***

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

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

I. Introduction

*

II. State of the Art

*

III. Model

The model presented in this section deals with the motion and the energy using two distinct systems, subject to the same process noise w (i.e., the atmospheric interferences such as wind).

The motion defines a trajectory and a position $\mathbf{p}$; the trajectory is expressed through explicit trajectory equation φ, while the position determines a vehicle frame $\mathcal{O}_V$ in the 2D Euler space relative to an inertial frame $\mathcal{O}_W$. For the sake of simplicity, it is assumed that the z-axis is constrained to a specific altitude h, with the system being free to evolve in the x, y-axes (i.e., the drone is flying, performing a mission, at a given height). A motion guidance action is derived using vector field design, enabling the convergence to the desired trajectory anywhere in $\mathcal{O}_W$.

The energy $\mathbf{q}$ spent to perform such guidance action is later derived using Fourier analysis and its state-space representation. This component is decomposed in the energy needed for the actuation (i.e., the mechanical energy), and the computations (i.e., the computational energy). A control action, which varies quality of service (QoS) levels of the software being executed, is derived for this purpose. The main goal is to tune the explicit energy equation allowing to maximize the computational and mechanical outcome of the mission while trying to minimize the eventuality of battery discharge.

[1]Corresponding author; email: `ads@mmmi.sdu.dk`

Adam Seewald, Ulrik Pagh Schultz are with the SDU UAS Center, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, Denmark.

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

A. Motion and Guidance Model

The position $\mathbf{p}$, derived from [1], is described by the following non-holonomic model:

$$\begin{cases} \dot{\mathbf{p}}(t) &= s\Psi(\psi(t)) + w(t), \\ \dot{\psi}(t) &= u(t), \end{cases} \tag{1}$$

where $\mathbf{p}(t) \in \mathbb{R}^2$ is a point in $\mathcal{O}_W$, $s \in \mathbb{R}$ describes the airspeed assumed constant, $\psi \in (-\pi, \pi]$ the attitude yaw angle and $\Psi(\psi(t)) = \begin{bmatrix}\cos\psi(t) & \sin\psi(t)\end{bmatrix}^T$, and $u \in \mathbb{R}$ the guidance action which express the angular velocity $\dot{\psi}$ of $\mathcal{O}_V$.

Let us define a generic continuously differentiable function $\varphi : \mathbb{R}^2 \to \mathbb{R}$. The desired trajectory $\mathcal{P}$ can be hence defined:

$$\mathcal{P} := \{\mathbf{p} : \|\varphi(\mathbf{p})\| < c + \varepsilon\}, \tag{2}$$

as the set of points that follows the trajectory $\varphi(\mathbf{p}) = c$ within a given $\varepsilon \in \mathbb{R}_{\geq 0}$.

Given a set of discrete values c, the space relative to $\mathcal{O}_W$ can be covered by all the trajectories $\mathcal{P}$ within specific boundaries $\underline{c} \leq c \leq \overline{c}$, where $\underline{a}, \overline{a}$ returns the upper bound and lower bound limit of a from the mission specification, which is a lookup table.

The concept of a trajectory is used later to design a controller that selects c with the highest energy value under the energy budget constraints defined in the next subsection. Here we design a guidance action u that allows following such trajectory. The guidance is derived using the vector field approach presented in [1]. The algorithm heads to $\mathcal{P}$ using a vector field based on the minimization of the norm in Equation (2).

Let us define $\mathit{\Phi} := \varphi(\mathbf{p})$. The vector field is hence defined as a desired velocity vector

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

where $\nabla\mathit{\Phi} \in \mathbb{R}^2$ is defined as the gradient of φ at the point $\mathbf{p}$ (i.e., it's vector field), E specifies the tracking direction, and $k_e \in \mathbb{R}_{\geq 0}$ the gain which adjusts the speed of convergence of the vector field to the desired trajectory.

The direction the velocity vector $\dot{\mathbf{p}}$ is pointing at is generally different from the course heading $\chi \in (-\pi, \pi]$ due to the atmospheric interferences.

Let us further define $\hat{\mathbf{p}} := \mathbf{p}/\|\mathbf{p}\|$, the desired course heading rate $\dot{\chi}_d$ is computed by sensing the position $\mathbf{p}$, the ground velocity $\dot{\mathbf{p}}$, and is expressed

$$\dot{\chi}(\mathbf{p},\dot{\mathbf{p}}) = -E\frac{\dot{\mathbf{p}}_d}{\|\dot{\mathbf{p}}_d\|^2}\cdot \left(E\hat{\dot{\mathbf{p}}}_d\hat{\dot{\mathbf{p}}}_d^T\left((E-k_e\Phi)H(\Phi)\dot{\mathbf{p}} - k_e\nabla\Phi^T\dot{\mathbf{p}}\nabla\Phi\right)\right)^T, \tag{4}$$

where $H(\cdot)$ is defined as the Hessian operator, with the physical meaning being that the curvature of the desired trajectory has to be known in order to be tracked.

Under the assumption of the airspeed $s = \|w(t)\|$ for $t > 0$ (i.e., the constant airspeed being greater than the norm of the wind), the guidance action defines how fast the drone converges to the desired trajectory and can be expressed

$$u(\dot{\mathbf{p}},\mathbf{p},\psi) = \frac{\|\mathbf{p}\|}{s\cos\beta}\left(\dot{\chi}_d(\dot{\mathbf{p}},\mathbf{p}) + k_d\hat{\dot{\mathbf{p}}}^T E\hat{\dot{\mathbf{p}}}_d\right), \tag{5}$$

where $k_d \in \mathbb{R}_{\geq 0}$ is the gain which adjusts the speed of convergence of the drone to the vector field $\dot{\mathbf{p}}_d$ to the desired trajectory, $\beta = \cos^{-1}\left(\hat{\dot{\mathbf{p}}}^T\Psi(\psi)\right)$ is the slideslip angle, and $\dot{\chi}_d$ is given in Equation (4).

—

B. Energy Model

We consider a Fourier series $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ of an arbitrary order r to evaluate the energy spent performing a given motion and software configuration

$$f(t) = \sum_{n=0}^{r} a_n \cos\frac{nt}{\xi} + b_n \sin\frac{nt}{\xi}, \tag{6}$$

where $\xi \in \mathbb{R}$ is the characteristic time, and $a_n, b_n \in \mathbb{R}$ for $n \in \{0,\dots,r\}$ the Fourier series coefficients. The Fourier analysis allows to account for the periodicity of the mission, with a trajectory $\mathcal{P}$ being reitared over time.

The non-linear model in Equation (6) can be expressed using an equivalent linear time-varying state-space model, expressed in the following form:

$$\begin{cases} \dot{\mathbf{q}}(t) &= A\mathbf{q}(t) + B\mathbf{u}(t) + w(t), \\ y(t) &= C\mathbf{q}(t) + v(t), \end{cases} \tag{7}$$

where $y(t) \in \mathbb{R}_{\geq 0}$ is the energy evolution of the controlled system, $w(t)$ the same process noise from the system in Equation (1), $v(t) \in \mathbb{R}_{\geq 0}$ the measurement noise, and the control $\mathbf{u}$ along with the input matrix B are defined subsequently. The state $\mathbf{q}$ respresents the evolution of the Fourier series coefficients in time, which can be expressed along the state transition matrix, and the output matrix as follows

$$\mathbf{q}(t) = \left[\begin{array}{c|ccccc} a_0 & a_1 & b_1 & \cdots & a_r & b_r \end{array}\right],$$
$$A = \left[\begin{array}{c|ccc} 1 & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & A_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_r \end{array}\right], A_n = \begin{bmatrix} 0 & n \\ -n^2 & 0 \end{bmatrix}, \tag{8}$$
$$C = \left[\begin{array}{c|ccccc} 1 & 1 & 0 & \cdots & 1 & 0 \end{array}\right],$$

where $\mathbf{q}(t) \in \mathbb{R}^j$ given $j := 2r+1$, $A \in \mathbb{R}^{j\times j}$ is the state transmission matrix, $C \in \mathbb{R}^j$ is the output matrix respectively. Furthermore, the first row and column of the matrix A contain a zeros row vectors and columns vectors respectively.

Motivated by the fact that the system of interest is sampled at discrete time, and for the sake of simplicity, we consider the discretized version of Equations (1 and 7) by employing the following expression

$$\begin{cases} \mathbf{p}_{k+1} &= s\Psi(\psi_k) + w_k, \\ \psi_{k+1} &= u_k, \end{cases} \tag{9b}$$

$$\begin{cases} \mathbf{q}_{k+1} &= A\mathbf{q}_k + B\mathbf{u}_k + w_k, \\ y_k &= C\mathbf{q}_k + v_k. \end{cases} \tag{9a}$$

As the system was observed to behave stochastically, with the process and measurement noise evolving in a normal distribution, a Kalman filter [2], [3] is employed to predict the state $\hat{\mathbf{q}}$. Such a state, along with the predicted output $\hat{y}$, differs from the model state $\mathbf{q}$ and measured output y in Equation (9b) due to the presence of uncertainty.

The prediction is done using the following expression:

$$\hat{\mathbf{q}}_{k+1}^- = A\hat{\mathbf{q}}_k + B\mathbf{u}_k, \tag{10a}$$
$$P_{k+1}^- = AP_kA^T + Q, \tag{10b}$$

where $\hat{\mathbf{q}}_k^-, \hat{\mathbf{q}}_k \in \mathbb{R}^j$ depicts the estimate of the state before and after measurement (or simply estimate), and $P_k, P_k^- \in \mathbb{R}^{j\times j}$ the error covariance matrix (i.e., the variance of the estimate before measurement).

The estimation of the state and the update of the predicted output is done using the following expression:

$$K_k = (CP_{k+1}^-C^T + R)^{-1}(P_{k+1}^-C^T), \tag{11a}$$
$$\hat{\mathbf{q}}_{k+1} = \hat{\mathbf{q}}_{k+1}^- + K_k(y_k - C\hat{\mathbf{q}}_{k+1}^-), \tag{11b}$$
$$P_{k+1} = (I - G_{k+1}C)P_{k+1}^-, \tag{11c}$$
$$\hat{y}_k = C\hat{\mathbf{q}}_{k+1}, \tag{11d}$$

where $K_k \in \mathbb{R}^j$ is the gain of the Kalman filter, and I the identity matrix. The noise covariance matrixes $Q \in \mathbb{R}^{j\times j}, R \in \mathbb{R}$ indicates the process noise and sensor noise covariance respectively, and $\hat{y}_k \in \mathbb{R}_{\geq 0}$ is the estimated energy.

Equation (10) converges to the predicted energy evolution as follows. An initial guess of the values $\hat{\mathbf{q}}_0, P_0$ is derived empirically from collected data. It is worth considering that an appropriate guess of these parameters allows the system to converge to the desired energy evolution in a shorter amount of time. The tuning parameters Q, R are also derived from the collected data and may differ due to i.e., different sensors used to measure the instantaneous energy consumption, or different atmospheric conditions accounting for the process noise.

At time $k = 0$, the initial estimate before measurement of the state and of the error covariance matrix is updated in Equation (10a) and (10b) respectively. The value of $\hat{\mathbf{q}}_1$ is then used in Equation (11b) to estimate the current state along with the measurment from the sensor y_0, where the sensor noise covariance matrix R accounts for the amount of uncertainity in the measurement. The estimated ouput $\hat{y}_0$ is then obtained from Equation (11d). The algorithm is iterative. At time $k=1$ the values $\hat{\mathbf{q}}=1, P_1$ computed at previous step are used to estimate the values $\hat{\mathbf{q}}_2, P_2$, and y_1.

Two different components of the overall energy consumption are being modeled in our analysis, an approach that has been extensively reviewed in our previous work [4]–[7]. A mechanical energy model accounts for the energy by reason of the physical system being moved in space, whereas a computational energy model accounts for the computations. The main goal is to derive a control action $\mathbf{u}$ for the current state $\hat{\mathbf{q}}$ from the optimal control law $\kappa(\hat{\mathbf{q}})$. This is achieved by solving online a finite horizon optimal control problem by the hand of a model predictive control (MPC) derived later in this section.

C. Computational Model

A computational energy model is built using `powprofiler`[1], an open-source modeling tool that measures empirically a number of software configuration and build an energy model accordingly [4]. The tool builds a multivariate linear interpolation which is accessed online at the hand of a lookup table in the optimal control algorithm. An existing ROS system composes a number of computationally expensive ROS nodes, allowing to vary the amount of computations changing node-specific quality of service (QoS) values. The tool builds the energy model using a file specification which specifies per each ROS node a QoS range.

Suppose the system is composed of σ computationally expensive ROS nodes. Let us define the computational control action as

$$\mathcal{C}_k := \left\{u : u \in \mathrm{QoS}_n(k) \, \forall n \in \{0, \ldots, \sigma\}\right\}, \tag{12}$$

where $\mathrm{QoS}_n(t) : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ returns the n-th QoS value at time t, and $\mathcal{C}_k \in \mathbb{Z}^{\sigma}_{\geq 0}$ the set of σ QoS values the system is composed of at time k. Let us further define $g : \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as the instantaneous energy value obtained interogating `powprofiler`. The computational energy component can be hence described

$$\begin{aligned} g(\mathcal{C}_k) &:= \{g\left(0, \mathrm{QoS}_0(k)\right), \ldots, g\left(\sigma, \mathrm{QoS}_\sigma(k)\right)\}, \\ y_c(t) &= \sum_{x \in g\left(\mathcal{C}_{\lceil t \rceil}\right)} x, \end{aligned} \tag{13}$$

where $\lceil \cdot \rceil$ is the ceiling function that returns the least integer greater than or equal to the argument.

The QoS parameters $\mathcal{C}_k$ can be subject to different constraints at different states. Physically, this means that the drone can perform the ROS nodes within different QoS ranges while flying different phases of a mission

$$\underline{\mathrm{QoS}}_n(k) \leq \mathrm{QoS}_n(k) \leq \overline{\mathrm{QoS}}_n(k), \, \forall n \in \{0, \cdots, \sigma\}, \tag{14}$$

where the values $\underline{\mathrm{QoS}}_n(k), \overline{\mathrm{QoS}}_n(k)$ are retrived from the mission specification.

[1] https://bitbucket.org/adamseew/powprofiler

The control action is constructed in two steps. Equation (12) defines the control due to the computational model. The control due to the energy model, which describes the energy of the mechanical elements of the drone $\mathcal{M}$, is derived in the following subsection.

D. Control Action

Given a generic trajectory equation φ, the trajectory can be modeled by ρ parameters $\mathbf{u} \in \mathbb{R}^\rho$, e.g., the constants of a linear function, the radius of a circonference, and semi-major and minor axis of an ellipse. The set of these parameters can be expressed

$$\mathcal{M}_k := \{\mathbf{u}_k : \varphi_k(\mathbf{p}^0_k, \mathbf{u}_k) = c_k\}, \tag{15}$$

where c is being defined in Equation (2), and $\mathbf{p}^0$ is the optimal point which lets the trajectory explicit function φ_k converge to the value c.

The explicit trajectory equation φ_k can be different for different states k, meaning the vector field and guidance action, from Equations (3) and (5) respectively, will account for the sudden change of trajectory during the mission. Alike Equation (15), the parameters $\mathbf{u}_k = \{u_{k,1}, \ldots, u_{k,\rho_k}\}$ at time k are constrained

$$\underline{u}_{k,n} \leq u_{k,n} \leq \overline{u}_{k,n} \, \forall n \in \{0, \cdots, \rho_k\}, \tag{16}$$

where the values $\underline{u}_{k,n}, \overline{u}_{k,n}$ are also retrived from the mission specifcation.

It is worth considering that the number of parameters at state k is also a parameter of the state. This is for the sake of generality, as the mission specification might contain different explicit equations for different states. Meaning that the drone might follow in an ellipse function thorought the mission and heading a linear function while landing.

The mechanical and computational control actions, $\mathcal{C}$ and $\mathcal{M}$ defined in Equation (12) and (15) respectively, are incorporated in the system in Equation (9b) using the input matrix

$$\begin{aligned} \mathbf{u}_k &= \begin{bmatrix} g\left(\mathcal{C}_k\right) & \mathcal{M}_k \end{bmatrix}^T, \\ B &= \left[\begin{array}{ccc|ccc} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right], \end{aligned} \tag{17}$$

where $\mathbf{u}_k \in \mathbb{R}^l$ is the control given $l := \sigma + \rho$, and $B \in \mathbb{R}^{j \times l}$ is the input matrix from Equations (7) and (9b). Moreover, the first σ columns in the first row of the input matrix are 1, while all the other items are 0. This adds the computational model component to the energy evolution in the system in Equation (9b). The energy due to the change of explicit trajectory equation parameters is not directly added to the system, which hovewever will update the reading from the sensors in Equation (11b) and thus adjust the energy evolution accordingly.

IV. Evaluation

*

V. Conclusion and Future Work

*

Acknowledgment

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

References

[1] 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.

[2] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 1994.

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

[4] 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.

[5] 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.

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

[7] A. Seewald, H. Garcia de Marina, A. Hasan, H. S. Midtiby, and U. P. Schultz, "Towards mission-aware energy assessment for autonomous drones," 2020, p. submitted.