Energy-Aware Dynamic Mission Planning Algorithm for UAVs

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

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

I. Introduction

Many scenarios involving unmanned aerial vehicles (UAVs) such as precision agriculture, search and rescue, and surveillance require high autonomy but have limited energy budgets. A typical example of a mission in these scenarios is a UAV flying a trajectory and performing some on-board computations with little human interaction. For instance, the UAV might detect ground patterns and inform other ground-based actors. Energy-aware dynamic mission planning finds optimal tradeoffs between the trajectory, computations, and energy requirements. However, current generic planning solutions for UAVs flying outdoors do not plan the mission dynamically, nor are energy-aware. They are often semi-autonomous; the mission is static and usually defined using mission planning software [1]. Such a state of practice has prompted us to propose an *energy-aware dynamic mission 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 mission planning problem.

Planning algorithms literature for mobile robots is by now large and related to topics such as trajectory generation and path planning. Generally, the algorithms select an energy-optimized trajectory [2], e.g., by maximizing the operational time [3]. However, they apply to a small number of robots [4], and focus on planning the trajectory for these robots [5], despite compelling evidence for the energy consumption also being influenced by computations [6]. Given the availability of powerful GPU-equipped mobile hardware, the use of computations is expected to increase in the near future. Mission planning algorithms, which include a broader concept of a mission being a set of tasks along with a motion plan, all focus on the trajectory [6], [7] and apply to a small number of robots [8], [9]. For UAVs specifically, rotorcrafts have gained interest in terms of algorithms for energy-optimized trajectory generation [10], [11].

Our goal is to complete the mission with the highest possible computations and adjustments as the UAV flies and its batteries drain. The algorithm plans both tasks and trajectory unlike most of the past planning algorithms literature. It guides the UAV using a vector field [12] that converges smoothly to the planned trajectory. The vector field for planning is widely discussed in the literature [13]–[17]. The user specifies an initial mission plan–the trajectory and the tasks–along with an energy budget and some parameters: the Quality of Service (QoS) computations are relative to the tasks. The trajectory-explicit equations (TEEs) adjustments are relative to the path. The computations specify a given task's computational level. The adjustments are a way to alter the TEE. The algorithm replans the initial mission plan as the energy budget changes due to atmospheric interferences. In the remainder of the paper, we adopt the following notation. We refer to the values of mission-specific QoS and TEEs parameters as computations and adjustments, we refer to the constraints sets that delimit such computations and adjustments as QoS and TEEs sets, and we refer to the current trajectory as TEE.

The algorithm contributes to the planning literature further with a periodic energy model that accounts for the uncertainty (atmospheric interference). Mission periodicity is usually present due to the mission's repetitive patterns [18]. Indeed UAV scenarios often iterate over a set of tasks and trajectories (e.g., monitoring or search and rescue). The mission is periodic, so we expect the energy to approximately evolve periodically. The observation applies broadly; repetitive patterns have been observed in scenarios as far as space rovers [19]. The algorithm addresses the periodicity through Fourier analysis. The uncertainty with a state estimator. It outputs the control input (adjustments and computations) using output model predictive control (MPC).

In the spirit of reducing costs and resources, we showcase the algorithm using the problem of dynamic mission planning for a precision agriculture fixed-wing UAV. Precision agriculture is often put into practice [20] with ground mobile robots used for harvesting [21]–[26], and UAVs for preventing damage and ensuring better crop quality [1], [27]. The mission

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.

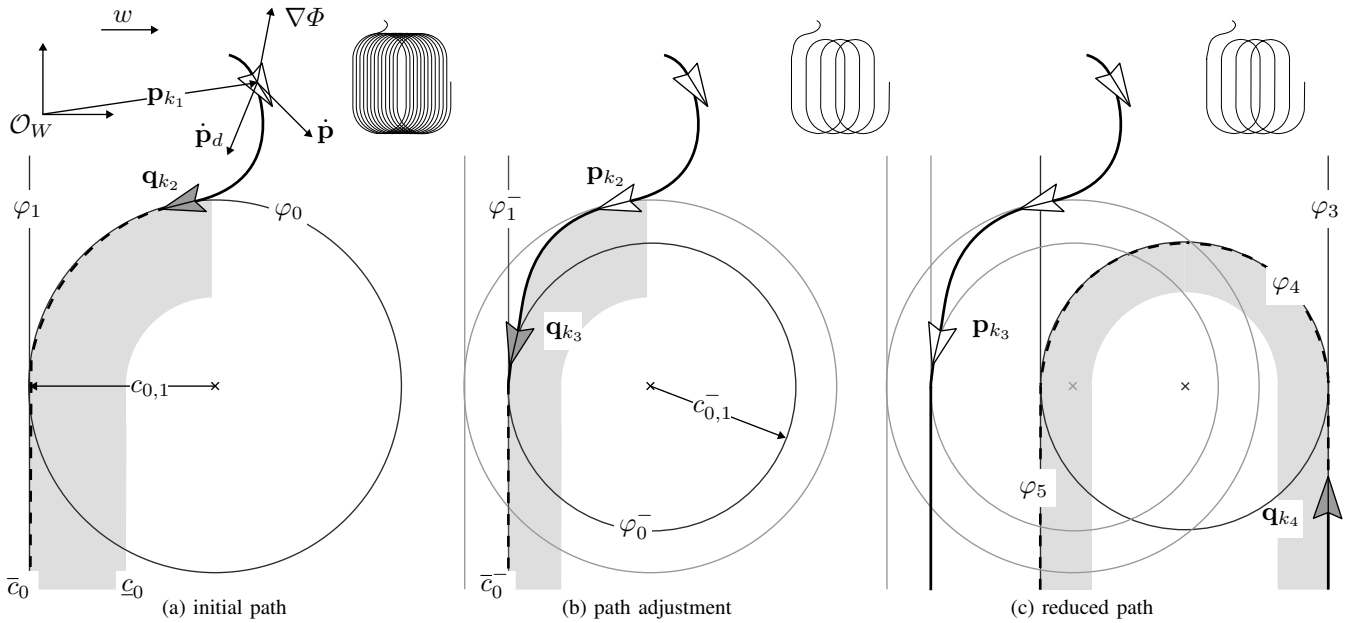

Fig. 1. The mission consists of circles and lines covering a polygon. The UAV heads to φ_0 initially, using the desired velocity vector $\dot{\mathbf{p}}_d$. It later heads to φ_0^- reducing the radius $c_{0,1}$ to satisfy the battery constraints. The UAV then converges to φ_1 in stage $\mathcal{M}_1$, φ_2 in stage $\mathcal{M}_2$, and so on (the circle φ_2 is not visible in the figure; it connects φ_1 and φ_3).

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 informs grounded mobile robots employed for harvesting. The algorithm plans the mission; it controls the processing rate and the radius of the circles (affecting the distance between the lines). Figure 1 shows an initial slice of such a mission. The UAV first heads to a circular TEE with a given radius, which is later reduced as, e.g., windy weather requires the adjustment of the control input to avoid battery depletion. Computations significantly impact the battery, with a potential extension of up to 13 minutes over an hour by merely switching to the lowest computations (see Section IV).

The remainder of the paper is organized as follows. The overview of dynamic mission planning is set in Section II, along with a suitable model for the position and energy. The algorithm that uses the model and solves the UAV dynamic mission planning problem is proposed in Section III. Section IV presents the result and showcase the performances. The paper finishes with some conclusions in Section V.

II. Mission Planning Overview

We assume that the initial mission plan consists of different stages. At each stage the robot must follow a path and do some tasks.

Let the path at stage i be characterized by a generic continuous twice differentiable TEE $\varphi_i : \mathbb{R}^2 \times \mathbb{R}^\rho \to \mathbb{R}$ and the tasks by functions $\psi_1, \dots, \psi_\sigma : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$. Moreover, let $[a]$ be the set $\{0, 1, \dots, a\}$, $[a]^+$ the set $[a]/\{0\}$, $\langle a_1, a_2, ..., a_n \rangle$ an ordered list of n elements, and $\underline{c}, \overline{c}$ the lower and upper bound of the parameter c retrieved from a lookup table.

Definition II.1 (Stage and mission)**.** A *stage* $\mathcal{M}_i$ at time instant k of a mission $\mathcal{M}$ is defined as the ordered list

$$\begin{aligned}\mathcal{M}_i := \{\langle \varphi_i(\mathbf{p}_k, c_{i,1}, \dots, c_{i,\rho}), \psi_1(s_{i,1}), \dots, \psi_\sigma(s_{i,\sigma})\rangle \\ \mid \exists\ \mathbf{p}_k, \varphi_i(\mathbf{p}_k, c_{i,1}, \dots, c_{i,\rho}) \in \mathbb{C}_i, \\ \psi_j(s_{i,j}) \in \mathbb{S}_{i,j}\ \forall j \in [\sigma]^+\},\end{aligned} \tag{1}$$

where $\mathbb{C}_i := [\underline{c}_i, \overline{c}_i] \subseteq \mathbb{R}$ is the TEEs set, and $\mathbb{S}_{i,j} := [\underline{s}_{i,j}, \overline{s}_{i,j}] \subseteq \mathbb{Z}_{\geq 0}$ the j-th task QoS set. $\mathbf{p}_k := (x, y)$ is a point of a UAV flying at an assigned altitude $h \in \mathbb{R}_{>0}$ w.r.t. some inertial navigation frame $\mathcal{O}_W$. The parameters of the TEE φ_i are the point and the adjustments (Subsection II-A). The parameters of the tasks $\psi_1, \dots, \psi_\sigma$ are the computations (Subsection II-C). Note that while the TEE can differ for each stage, the tasks remain the same. However, the user can inhibit or enable a task varying QoS set.

The *mission* is then $\mathcal{M} : \mathbb{R}^2 \to \mathcal{M}_i$ a function which maps the point $\mathbf{p}_k$ to a specific stage $\mathcal{M}_i$.

For simplicity the system is sampled in discrete time. The algorithm takes as input $\mathcal{M}$, initial position, and energy coefficients guess, and outputs the new position, the instantaneous energy consumption, and the control input–an action performed evolving the mission state.

A. State: position and energy

The state is the UAV's position in space and the energy coefficients in time. We show a linear relation between the instantaneous energy consumption and the energy coefficients in Theorem III.1, but the two are different. 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.

The set

$$\mathcal{P}_i := \{\mathbf{p}_k \mid \varphi_i(\mathbf{p}_k, c_{i,1}, \dots, c_{i,\rho}) \in \mathbb{C}_i\}, \tag{2}$$

delimits the area where the i-th TEE φ_i is free to evolve using ρ adjustments $\mathbf{c}_i := c_{i,1}, ..., c_{i,\rho}$ (the gray area in Figure 1).

The algorithm uses the set from Equation (2) to select the optimal adjustments $\mathbf{c}_i^0$ s.t. $\varphi_i(\mathbf{p}_k, \mathbf{c}_i^0)$ has the highest instantaneous energy consumption (while still respecting the energy budget). It guides the UAV to the new position $\mathbf{p}_{k+1}$ using the function $\Phi := \varphi_i(\mathbf{p}_k, \mathbf{c}_i^0)$, computing its vector field $\nabla\Phi := (\partial\Phi/\partial x, \partial\Phi/\partial y)$, and deriving the direction to follow in the form of velocity vector

$$\dot{\mathbf{p}}_d(\mathbf{p}_k) := E\nabla\Phi - k_e\Phi\nabla\Phi, \;\; E = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad (3)$$

where E specifies the tracking direction, and $k_e \in \mathbb{R}_{\geq 0}$ the gain to adjusts the speed of convergence. The direction the velocity vector $\dot{\mathbf{p}}_d$ is pointing at is generally different from the course heading due to the atmospheric interference, such as wind ($w \in \mathbb{R}$ in Figure 1).

B. Energy evolution due to trajectory

The algorithm models the energy using as state energy coefficients $\mathbf{q} \in \mathbb{R}^m$ derived from Fourier analysis (the size of the energy coefficients vector m is related to the order of a Fourier series) and decomposes the evolution in energy due to the trajectory and computations. This approach is adapted from our earlier work on computational energy analysis [28], [29], and energy estimation of a fixed-wing UAV [18].

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

$$h(k) = a_0/T + (2/T)\sum_{j=1}^{r}(a_j \cos\omega jk + b_j \sin\omega jk), \quad (4)$$

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

Consider a linear time-invariant (LTI) state-space model

$$\begin{aligned} \mathbf{q}_{k+1} &= A\mathbf{q}_k + B\mathbf{u}_k + \mathbf{w}_k, \\ y_k &= C\mathbf{q}_k + v_k, \end{aligned} \qquad (5)$$

where $y_k \in \mathbb{R}_{\geq 0}$ is the instantaneous energy consumption. The state $\mathbf{q}$ are the energy coefficients

$$\begin{aligned} \mathbf{q}_k &= \begin{bmatrix} \alpha_0 & \alpha_1 & \beta_1 & \cdots & \alpha_r & \beta_r \end{bmatrix}^T, \\ A &= \begin{bmatrix} 1 & & & & \\ & A_1 & & & \\ & & \ddots & & \\ & & & A_r \end{bmatrix}, A_j \begin{bmatrix} 1 & \omega j \\ -\omega j & 1 \end{bmatrix}, \\ C &= (1/T)\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \end{aligned} \qquad (6)$$

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

Lemma II.1 (Fourier series, LTI state-space equivalence)**.** Suppose there is no state and output uncertainty (the contributions $\mathbf{w}_k \in \mathbb{R}^m, v_k \in \mathbb{R}$ are zero) and the control is zero. The non-linear model in Equation (4) can be expressed using the LTI state-space model in Equation (5)

Proof. We re-write the Fourier series expression in Equation (4) in its complex form with the well-known Euler's formula $e^{it} = \cos t + i \sin t$. With $t = \omega jk$, we find the expression for $\cos\omega jk = (e^{i\omega jk} + e^{-i\omega jk})/2$ and $\sin\omega jk = (e^{i\omega jk} - e^{-i\omega jk})/(2i)$ by substitution of $\sin\omega jk$ and $\cos\omega jk$ respectively. This leads [30]

$$\begin{aligned} h(k) = a_0/T + (1/T)\sum_{j=1}^{r} e^{i\omega jk}(a_j - ib_j) + \\ (1/T)\sum_{j=1}^{r} e^{-i\omega jk}(a_j + ib_j), \end{aligned} \qquad (7)$$

where i is the imaginary unit.

Let us consider the continuous form of Equation (5), $\dot{\mathbf{q}} = A_c\mathbf{q}$, and an initial guess $\mathbf{q}_0$. A_c is defined as the continuous version of matrix A in Equation (6); $A_c := A - I$, where I is a properly sized identity matrix. The solution at time k can be expressed $\mathbf{q} = e^{A_c k}\mathbf{q}_0$. Both the solution and the system in Equation (5) are well established expressions derived using a standard textbooks [30], [31]. To solve the matrix exponential $e^{A_c k}$, we use the eigenvectors matrix decomposition method [32].

The method works on the similarity transformation $A_c = VDV^{-1}$. The power series definition of $e^{A_c k}$ implies $e^{A_c k} = Ve^{Dk}V^{-1}$ [32]. We consider the non-singular matrix V, whose columns are eigenvectors of A_c; $V := \begin{bmatrix} v_0 & v_1^0 & v_1^1 & \ldots & v_r^0 & v_r^1 \end{bmatrix}$. We then consider the diagonal matrix of eigenvalues $D = \mathrm{diag}(\lambda_0, \lambda_1^0, \lambda_1^1, \ldots, \lambda_r^0, \lambda_r^1)$. λ_0 is the eigenvalue associated to the first item of A_c. λ_j^0, λ_j^1 are the two eigenvalues associated with the block $A_j - I$. We can write $A_c v_j = \lambda_j v_j \;\; \forall j = \{1, \ldots, m\}$, and $AV = VD$.

We apply the approach in terms of the continuous equivalent of the model in Equation (5), $\dot{\mathbf{q}} = A_c\mathbf{q}$. Initial guess is expressed as a combination of eigenvectors $\mathbf{q}_0 = \gamma V$. The generic solution is $\mathbf{q}_k = \gamma e^{Dk} V$.

Let us express the output at time k as a linear combination of the state $y = C\mathbf{q} = C\gamma e^{Dk}V$

$$\begin{aligned} y = \gamma_0 e^{\lambda_0 k} v_0/T + (1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^0 k} v_j^0 + \\ (1/T)\sum_{j=1}^{r} \gamma_j e^{\lambda_j^1 k} v_j^1, \end{aligned} \qquad (8)$$

we proof that the eigenvalues λ and eigenvectors V are such that Equation (8) is equivalent to Equation (7).

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

First, we proof that the first term of both the Equation (7) and (8) matches. We find the eigenvalue from $\det(0 - \lambda I) = 0$, which is $\lambda_0 = 0$. The corresponding eigenvector can be chosen arbitrarily $(0 - \lambda_0)v_0 = 0 \;\; \forall v_0$, thus we choose $v_0 =$

1. We find the value γ_0 of the vector γ so that the terms are equal $a_0 = \gamma_0$.

Then, we proof that all the terms in the sum of both the Equations (7) and (8) match.

For the first block $A_1 - I$, we find the eigenvalues from $\det(A_1 - (1+\lambda)I) = 0$. 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 $v_1^0 = \begin{bmatrix} -i & 1 \end{bmatrix}^T$. The eigenvector associated with the eigenvalue λ_1^1 is $v_1^1 = \begin{bmatrix} i & 1 \end{bmatrix}^T$. Again, we find the subvector γ_1 of the vector γ such that the equivalences $e^{i\omega k}(a_1 - ib_1) = \gamma_1 e^{i\omega k} \begin{bmatrix} -i & 1 \end{bmatrix}^T$ and $e^{-i\omega k}(a_1 + ib_1) = \gamma_1 e^{i\omega k} \begin{bmatrix} i & 1 \end{bmatrix}^T$ 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 discretized version A of the matrix A_c can be expressed $A = A_c + I$ for small enough values of time step (forward Euler approximation[1]). We thus conclude that the lemma holds.

■

Suppose at time instant k the mission reached stage i and the control input is $\mathbf{u}_k^a := \langle \mathbf{c}_k, \mathbf{s}_k \rangle$. The control along with the input matrix

$$\mathbf{u}_k = \begin{bmatrix} g(\mathbf{s}_k) - g(\mathbf{s}_{k-1}) & \mathbf{c}_k - \mathbf{c}_{k-1} \end{bmatrix}^T,$$
$$B = \begin{bmatrix} 1 & \omega_{i,1} & \cdots & \omega_{i,\rho} \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}, \tag{9}$$

where $\mathbf{s}_k, g(\mathbf{s}_k)$ are the computations and the instantaneous computational energy consumption (Subsection II-C), $\mathbf{u}_k \in \mathbb{R}^n$ is the control given $n = 1+\rho$, and $B \in \mathbb{R}^{m \times n}$. Moreover, the top-left entry of B is one, while the others on the first row are gain factors $\Omega_i := \{\omega_{i,1}, \ldots, \omega_{i,\rho}\} \in \mathbb{R}^\rho$, quantifying the contribution of a given adjustment to the instantaneous energy consumption.

Equation (9) accounts for the energy due to the adjustments and computations. The difference $g(\mathbf{s}_k) - g(\mathbf{s}_{k-1})$ quantifies the change in the instantaneous computational energy consumption. Similarly, $\Omega_i(\mathbf{c}_k - \mathbf{c}_{k-1})$ quantifies the change in the instantaneous trajectory energy consumption. For instance, when the TEE φ_0 is a circle (see Figure 1), a decrement in the adjustment radius of the circle $c_{0,1}$ adds a negative contribution. It thus simulates the lowering of instantaneous energy consumption $\omega_{0,1}(c_{0,1} - c_{0,1}^-) < 0$.

Note from Equation (9) that the control input $\mathbf{u}_k^a$ differs from the nominal control $\mathbf{u}_k$ in Equation (5). The first is an output of the algorithm. The second is a function that maps two consecutive control inputs to the difference in instantaneous energy consumption $\mathbf{u}_k : \mathbb{R}^{2\rho} \times \mathbb{Z}_{\geq 0}^{2\sigma} \to \mathbb{R}^{1+\rho}$.

The energy consumption modeling of the mission necessitates the following assumption.

[1]One can guarantee to have the same outputs rather than an approximation with $A = e^{A_c k}$

Assumption II.2 (Energy evelution periodicity)**.** Given two time instants k_1, k_2 s.t. $k_1 > k_2$ and a constant value $n \in \mathbb{Z}_{>0}$

$$|y_k - y_{k+n}| \in \mathbb{E} \subset \mathbb{R}_{\geq 0} \ \forall k \in [k_1, k_2]. \tag{10}$$

Physically this means that the time evolution of the instantaneous energy consumption is assumed to be approximately periodic.

C. Energy evolution due to computations

The energy cost of the computations (the set of tasks being executed on an embedded board on the UAV) is assessed using `powprofiler`, an open-source modeling tool presented in our previous work [28]. The tool measures the instantaneous computational energy consumption of software components within the QoS sets. It builds an energy model: a linear interpolation, one per each task. In a ROS based system, it requires the user to implement one or more ROS nodes changing the computational load by node-specific ROS parameters. The approach has been tested in simulation in our previous work [33].

Specifically, the mission $\mathcal{M}$ contains a set of ordered lists with tasks (recall Definition II.1). These tasks are implemented by software components (such as ROS nodes in a ROS based system) $\Psi(\mathbf{s}_i) := \langle \psi_1(s_{i,1}), \ldots, \psi_\sigma(s_{i,\sigma}) \rangle$. They input the desired and output the actual computations

$$\mathbf{s}_i := \left\{ \langle s_{i,1}, \ldots, s_{i,\sigma} \rangle \mid \psi_j(s_{i,j}) \in \mathbb{S}_i \, \forall j \in [\sigma]^+ \right\}, \tag{11}$$

where $s_{i,j} : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ returns the j-th desired computation at stage i, $\mathbf{s}_i \in \mathbb{S}_i \subseteq \mathbb{Z}_{\geq 0}^\sigma$ the union of all QoS sets given $\mathbb{S}_i := \bigcup_{j \in [\sigma]^+} \mathbb{S}_{i,j}$.

For instance, if the task ψ_1 is a CNN object detector, $s_{1,1}$ corresponds to the computation frames-per-second (fps) rate changing the detection frequency. The algorithm outputs the control input which contains the desired fps rate $s_{1,1} \in \mathbb{S}_{1,1}$. The task $\psi_1(s_{1,1})$ outputs the actual fps rate $s_{k,1}$ which might differ from the desired one (e.g., the CNN object detector might not be able to reach a high fps rate with the current computational resources).

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

$$y_k^s := g\left(\Psi(\mathbf{s}_i)\right) = g\left(\mathbf{s}_k\right). \tag{12}$$

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

III. Algorithm

Given l stages $\mathcal{M}_i$ (TEE φ_i, tasks Ψ, computations $\mathbf{s}_i \in \mathbb{S}_i$, and adaptations $\mathbf{c}_i \in \mathbb{C}_i$ for all $i \in [l]$), the main purpose of the algorithm is to output a control input sequence $\mathbf{u}^a := \{\mathbf{u}_0^a, \mathbf{u}_1^a, \ldots\}$ in a valid mission.

Definition III.1 (Valid mission)**.** A mission is valid if for every stage $\mathcal{M}_{i-1}$, $i \in [l]^+$ there exist a control input $\mathbf{u}_k^a$ that produce the next stage $\mathcal{M}_i$

$$\begin{aligned} \mathbf{u}_k^a = \{ & \langle \mathbf{c}_k, \mathbf{s}_k \rangle \mid \exists n \in \mathbb{Z}_{>0}, \\ & \langle \varphi_{i-1}(\mathbf{p}_{k-n}, \mathbf{c}_{k-n}), \Psi(\mathbf{s}_{k-n}) \rangle \in \mathcal{M}_{i-1} \\ & \implies \langle \varphi_i(\mathbf{p}_k, \mathbf{c}_k), \Psi(\mathbf{s}_k) \rangle \in \mathcal{M}_i \}. \end{aligned} \tag{13}$$

Let us proof that if the mission is valid, the instantaneous energy consumption can be modeled as linear combination of the state from the Equation (5).

Theorem III.1 (Periodic energy model)**.** Consider the mission from Definition II.1, the valid mission from III.1. Assume Assumption II.2 holds, the model of Equation (5) behaves ideally ($\mathbf{w} = \mathbf{0}, v = 0$), the initial energy coefficients state $\mathbf{q}_0$ is y_0^a/m for the first coefficient where $y_0^a \in \mathbb{R}_{>0}$ is an initial measurement[2], $(1/2)y_0^a/m$ for all the others, and the mission is valid. Then, the instantaneous energy consumption y_k is a linear combination of the state $\mathbf{q}_k$.

Proof. The proof is based on mathematical induction. Base case: we proof that $y_0 = y_0^a$. Recall the definition of the state in Equation (6). The output is $y_0 = \alpha_{0,0}+\alpha_{0,1}+\cdots+\alpha_{0,r} = y_0^a/m + (1/2)y_0^a/m + \cdots + (1/2)y_0^a/m = y_0^a$.

Induction step: by inspection of Equation (5), the output at instant k can be expressed $y_k = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_{k-1}) + p_1(k)\alpha_{0,1} + \cdots + p_r(k)\alpha_{0,r}$, where $\forall t \in \mathbb{Z}_{\geq 2}$

$$p_r(t) := \begin{cases} \prod_{i=1}^{t/2} r^3/\xi^3 & \text{for even } t \\ (r/\xi)\prod_{i=1}^{(t-1)/2} r^3/\xi^3 & \text{for odd } t \end{cases}. \quad (14)$$

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

We prove now that the instantaneous energy consumption at $k+1$ is still a linear combination of the state. We express the output in function of the previous state $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + (1/\xi)\beta_{k,1} + \cdots + (r/\xi)\beta_{k,r}$. Notice that the coefficients α, β have an equivalent evolution (indeed this allows to simulate the periodicity) and $\beta_{k,r} = p_r(k)\beta_{0,r}$. Thus, the output can be expressed $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + (1/\xi)p_1(k)\beta_{0,1} + \cdots + (r/\xi)p_r(k)\beta_{0,r}$. The expression is equivalent to $y_{k+1} = (\alpha_{0,0} + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + p_r(k+1)\beta_{0,r} + \cdots + p_r(k+1)\beta_{0,r}$ using the definition of p_r in Equation (14). Again, the state $\mathbf{q}_0$ leads to $y_{k+1} = (y_0^a/m + B\mathbf{u}_0 + \cdots + B\mathbf{u}_k) + p_r(k+1)(1/2)y_0^a/m + \cdots + p_r(k+1)(1/2)y_0^a/m = y_{k+1}^a$, alike the previous statement, but at instant $k+1$. The proof for odd k is equivalent. ■

A. Output constraints set

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

Let us hence consider the state of charge (SoC) of such battery with a simplistic difference equation [18]

$$\mathrm{SoC}_k = -\left(V - \sqrt{V^2 - 4R_r\tilde{V}y_kV^{-1}}\right)/2R_rQ_c, \quad (15)$$

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

$$\mathbb{Y}_k := \{y_k \mid y_k \in [0, \mathrm{SoC}_k Q_c V] \subseteq \mathbb{R}_{\geq 0}\}, \quad (16)$$

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

B. Deployment algorithm

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

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

Per each time step k (the final time t_f is unknown), the algorithm updates the state–the position at line 30 and the energy coefficients at line 29–and the control input. Note that the position can be computed directly from Equation (3). If the velocity is $v \in \mathbb{R}_{\geq 0}$, and the starting point $\mathbf{p}_0$, $\mathbf{p}_{k+1} = \mathbf{p}_k\dot{\mathbf{p}}_d(\mathbf{p}_k)/v$.

In detail, initial guess for $P_0 \in \mathbb{R}^{j\times j}$ is positive definite and derived empirically, for $\mathbf{q}_0$ the initial measurement is distributed to the coefficients (see Theorem III.1). Line 2 selects the maximum possible control from the current control input. Line 3 uses robust output feedback model predictive control (MPC) [34] to select the optimal control $\mathbf{u}^0$ for a

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

given horizon $N \in \mathbb{Z}_{>0}$ from the cost function

$$l(\mathbf{q}_k, \mathbf{u}_k) := (1/2)(\mathbf{q}_k^T Q \mathbf{q}_k + \mathbf{u}_k^T R \mathbf{u}_k), \\ V_f(\mathbf{q}_k) := (1/2)(\mathbf{q}_k^T P_f \mathbf{q}_k), \tag{17}$$

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

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

Before the next step, state estimator–the discrete-time Kalman filter [35] at lines 13–16–predicts the state $\mathbf{q}$ if the modeled instantaneous energy consumption diverges from the sensor's value y_k^a more than a given $\varepsilon \in \mathbb{R}_{\geq 0}$, or sensor measurements are unavailable ($y_k^a = 0$).

IV. Results

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

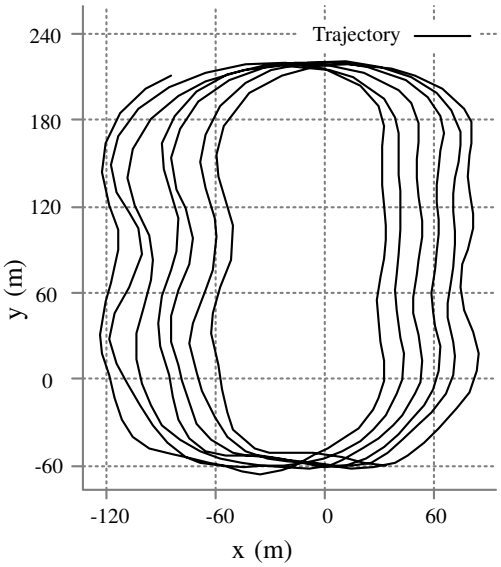

Fig. 2. The true trajectory. The trajectory is retrieved from the flight log.

V. Conclusion and Future Work

*

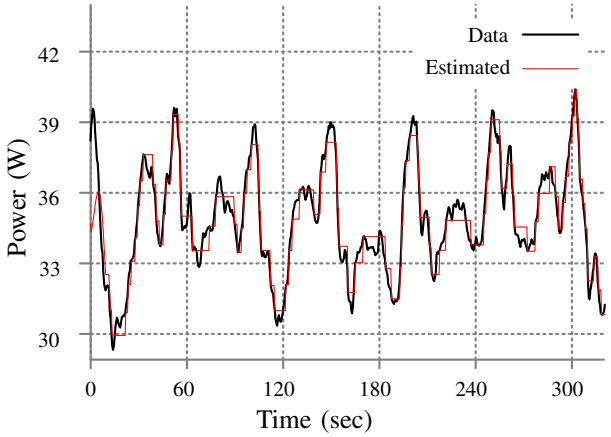

Fig. 3. Derived energy consumption from the sensors (throttle). Red line is the estimated value. We use KF for state estimation when the difference between the sensor and model value is $> \varepsilon$. Otherwise we evolve the state without noise.

REFERENCES

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] ——, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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

[19] 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://adamseew.bitbucket.io/short/beyond2020

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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

[29] 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

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

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

[32] 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.

[33] 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

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

APPENDIX I
MISSION PLAN FORMAT

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

Recall that the mission is composed of a set of stages (Section II). The plan has a variable number of entries per each stage. We plan the mission in Figures 1, 2. The first line

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

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

The following line

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

specifies the triggering point. **sqrt** is the square root. The UAV reaches the triggering point within a given ε (the tolerance), and the algorithm switches to the next stage $\mathcal{M}_1$. The algorithm ensures that the overall mission is still valid–Equation (13). It ensure that the TEEs and QoS constraints sets are satisfied with the generated control input. Note that the triggering point is in function of the adjustment **c1**. This is necessary; an adjustment in one stage affects the overall flight.

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

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

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

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

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

line 6 corresponds to TEE $\varphi_1(\mathbf{p}_k, c_{0,1}) := x + \sqrt{4900 + c_{0,1}} + 45$, a linear equation that intersects the triggering point (line 2). The gain is $k_e = 5 \cdot 10^{-2}$, and the rotation is opposite to line 1. The algorithm computes $-E$ from the rotation matrix in Equation 3.

The gain is different from the one used in φ_0 as different equations require different convergence rate. When we follow a circle it is useful to turn the rotation direction in advance. When we follow a straight line, it is preferable to turn the rotation direction closer to the line. We refer to the simulator in Appendix III for more details on how to tune the gain k_e. The reader can simulate different rates to see how they affect the flight. The rotation is likewise different. In fact, the rotation matrix E points upwards in the space. It guides the UAV in the counter-clockwise direction (see Figure 1; the UAV is traveling counter-clockwise). The rotation matrix $-E$ points downwards and guides the UAV in the clockwise direction.

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

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

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

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

The lines

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

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

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

computations

24: `[0,0]`

25: `[0,0]`

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

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

Appendix II
Components Specification Format

*

Appendix III
Simulator

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

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

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