Energy-Aware Dynamic Planning Algorithm for Autonomous UAVs

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

***Abstract*— The paper presents a planning algorithm for autonomous UAVs with energy constraints and solves the problem of dynamic planning of the path and computations simultaneously. The planning strategy is based on a periodic energy model that is empirically motivated and formally proved. The model is enhanced with the time and energy contribution of the path and computations. The algorithm replans both in function of the battery state accounting for uncertainty. The dynamic planning allows to exploit all the available resources, yet avoid possible in-flight failure in case of unexpected battery drops due to, e.g., adverse atmospheric conditions.**

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 following a path and performing some on-board 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.

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.

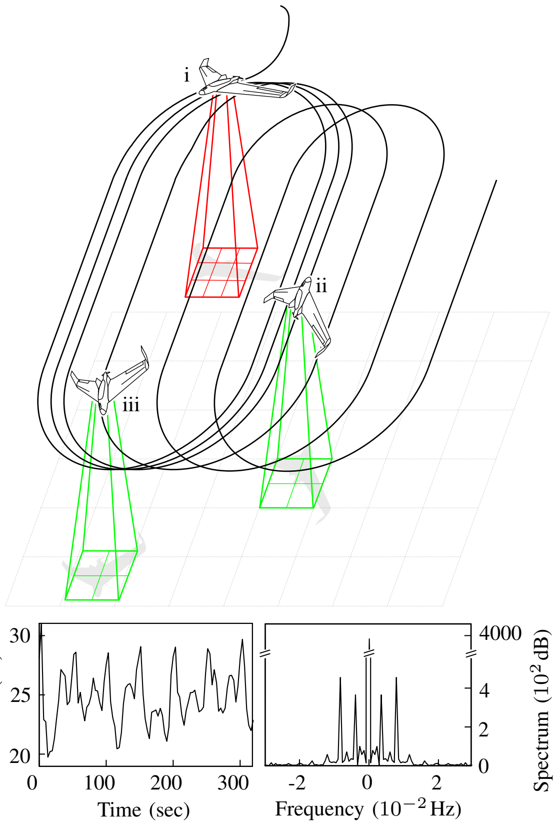

Fig. 1. Dynamic replanning of an agricultural scenario in terms of the path and computations simultaneously. An initial plan (in i) is adapted computation-wise (in ii) and path-wise (in iii). In the bottom left, the collected energy data of a physical UAV (Opterra fixed-wing drone) flying the scenario is shown along its power spectrum.

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 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 the path and computations simultaneously. To model the path we use multiple mathematical functions. In Figure 1, the path contains multiple circles and lines. To model the computations we use `powprofiler`, a profiling tool presented in previous work [18]. To guide the UAV we use a vector field [19] that converges to the path. 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 prove 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 paths (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. Path-wise, the UAV flies in circles and lines covering a polygon. Computation-wise, it detects obstacles using a neural network and notifies grounded mobile robots employed for, e.g., 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 such plan.

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 in Appendixes.

II. Planning Overview

The algorithm inputs a user-specified initial plan that consist of different stages. At 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 path 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 path (ii has a shorter distance between the lines than iii), and the other one to the computations (i processes more images per second than ii).

The algorithm outputs the control (the parameters) using output model predictive control (MPC) [34] where it checks the satisfaction of the battery constraints. 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 configuration 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 $\{0, 1, \ldots, a\}$, $[a]^+$ the set $[a]/\{0\}$. Bold lower-case letters indicates vectors. $c_{i,j}$ the j-th parameter of the i-th parameters set c_i. $\underline{c}_{i,j}, \overline{c}_{i,j}$ are the lower and upper bounds of the parameter $c_{i,j}$.

Let us assume that the path at stage i can be altered with ρ path parameters $c_i^\rho := \{c_{i,1}, c_{i,2}, \ldots, c_{i,\rho}\}$, and the computations with σ computation parameters $c_i^\sigma := \{c_{i,\rho+1}, c_{i,\rho+2}, \ldots, c_{i,\rho+\sigma}\}$. We then express the path as a continuous twice differentiable function $\varphi_i : \mathbb{R}^2 \times \mathbb{R}^\rho \to \mathbb{R}$ of a point and the path 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 computation parameters. We discuss the concrete meaning of the value of path parameters in Subsection III-A, and computation parameters in Subsection III-B.

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

$$\Gamma_i := \{\varphi_i(\mathbf{p}_k, c_i^\rho), c_i^\sigma \mid \exists\, \mathbf{p}_k,\, \varphi_i(\mathbf{p}_k, c_i^\rho) \in \mathcal{C}_i, \\ \forall j \in [\sigma]^+,\, c_{i,\rho+j} \in \mathcal{S}_{i,j}\,\},$$

where $\mathcal{C}_i := [\underline{c}_i, \overline{c}_i] \subseteq \mathbb{R}$ is the path 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 altitude $h \in \mathbb{R}_{>0}$[1] w.r.t. some inertial navigation frame $\mathcal{O}_W$. In Figure 2, $\varphi_1, \ldots, \varphi_6$ are paths. φ_1 and φ_5 are circles, while φ_2, φ_4, and φ_6 are lines. They are all relative to different stages $\Gamma_1, \Gamma_2, \ldots$. The constraints set $\mathcal{C}_1, \mathcal{C}_2, \ldots$ forms the area where the paths $\varphi_1, \varphi_2, \ldots$ can be altered with the parameters $c_{i,1}, \ldots, c_{i,\rho}$ (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 Γ_4 is bounded by $\underline{c}_4, \overline{c}_4$).

[1]The altitude might change at different flying phases and under different atmospheric conditions

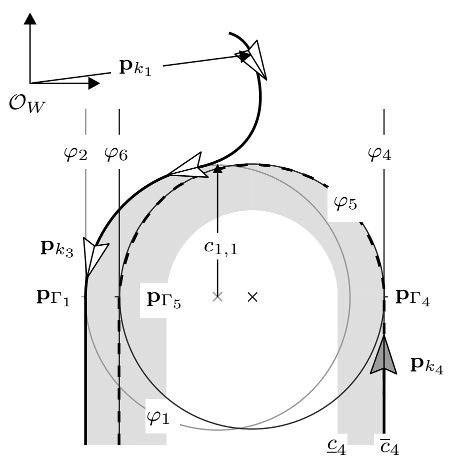

Fig. 2. Plan definition notation shown on an initial slice of the plan in Figure 1.

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

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

The point $\mathbf{p}_{\Gamma_i}$ that allows the transition between Γ_i and Γ_{i+1} is called *triggering point*. In Figure 2, $\mathbf{p}_{\Gamma_1}$ allows the transition between Γ_1 and Γ_2, $\mathbf{p}_{\Gamma_4}$ between Γ_4 and Γ_5, and $\mathbf{p}_{\Gamma_5}$ between Γ_5 and Γ_6. The last triggering point $\mathbf{p}_{\Gamma_l}$ relative to the last stage Γ_l is called *final point*.

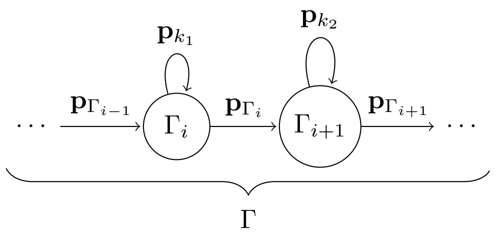

Fig. 3. The plan defined as a FSM

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 Γ_i. The UAV remains in the stage with any generic point $\mathbf{p}_{k_1}$. It eventually enters the stage Γ_{i+1} with the triggering point $\mathbf{p}_{\Gamma_i}$ and so on, until it reaches the final point.

B. Problem formulation

In order to simplify the problem formulation, we consider some primitive paths. All the other paths are build from these paths with a shift $\mathbf{d} := (x_d, y_d)$.

Given $n \in \mathbb{Z}_{>0}$ ($n < l, l/n \in \mathbb{Z}$) primitive paths $\varphi_1, \dots \varphi_n$, a generic starting point $\mathbf{p}$ and the current levels of the path parameters c_1^ρ, all the other paths $\varphi_{n+1}, \dots, \varphi_l$ are build

$$\begin{aligned} &\varphi_{(i-1)n+j}(\mathbf{p} + (i-1)\mathbf{d}, c_1^\rho) - \\ &\varphi_{in+j}(\mathbf{p} + i\mathbf{d}, c_1^\rho) = e_j, \end{aligned} \tag{1}$$

$\forall i \in [l/n - 1]^+, j \in [n]^+$, where $e_j \in \mathbb{R}$ is the j-th constant difference.

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

The algorithm measures the time between the paths and assumes the initial period is one. The periods might be different for different js due to atmospheric interferences.

One can define the plan using primitive paths or define all the stages explicitly and find n searching the value which satisfies the Equation (1). If there is no such value, (e.g., when the plan is composed of only one path), the period T from Definition II.2 can be determined empirically from energy data (such as these shown in Figure 1).

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

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 prove 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 non-periodic signals.

After having illustrated the energy model, we enhance it with the energy contribution of the path in Subsection III-A, and of the computations in Subsection III-B.

Let us consider a periodic energy 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

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

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 by the output of a linear model

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

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

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

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

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 (*).

The linear model in Equation (3) allows us to include the control in the model of Equation (2).

Lemma III.1 (Signal, output equality)**.** Suppose control $\mathbf{u}$ is a zero vector, 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 output y in Equation (3).

Proof. The equality of the signal and output is achieved by a proper choice of the 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 suppose that at time instant k the plan reached the i-th stage Γ_i and the control

$$\mathbf{u}_k = \begin{bmatrix} c_k^\rho & c_k^\sigma \end{bmatrix}^T, \tag{5}$$

where $\mathbf{u}_k \in \mathbb{R}^n$ with $n = \rho+\sigma$ differs from the nominal control $\mathbf{u}$ in Equation (3). We include the control in the nominal control exploiting the following observation.

We observe that a change in path parameters affects the energy indirectly. It alters the time when the UAV reaches the final point $\mathbf{p}_{\Gamma_l}$. We use this information later in the algorithm to check that the battery discharge time is greater and replan the path parameters accordingly. A change in computation parameters affects the energy directly. It alters the instantaneous energy consumption as more computations require more power (and vice versa). We replan the computation parameters to maximize the instantaneous energy consumption against the maximum battery discharge rate.

The nominal control and the input matrix

$$\begin{aligned}\mathbf{u} &= (\hat{\mathbf{u}}_k - \hat{\mathbf{u}}_{k-1}),\ \ \hat{\mathbf{u}}_k := \mathrm{diag}(\nu_i)\mathbf{u}_k + \tau_i,\\ B &= \begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & 1 \\ & & & * & & \end{bmatrix}.\end{aligned} \tag{6}$$

The matrix $B \in \mathbb{R}^{m\times n}$ contains zeros (*) except the first row where the first ρ columns are still zeros and the remaining σ are ones. $\hat{\mathbf{u}}_k$ is a scale transformation with $\nu_i = \begin{bmatrix}\nu_i^\rho & \nu_i^\sigma\end{bmatrix}^T$ and $\tau_i = \begin{bmatrix}\tau_i^\rho & \tau_i^\sigma\end{bmatrix}^T$ scaling factors quantifying the contribution to the plan of a given parameter in terms of time for the first ρ parameters, and power for the remaining σ (we use the same notation for the path and computation scaling factors as for the parameters). The nominal control $\mathbf{u}$ is then the difference of these contributions of two consecutive controls $\mathbf{u}_{k-1}, \mathbf{u}_k$ applied to the system. $B\mathbf{u}$ merely includes the difference in power into the model in Equation (3).

We clarify how we derive the factors ν_i, τ_i in the next two subsections.

A. Path parameters energy contribution

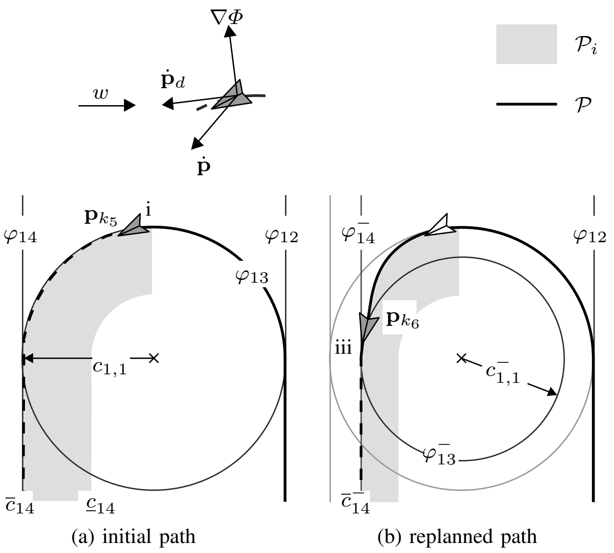

Fig. 4. The alteration of the path parameter $c_{1,1}$, the radius of the circle (it corresponds to the alteration of the path in Figure 1).

Let us assume that the set

$$\mathcal{P}_i := \{\mathbf{p}_k \mid \varphi_i(\mathbf{p}_k, c_{i,1}^\rho) \in \mathcal{C}_i\}, \tag{7}$$

delimits the area where the i-th path φ_i is free to evolve using the path parameters c_i^ρ (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 path. Physically, this means the UAV is flying exactly over the nominal trajectory. The path parameters allows to change the path. They are a way to alter the nominal trajectory in the initial plan and thus alter the energy by changing the flying time in the example in Figure 1. In fact, the algorithm uses the set from Equation (7) to find the path parameters such that the plan consisting of flying φ_i has the highest energy, while still respecting the constraints. In Figure 4, the parameter radius of the circle $c_{1,1}$ is replanned as, e.g., averse atmospheric conditions do not allow to terminate the plan.

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]

$$\dot{\mathbf{p}}_d(\mathbf{p}_k) := E\nabla\varphi_i - k_e\varphi_i\nabla\varphi_i, \;\; E = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad (8)$$

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 4).

The scaling factors for the path parameters from Equation (6) are derived empirically. For the example in Figure 4, we can obtain the scaling factor $\nu_{1,1}$ measuring the time needed to compute the path with the lowest configuration $\underline{c}_1$, $\underline{t}$ and the highest $\overline{t}$. The variation of the control hence results in an approximate measure of the plans' time variation with factors

$$\begin{aligned} \nu_{i,j} &= \left((\overline{t} - \underline{t})/(\overline{c}_{i,j} - \underline{c}_{i,j})\right)/\rho, \\ \tau_{i,j} &= \left(\underline{c}_{i,j}(\underline{t} - \overline{t})/(\overline{c}_{i,j} - \underline{c}_{i,j}) + \underline{t}\right)/\rho, \end{aligned} \quad (9)$$

$\forall j \in [\rho]^+$. Moreover, let the factors be zero when the parameters set $c_i^\rho = \{\emptyset\}$.

B. Computation parameters energy contribution

Let us recall from Definition II.1 that the i-th stage Γ_i of the plan Γ contains the computation parameters which characterize the computations. We estimate 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. Our tool measures the instantaneous energy consumption of a subset of possible computation 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., Robot Operating System (ROS) nodes in a ROS-based system [36]. The user implements these nodes such that they change the computational load according to node-specific ROS parameters–the computation parameters. In a generic software component system, the user maps the computational load to the arguments [35]. In both cases, with ROS [37] or with generic software components system [35], the tool performs automatic modeling. For instance, if the computation is an object detector, a computation parameter $c_{1,2}$ might correspond to frames-per-second (fps) rate. The tool then measures power according to the detection frequency.

We note that while the path 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 scaling factors add the computational energy component to the model in Equation (3). They are derived similarly to Equation (9)

$$\begin{aligned} \nu_{i,j} &= (g(\overline{c}_{i,j}) - g(\underline{c}_{i,j}))/(\overline{c}_{i,j} - \underline{c}_{i,j}), \\ \tau_{i,j} &= \underline{c}_{i,j}(g(\underline{c}_{i,j}) - g(\overline{c}_{i,j}))/(\overline{c}_{i,j} - \underline{c}_{i,j}) + g(\underline{c}_{i,j}), \end{aligned}$$

$\forall j \in [\rho+1, \rho+\sigma]$. Moreover, let the factors be zero when the parameters set $c_i^\sigma = \{\emptyset\}$.

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 guide the UAV on the path resulting from such sequence–to solve Problem II.1.

A valid control sequence has to respect the energy constraints. We consider some realistic constraints to the energy of a flying UAV in the following subsection.

A. Output and control constraint sets

We stated earlier the output y–the instantaneous energy consumption–evolves in $\mathbb{R}$. 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) b of a UAV battery with a simplistic difference equation [25]

$$b_k = b_{k-1} - k_b\left(V - \sqrt{V^2 - 4R_r y_k}\right)/(2R_rQ_c), \quad (10)$$

where k_b is the battery coefficient determined experimentally, $V \in \mathbb{R}$ is the internal battery voltage measured in volts, $R_r \in \mathbb{R}$ the resistance measured in ohms, and $Q_c \in \mathbb{R}$ the constant nominal capacity measured in amperes per hour.

Definition IV.1 (Output, control constrain sets)**.** The output constrain set is then the set

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

and b_kQ_cV is the maximum instantaneous energy consumption.

Algorithm Energy-Aware Dynamic Planning

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

The control constraint set is the path constraint set for the path 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 2). It updates the position at line 8, using the expression from Equation (8) and the velocity $v \in \mathbb{R}_{\geq 0}$. The expression depends on the path φ_i from stage Γ_i. The algorithm iterates all the stages in the plan Γ (line 3), and enters the next stage Γ_{i+1} when the UAV reaches the triggering point $\mathbf{p}_{\Gamma_i}$ (FSM in Definition II.1).

The energy coefficients are updated at line 6, using the expression from Equation (3). A priori state estimate $\hat{\mathbf{q}}_k$ is refined using a state estimator–such as Kalman filter (KF) [38]–and the data from an energy sensor (line 7). At line 5, the algorithm uses MPC to select the control $\mathbf{u}_k$ for a given horizon $N \in \mathbb{Z}_{>0}$ from the cost function (higher the horizon, higher the complexity and the robustness of the control to the output constraints)

$$V_f(\mathbf{q}_k) := (1/2)\mathbf{q}_k^T \operatorname{diag}(C)\mathbf{q}_k,$$
$$l(\mathbf{q}_k, \mathbf{u}_k) := (1/2)(\mathbf{q}_k + B\mathbf{u})^T \operatorname{diag}(C)(\mathbf{q}_k + B\mathbf{u}),$$

where $\operatorname{diag}(C)$ is a diagonal matrix with the items of C from Equation (4), and B, $\mathbf{u}$ are defined in Equation (6).

We note that at every step of the sum on line 5, the algorithm evolves the state to check if the output satisfies the output constraint set, and if the control satisfies the control constraint set. In particular, it performs a subroutine

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

where $\delta \in \mathbb{R}^\rho \times \mathbb{Z}^\sigma_{\geq 0}$ are reduction steps. The maximum values of path and computation parameters are reduced by the steps if they don't meet the constraints (or increased if with $\bar{c}_i + \delta$ the constraints are still respected).

We finally note that one can express the tradeoffs between 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).

V. Results

In this section, we discuss our experimental setup, the implementation strategy, and some results.

A. Experimental setup

The algorithm that we presented in this paper is motivated by a periodic behavior of empirical data on energy consumption (see the subfigures of Figure 1). We collected these data flying Opterra, a fixed-wing UAV that we adapted for an agricultural scenario. The UAV was flying in standard atmospheric conditions like the path 5.I in Figure 5, or the first "non-adapted" part of Figure 1. We later extended the UAV to carry a companion computer, NVIDIA Jetson Nano [39], running ROS. The companion computer has two ROS nodes; one detects obstacles using PedNet, a Fully Convolutional Neural Network [40], and the other communicates with a ground station.

We implemented a realistic simulator to simulate the empirical data of a given plan and atmospheric conditions. We show the effects of different conditions in Figure 5. Both paths 5.I and 5.II are flying the same plan with different initial parameters values. Path 5.I is flying with a constant wind speed of 5 meters per second, wind direction of 0 degrees, and an initial path parameter $c_{1,1}$ value of 0. Path 5.II is flying under the same conditions but a wind direction of 90 degrees and the initial path parameter value of -1000. The energy evolutions of these two paths are the top two subfigures 7.I and 7.II of Figure 7. The simulated energy data are in black, the model evolution in red.

B. Algorithm evaluation

Periodic energy model: We showcase the energy model from Section III in Figure 6. Left figures illustrate an initial slice of the model and period estimation (Definition II.2). We used order r equal to 3. We motivate this choice again with Figure 1, where the power spectrum

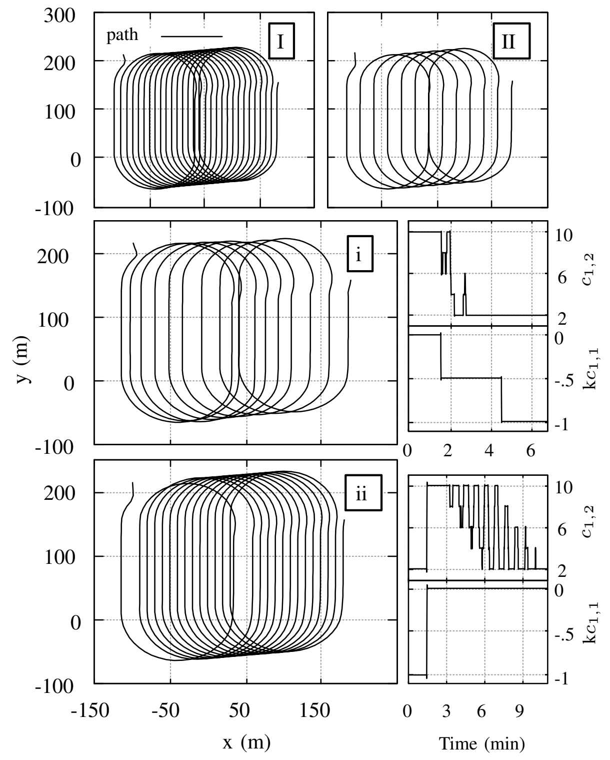

Fig. 5. Path simulations with variations of wind speed and direction. In 5.I and 5.II the path is static. It is dynamically replanned with the algorithm in 5.i and 5.ii. The algorithm adapts path parameter radius of the circle $c_{1,1}$ and computation parameter fps rate $c_{1,2}$.

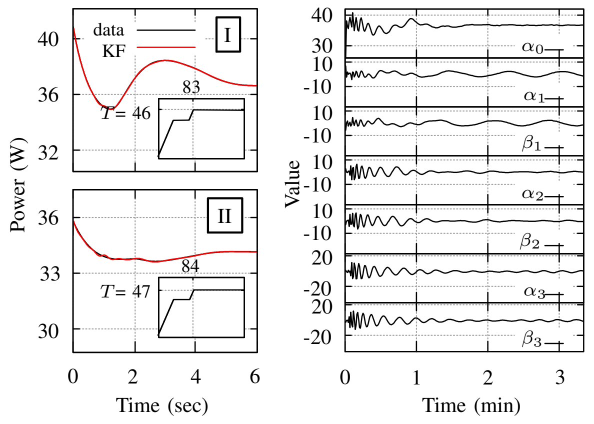

Fig. 6. Energy estimation for the first 6 seconds on the left side, the evolution of the state $\mathbf{q}$ on the right.

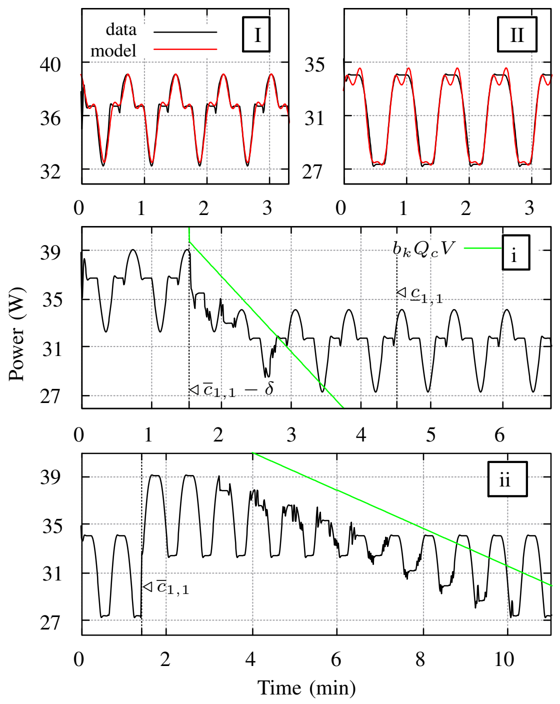

Fig. 7. The energy models of the paths from Figure 5 for 200 seconds against the simulated data (7.I and 7.II). Below are the energy evolutions from the algorithm (7.i and 7.ii). It replans the path when the final time and battery time do not match, and the computation when the battery is discharged.

subfigure shows that 3 frequencies are adequate. On the right of Figure 6, we show the states $\alpha_0, ..., \beta_3$ in time, concluding that approximately 2 periods are sufficient to obtain a consistent state estimate. With non-periodic signals, we observed that the estimator estimates primarily the first state α_0 and it neglects the others. It hence approximates the non-periodicity with a linear model.

Implementation of the planning strategy: The practical implementation is based on observations of different variations of paths and computations. A variation of path alters the overall flying time, which we reflect in the factors $\nu_{1,1}, \tau_{1,1}$ from Equation (9). We compare the remaining flight time with the time needed to completely deplete the battery from Equation (10). We reduce or increase the parameter to optimize the battery time. The path parameter $c_{1,1}$ is equal for all the stages and it changes the radius of the first circle (the change is illustrated in Figure 4) in the current period and therefore shifts the other paths accordingly. It results in a shorter or longer distance between the survey lines and in an increment or reduction of the flying time respectively. The path constraint set is set to $\underline{c}_{1,1}$ = -1000 and $\overline{c}_{1,1} =$ 0 equal for all the stages.

A variation of computations affects directly the power. We thus select the highest computation which satisfies the constraints from Definition IV.1 in the line 5 of the algorithm. We observed a low effect on the power of the communication ROS node. Nevertheless, the detection node varies between 5 and 10 watts for the lowest and highest fps. We implemented fps rate parameter $c_{1,2}$ with factors $\nu_{1,2}, \tau_{1,2}$ mapping $c_{1,2}$ to the data from `powprofiler`. The computation constraint set is set to $\underline{c}_{1,2} = 2$ and $\overline{c}_{1,2} = 10$ equal for all the stages.

Dynamic adaptation of the path and computation parameters: We have tested the validity of the algorithm showing the dynamic adaptation in the subfigures 5.i and 5.ii of Figure 5 path-wise, and 7.i and 7.ii of Figure 7 energy-wise. For the first path (Subfigure 5.I) the plan starts at the highest configuration of parameters. We simulated two unexpected battery drops at approximately 1 minute and a half and 4 minutes and a half. The algorithm optimizes the path in the proximity of the drops to ensure that the flight is completed. Moreover, it maximizes the parameter $c_{1,2}$ when the battery is discharging (green line) respecting the output constraint (Definition IV.1). We simulated the opposite scenario for the second path (Subfigure 5.II). The plan starts at the lowest configuration of parameters but the battery behaves linearly. We note that the path parameter is increased as soon as the algorithm estimated enough data (two periods T) and the computation parameter is optimized for the battery discharge rate. For both cases, we used reductions δ of 500 and 2 for $c_{1,1}$ and $c_{1,2}$ respectively, and the horizon N equal to the period T.

VI. Conclusion and Future Work

In this paper, we presented a planning algorithm for autonomous UAVs with a limited power source. The algorithm plans the path with computations dynamically and simultaneously. We collected empirical data of a fixed-wing UAV flying an agricultural scenario and built a simulator to test the algorithm in different conditions. The algorithm estimates the energy with a KF and optimizes the path and computations using MPC. It evolves the state to verify the satisfaction of battery and time constraints. It uses a plan where at each stage the UAV flies a path and does some computations. It guides the UAV using a vector field. Finally, it exploits the computational energy model varying the level of computations in the function of the current battery level.

We are currently extending the results of this paper to work with a standard flight controller. We are further investigating the validity of the algorithm on other energy-critical mobile robots. Such as on planetary exploration rovers in [41].

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.

[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://adamseewald.cc/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://adamseewald.cc/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://adamseewald.cc/short/component2019

[36] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[37] 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://adamseewald.cc/short/energy2020

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

[39] NVIDIA. NVIDIA Jetson Nano developer kit. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[40] M. Ullah, A. Mohammed, and F. Alaya Cheikh, "Pednet: A spatio-temporal deep convolutional neural network for pedestrian segmentation," *Journal of Imaging*, vol. 4, no. 9, p. 107, 2018.

[41] A. Seewald, "Beyond traditional energy planning: the weight of computations in planetary exploration," in *IROS Workshop on Planetary Exploration Robots: Challenges and Opportunities (PLANROBO20)*. ETH Zurich, Department of Mechanical and Process Engineering, 2020, p. 3. [Online]. Available: https://adamseewald.cc/short/beyond2020

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

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

[44] 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 $a_0, \dots, b_r$ are the same as the coefficients of the state $\alpha_0, \dots, \beta_r$.

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 [42]

$$h(t) = a_0/T + (1/T) \sum_{j=1}^{r} e^{i\omega j t}(a_j - ib_j) + (1/T) \sum_{j=1}^{r} e^{-i\omega j t}(a_j + ib_j), \tag{11}$$

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 [42], [43]. To solve the matrix exponential e^{At}, we use the eigenvectors matrix decomposition method [44].

The method works on the similarity transformation $A = VDV^{-1}$. The power series definition of e^{At} implies $e^{At} = Ve^{Dt}V^{-1}$ [44]. We consider the non-singular matrix V, whose columns are eigenvectors of A; $V := \begin{bmatrix} v_0 & v_1^0 & v_1^1 & \dots & v_r^0 & v_r^1 \end{bmatrix}$. We then consider the diagonal matrix of eigenvalues $D = \mathrm{diag}(\lambda_0, \lambda_1^0, \lambda_1^1, \dots, \lambda_r^0, \lambda_r^1)$. 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 $Av_j = \lambda_j v_j \ \forall j = \{1, \dots, m\}$, and $AV = VD$.

We apply the approach in terms of Equation (3), 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

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

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

We proof that the eigenvalues λ and eigenvectors V are such that Equation (13) is equivalent to Equation (11).

Let us consider the second expression in Equation (12). It represents the linear combination of all the coefficients of the state at time t. It can also be expressed in the following form

$$F\mathbf{q}(t)/T = \gamma_0 e^{\lambda_0 t} v_0/T + (1/T) \sum_{j=1}^{r} \gamma_j e^{\lambda_j^0 t} v_j^0 + (1/T) \sum_{j=1}^{r} \gamma_j e^{\lambda_j^1 t} v_j^1. \tag{13}$$

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

Thereby we start by proofing that the first terms of the Equation (11) and (13) 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 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 (11) and (13) 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

$$\begin{cases} e^{i\omega t}(a_1 - ib_1) &= \gamma_1 e^{i\omega t} v_1^0 \\ e^{-i\omega t}(a_1 + ib_1) &= \gamma_1 e^{i\omega t} v_1^1 \end{cases}$$

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 (11) and (13) and (2). At time instant zero, the coefficients b_j are not present and the coefficients a_j are doubled for each $j = 1, 2, \dots, r$ (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.

$\blacksquare$

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 $d \in \mathbb{R}$).