory in Subsection III-A, and 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 energy signal

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

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 (1) and by the output of a linear model

$$\begin{aligned}\dot{\mathbf{q}}&=A\mathbf{q}+B\mathbf{u},\\ y&=C\mathbf{q},\end{aligned}\quad (2)$$

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

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

$$\begin{aligned}\mathbf{q}&=\begin{bmatrix}\alpha_0 & \alpha_1 & \beta_1 & \cdots & \alpha_r & \beta_r\end{bmatrix}^T,\\ A&=\begin{bmatrix}0 & & & \\ & A_1 & & \\ & & \ddots & \\ & & & A_r\end{bmatrix},\ A_j\begin{bmatrix}0 & \omega j\\ -\omega j & 0\end{bmatrix},\\ C&=(1/T)\begin{bmatrix}1 & 1 & 0 & \cdots & 1 & 0\end{bmatrix},\end{aligned}\quad (3)$$

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 $A_1,\dots,A_r$, the remaining entries are zeros.

The linear model in Equation (2) allows us to include the control in the model of Equation (1).

**Lemma III.1** (Signal, output equality)**.** Suppose control $\mathbf{u}$ is a zero vector, matrices $A,C$ are described by Equation (3), 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 (1) is equal to the output $y$ in Equation (2).

*Proof.* The equality of the signal and output 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, where we justify the choices of the items of the matrices and of the initial guess. ■

Let us consider for practical reasons 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 $\Delta t$.

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

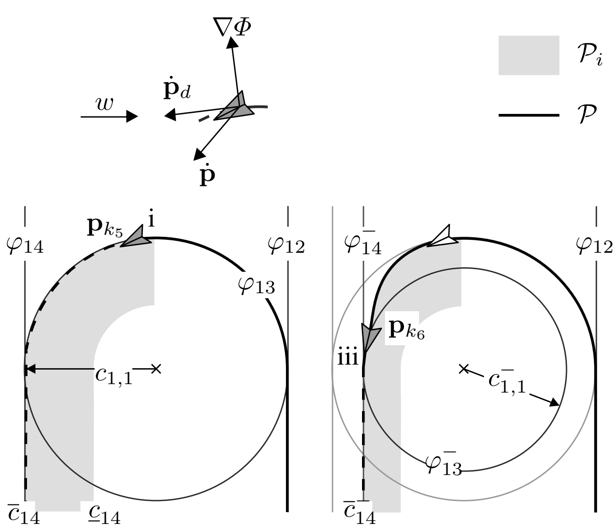
The control along with the input matrix

$$\mathbf{u}_k = \begin{bmatrix} c_{k,1} & \cdots & c_{k,\rho} & c_{k,\rho+1} & \cdots & c_{k,\rho+\sigma} \end{bmatrix}^T, \\ B\mathbf{u} := B_i(\mathbf{u}_k - \mathbf{u}_{k-1}),\ \ B_i = \begin{bmatrix} \nu_i \\ \mathbf{0} \end{bmatrix}, \tag{4}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the control with $n = \rho + \sigma$, $B \in \mathbb{R}^{m\times n}$. 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.

In particular, the first $\rho$ gain factors quantifies the contribution of the $i$-th trajectory parameters. The following $\sigma$ gain factors quantifies the contribution of the computations parameters. We elucidate how we derive these factors in the next two subsections, in Equation (6) and (8).

### A. Trajectory parameters energy contribution




(a) initial path (b) path replanned

Fig. 5. The alteration of the trajectory parameter $c_{1,1}$, the radius of the circle (it corresponds to the alteration of the path from Figure 1).

Equation (4) accounts for the energy due to the change of parameters $\mathbf{u}_k - \mathbf{u}_{k-1}$. For instance, when the trajectory $\varphi_1$ is a circle (see Figure 5), 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 ($\nu_{1,1}c_{1,1} > \nu_{1,1}c^-_{1,1}$) for a given $\nu_{1,1}$, that is then summed to the first coefficient $\alpha_0$ in Equation (3), shifting the modeled energy.

Let us assume that the set

$$\mathcal{P}_i := \{\mathbf{p}_k \mid \varphi_i(\mathbf{p}_k, c_{i,1}, \ldots, c_{i,\rho}) \in \mathcal{C}_i\}, \tag{5}$$

delimits the area where the $i$-th trajectory $\varphi_i$ is free to evolve using the trajectory parameters $c_{i,1}, ..., c_{i,\rho}$ (the gray area in Figure 5). $\varphi_i$ 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 parameters allows to change the trajectory. They are a way to alter the nominal trajectory in the initial plan and thus alter the energy. In fact, the algorithm uses the set from Equation (5) to find the trajectory parameters such that $\varphi_i$ has the highest instantaneous energy consumption, while still respecting the energy budget. In Figure 5, the parameter radius of the circle $c_{1,1}$ is replanned as, e.g., averse atmospheric conditions do not allow to terminate the plan.The sequence of these parameters (control sequence) forms the path $\mathcal{P}$. The gain factors for the trajectory parameters from Equation (4) are obtained

$$\begin{bmatrix} \nu_{i,1} \\ \vdots \\ \nu_{i,\rho} \end{bmatrix} = (\overline{c}_i - \underline{c}_i)^{-1}\rho^{-1} y_{\underline{c}_i} \begin{bmatrix} c_{i,1} \\ \vdots \\ c_{i,\rho} \end{bmatrix}, \tag{6}$$

where $y_{\underline{c}_i} \in \mathbb{R}$ is the maximum energy consumption of the highest possible parameters level at $i$-th stage. The value is obtained empirically (e.g., using an energy sensor). Whenever the trajectory parameters are not equally distributed, one can define $y_{\underline{c}_i}$ as a diagonal matrix where the items of the diagonal are the highest possible levels of specific trajectory parameters. Moreover, let $\nu_{i,1}, \ldots, \nu_{i,\rho}$ be zero when the parameters set $c_i^\rho = \{\emptyset\}$.

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

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

where $E$ specifies the rotation (it 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 $\dot{\mathbf{p}}$ due to the atmospheric interferences (wind $w \in \mathbb{R}$ in the top of Figure 5).

### B. Computations parameters energy contribution

Let us recall from Definition II.1 that the $i$-th stage $\Gamma_i$ of the plan $\Gamma$ 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].

For this purpose, 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 computation constraint sets and builds an energy model: a linear interpolation, one per each computation.

The computations are implemented by software components, e.g., ROS nodes in a ROS based system. The user implements these nodes such that they change the computational load with 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 $c_{i,\rho+1}$ might correspond to frames-per-second (fps) rate. The tool then changes the detection frequency.

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 constraint set.

Let us 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

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

where the matrix is a diagonal matrix of computations parameters. Moreover, let $\nu_{i,\rho+1}, \dots, \nu_{i,\rho+\sigma}$ be zero when the parameters set $c_i^{\sigma} = \{\emptyset\}$.

The factors thus add the computational energy component to the model in Equation (2).

## 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 $\Gamma$ and to guide the UAV on a valid path: to solve Problem II.1.

**Definition IV.1** (Valid control sequence and path)**.** Given a close strictly positive discrete interval $[j-n, j]$, a *control sequence* $\mathbf{u}$ is *valid* if for every stage $\Gamma_{i-1}$, $i \in [j-n, j]$ there exist a control $\mathbf{u}_k$ that produce the next stage $\Gamma_i$.

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

$$\begin{aligned} \mathcal{P} := \{\mathbf{p}_{k+1} \mid \exists \mathbf{u}_k, \mathbf{p}_k,& \\ &\langle \varphi_{i-1}(\mathbf{p}_k), \mathbf{u}_k \rangle \in \Gamma_{i-1} \\ &\implies \langle \varphi_i(\mathbf{p}_{k+1}), \mathbf{u}_{k+1} \rangle \in \Gamma_i\}, \end{aligned}$$

where for simplicity $\langle \varphi_i(\mathbf{p}_k), \mathbf{u}_k \rangle := \langle \varphi_i(\mathbf{p}_k, c_k^{\rho}), c_k^{\sigma} \rangle$.

Let us consider the assumption that at least one period in the user-defined initial plan is valid.

**Assumption IV.1.** Given an initial plan $\Gamma$, we assume that $\exists i \in [l/n-1]^+$, $j \in [n]^+$ s.t. $\Gamma_{(i-1)n+j}, \dots, \Gamma_{in+j}$ is valid.

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

**Theorem IV.2** (Valid periodic plan)**.** Suppose Assumption IV.1, Lemma III.1, and Proposition II.1 hold (a period of the plan is valid, its energy evolution is described by Equation (2), and the plan is periodic). Then the plan is valid and its energy evolution modeled by the periodic energy model in Section III-A.

*Proof.* The proof is based on mathematical induction. Base case: *

Induction step: *

■

### A. Output and control constraint sets

We stated earlier the output $y$–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 a UAV battery with a simplistic difference equation [25]

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

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.

**Definition IV.2** (Output, control constrain sets)**.** The output constrain set is then the set

$$\mathcal{Y}_k := \{y_k \mid y_k \in [0, \mathrm{SoC}_kQ_cV] \subseteq \mathbb{R}_{\geq 0}\},$$

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

The control constraint set is the trajectory contraint set for the trajectory parameters and computation constraint sets for the computation parameters (Definition II.1)

$$\mathcal{U}_k := \begin{cases} \mathcal{C}_i & \text{for } c_{i,j} \text{ with } j \leq \rho \\ \mathcal{S}_{i,j-\rho} & \text{for } c_{i,j} \text{ with } \rho < j \leq \sigma \end{cases}.$$

### B. Deployment algorithm

The algorithm first initializes the position, energy coefficients, and control (line 1).

The position is updated at line 7, using the expression from Equation (7) and the velocity $v \in \mathbb{R}_{\geq 0}$. The expression depends on the trajectory $\varphi_i$ from stage $\Gamma_i$. The algorithm iterates all the stages in the plan $\Gamma$ (line 2), and enters the next stage $\Gamma_{i+1}$ when the UAV reaches the triggering point $\mathbf{p}_{\Gamma_i}$.

**Algorithm** Energy-Aware Dynamic Planning

1: $\mathbf{p}_{k-1} \leftarrow \mathbf{p}_0, \mathbf{q}_{k-1} \leftarrow \mathbf{q}_0, \mathbf{u}_{k-1} \leftarrow \{\emptyset\}$
2: **for** $i \in [l]^+$ **do**
3: **while** $\mathbf{p}_{k-1} \neq \mathbf{p}_{\Gamma_i}$ **do**
4: $\mathbf{u}_k \leftarrow \arg\max_{\mathbf{u}} \sum_{j=k-1}^{k+N-2} l(\mathbf{q}_j, \mathbf{u}_j) + V_f(\mathbf{q}_{k+N-1})$
5: $\hat{\mathbf{q}}_k \leftarrow A\mathbf{q}_{k-1} + B_i(\mathbf{u}_k - \mathbf{u}_{k-1})$
6: $\mathbf{q}_k \leftarrow$ the estimate of the state from $\hat{\mathbf{q}}_k$ and sensor data
7: $\mathbf{p}_k \leftarrow \mathbf{p}_{k-1}\dot{\mathbf{p}}_d(\mathbf{p}_{k-1})/v$
8: $k \leftarrow k+1$
9: **end while**
10: **end for**

The energy coefficients are updated at line 5, using the expression from Equation (2). The a priori state estimate $\hat{\mathbf{q}}_k$ is refined using a state estimator–such as Kalman filter [37]–and the data from an energy sensor (line 6). At line 4, the algorithm uses robust output feedback model predictive control (MPC) [34] to select the control $\mathbf{u}_k$ for a given horizon $N \in \mathbb{Z}_{>0}$ from the cost function

$$\begin{aligned} l(\mathbf{q}_k, \mathbf{u}_k) &:= (1/2)(\mathbf{q}_k^T \operatorname{diag}(C)\mathbf{q}_k) + \\ &\qquad (1/2)(\mathbf{u}_k - \mathbf{u}_{k-1})^T B_i(\mathbf{u}_k - \mathbf{u}_{k-1}), \quad (9) \\ V_f(\mathbf{q}_k) &:= (1/2)(\mathbf{q}_k^T A\mathbf{q}_k), \end{aligned}$$

where $\operatorname{diag}(C)$ is a diagonal matrix with the items of the matrix $C$ from Equation (3) on the diagonal.

We note from Lemma III.1 that the expression in Equation (9) denotes the square of the predicted instantaneous energy consumption. Moreover, higher the horizon, higher the complexity and the accuracy of the control.

We further note that at every step of the summary on line 4 the algorithm has to evolve the state to check if the output satisfies the output constraint set, and if the control satisfies the control constraint set. In particular, at each steps it performs the subroutine

**while** $\overline{c}_i \notin \mathcal{U}_k$, $y_k \notin \mathcal{Y}_k$ **do**
$\overline{c}_i \leftarrow \overline{c}_i - \delta$
**end while**
$\mathbf{u}_k \leftarrow \overline{c}_i$

where to evaluate the inclusion in the output control set the algorithm evolves the state from its previous estimate similarly as at line 5 (we note that the condition can be written $\overline{c}_i \notin \mathcal{U}_k$, $C\mathbf{q}_k \notin \mathcal{Y}_k$).

We finally note that one can express the tradeoffs between trajectory and computation parameters (e.g., a decrement in the distance between the lines in a survey scenario is related to a decrement in the number of detections per second) enriching the control constraint set with the constraints

$$R_i\mathbf{u}_k - r_i \geq 0,$$

where $r_i \in \mathbb{R}^n$ and $R_i \in \mathbb{R}^{n\times n}$ expresses the relation between the parameters (if $R_i$ is the identity matrix, there is no relation between the parameters).

The algorithm can assess the validity of the plan using Theorem IV.2, but still find that there is no control available using the subroutine.

## V. Results

*

## VI. Conclusion and Future Work

*

## 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 omni-directional 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.

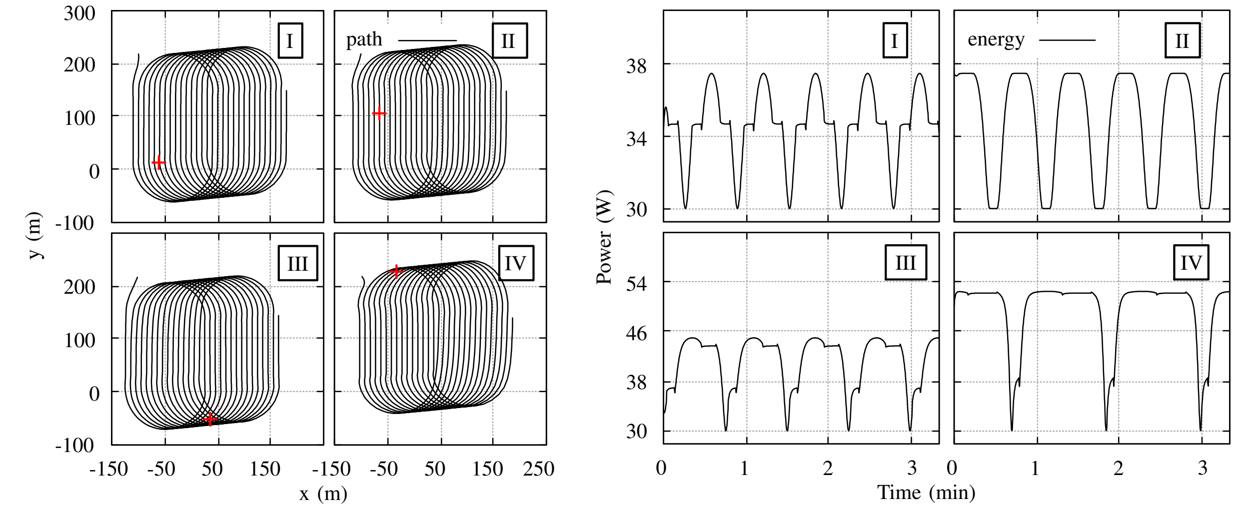
[9] S. T. H. Rizvi, G. Cabodi, D. Patti, and M. M. Gulzar, "A general-purpose graphics processing unit (gpgpu)-accelerated robotic controller using a low power mobile platform," *Journal of Low Power Electronics and Applications*, vol. 7, no. 2, p. 10, 2017.

[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.

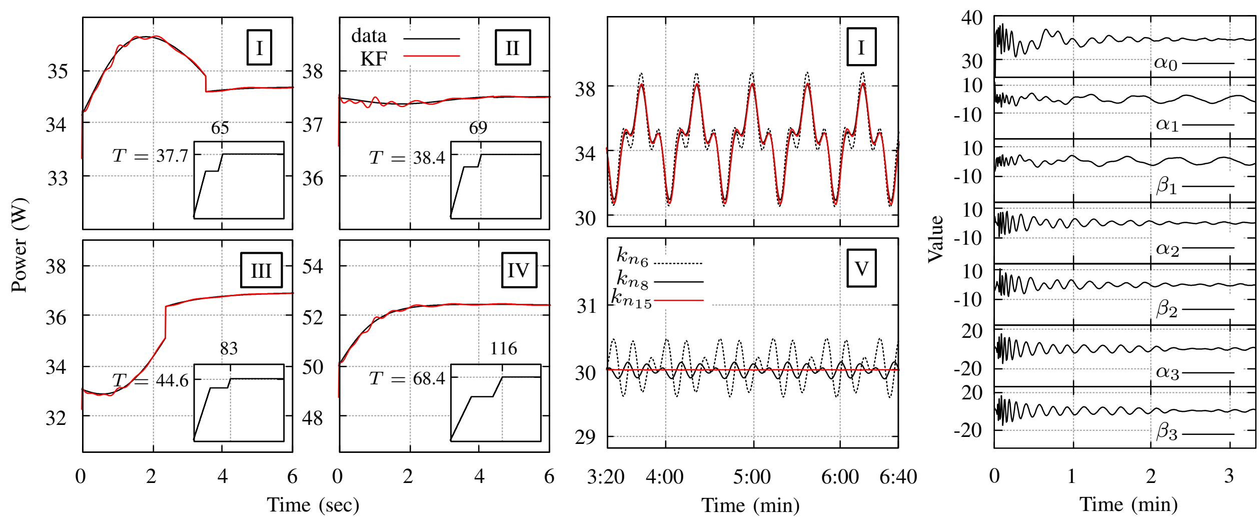
[13] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.




(a) path simulation, + is the UAV location after 200 seconds

(b) energy evolution simulation for 200 seconds interval

Fig. 6. Path and energy simulations with different atmospheric conditions. Simulations are performed from collected data by variations of wind speed and direction. The path and computations are static




(a) slice of the simulated energy along the estimated model

(b) model evolution at different $k$s

(c) path I coefficients evolution

Fig. 7. Evolution of the simulated energy data from Figure 6, using the estimation first in a, the data with no estimation predicting the future in b, and evolution of the estimated coefficients in c


[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 (3). 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 (1) 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]

$$
\begin{aligned}
h(t) = a_0/T + (1/T)\sum_{j=1}^{r} e^{i\omega j t}(a_j - i b_j) + \\
(1/T)\sum_{j=1}^{r} e^{-i\omega j t}(a_j + i b_j),
\end{aligned}
\tag{10}
$$

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 (2) 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 $D = \operatorname{diag}(\lambda_0, \lambda_1^0, \lambda_1^1, \ldots, \lambda_r^0, \lambda_r^1)$. $\lambda_0$ is the eigenvalue associated to the first item of $A$. $\lambda_j^0, \lambda_j^1$ 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 (2), under the assumptions made in the lemma (the control is a zero vector); $\dot{\mathbf{q}} = A\mathbf{q}$. The linear combination of the initial guess and the generic solution

$$
\begin{aligned}
F\mathbf{q}(0) &= \gamma_0 v_0 + \sum_{k=0}^{1}\sum_{j=1}^{r} \gamma_j v_j^k \\
F\mathbf{q}(t) &= \gamma_0 e^{\lambda_0 t} v_0 + \sum_{k=0}^{1}\sum_{j=1}^{r} \gamma_j e^{\lambda_j t} v_j^k
\end{aligned}
\tag{11}
$$

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

Let us consider the second expression in Equation (11). It represents the linear combination of all the coefficients of the state at time $t$. It can also be expressed in the following form

$$
\begin{aligned}
F\mathbf{q}(t)/T = \gamma_0 e^{\lambda_0 t} v_0/T + (1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^0 t} v_j^0 + \\
(1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^1 t} v_j^1.
\end{aligned}
\tag{12}
$$

We proof that the eigenvalues $\lambda$ and eigenvectors $V$ are such that Equation (12) is equivalent to Equation (10).

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.

Thereby we start by proofing that the first terms of the Equation (10) and (12) match. 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 $\gamma_0$ of the vector $\gamma$ 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 (10) and (12) 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 $\lambda_1^0$ is $v_1^0 = \begin{bmatrix} 0 & -i & 1 & 0 & \cdots & 0 \end{bmatrix}^T$. The eigenvector associated with the eigenvalue $\lambda_1^1$ is $v_1^1 = \begin{bmatrix} 0 & i & 1 & 0 & \cdots & 0 \end{bmatrix}^T$. Again, we find the values $\gamma_1$ of the vector $\gamma$ 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. In the output matrix, the frequency $1/T$ accounts for the period in Equation (10) and (12) and (1). At time instant zero, the coefficients $b_j$ are not present and the coefficients $a_j$ are doubled for each $j = 1, 2, \ldots, r$ (thus we multiply by a half the corresponding coefficients in $\mathbf{q}_0$). 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 signal and the output are equal, hence the lemma holds.

■

We note for practical reasons 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 $d \in \mathbb{R}$).

## APPENDIX II
## PLAN SPECIFICATION

### A. Format

The algorithm reads the plan $\Gamma$ from a file–the plan specification–with a specific file format.

The first stage $\Gamma_1$ 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 $\varphi_1$ the trajectory. 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 $\Gamma_2$. 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 $\varphi_1$.

The following $\sigma$ 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 $\varphi_1 \in \mathcal{C}_1$. It can alter the computations individually. In fact the $j$-th computation parameter has to be in $\mathcal{S}_{1,j}$.

Follows the entries for stage $\Gamma_2$ (equivalent to lines 1–4), $\Gamma_3$ 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 plan with an example of a survey scenario, equivalent to the one in Figure 1.

Recall that the plan is composed of a set of stages (Section II). It has a variable number of entries per each stage. The first line

```
1: 0.0003;0;(x+45)^2+(y-146)^2-4900+c1
```

corresponds to the trajectory $\varphi_1(\mathbf{p}_k, c_{1,1}) := (x+45)^2 + (y-146)^2 - 4900 + c_{1,1}$ in the initial stage $\Gamma_1$. The trajectory is a circle. The gain is $k_e = 3 \cdot 10^{-4}$, and the rotation is counter-clockwise–we use the rotation matrix $E$ from Equation (7). We note that line 1 further specifies the parameter `c1` of the radius of the circle. It corresponds to $c_{1,1}$ in Definition II.1.

The following line

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

specifies the triggering point $\mathbf{p}_{\Gamma_1}$. `sqrt` is the square root. The UAV reaches the triggering point within a given $\varepsilon$ (the tolerance), and the algorithm switches to the next stage $\Gamma_2$. We note that the triggering point is in function of the parameter `c1`. This is necessary; an change in a parameter in one stage affects the overall flight.

Let us assume the scenario has two computations. One computation is the CNN object detection algorithm, and it can be planned by one parameter $c_{1,2}$, the fps rate. Another computation is the variable key-size encryption algorithm. It can be planned by one parameter $c_{1,2}$, the key-size. Then in the plan

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

line 3 corresponds to the trajectory $\varphi_1$'s contraints set $\mathcal{C}_1 = [-3 \cdot 10^3, 0]$. The algorithm selects $c_{1,1} \in \mathcal{C}_1$. Line 4 corresponds to the computation constraints set $\mathcal{S}_{1,1}$ for the computation parameter $c_{1,2}$. Line 4 corresponds to the computation constraints set $\mathcal{S}_{1,2}$ for the computation parameter $c_{1,2}$. Lines 3–5 must follow the ascending order of the parameters. Lines 1–5 all describe the stage $\Gamma_1$.

Once the UAV reaches the triggering point (line 2) and there are no further lines, the algorithm terminates (the last triggering point is the final point). If there are further lines, the stage switches to $\Gamma_2$

```
6: 0.05;1;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 the trajectory $\varphi_2(\mathbf{p}_k, c_{1,1}) := x + \sqrt{4900 + c_{1,1}} + 45$, 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 $\varphi_1$ 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. 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 5; 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 $\Gamma_2$ at line 7 also depends on the parameter `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 $\Gamma_3$

```
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 the trajectory $\varphi_3(\mathbf{p}_k, c_{1,1}) := (x + \sqrt{4900 + c_{1,1}} - 30)^2 + (y-11)^2 - 5625$, a circle of fixed size that changes the coordinates in function of the parameter **c1**. The rotation is counter-clockwise, alike $\varphi_1$.

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]
```

defines the stages $\Gamma_4$ and $\Gamma_5$. Line 16 corresponds to the trajectory $\varphi_4(\mathbf{p}_k, c_{1,1}) := x + \sqrt{4900 + c_{1,1}} - 105$. Line 21 corresponds to the trajectory $\varphi_4(\mathbf{p}_k, c_{0,1}) := (x - \sqrt{4900 + c_{0,1}} + 105)^2 + (y - 147)^2 - 4900 + c_{0,1}$.

We note that the survey scenario example changes the radius of the circular trajectory $\varphi_1$ through the parameter $c_{1,1}$. It also changes the radius of $\varphi_5$ in the same way (recall Assumption II.1). The other trajectories ($\varphi_2, \varphi_3, \varphi_4$) displacement alters the distance between the lines in the survey. For simplicity, we suppose line 22 describes the final point. We further suppose the user does not desire to process any computation in the last stage $\Gamma_5$. To inhibit the computations

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

If there are any following stages in the survey, they are constructed the same way. The stages $\Gamma_i$ for $i := \{0, 4, 8, \dots\}$ contain trajectory circle with variable radius. For $i := \{1, 3, 5, \dots\}$ trajectory line with variable displacement. For $i := \{2, 6, 10, \dots\}$ trajectory circle with fixed radius and variable displacement.