Mechanical and Computational Energy Estimation of a Fixed-Wing Drone

Adam Seewald, Hector Garcia de Marina, Henrik Skov Midtiby, and Ulrik Pagh Schultz
SDU UAS, Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Email: {ads,hgm,hemi,ups}@mmmi.sdu.dk

***Abstract*—In this paper, we present a case study on the energy estimation of drones and derive a general modeling approach that estimates computational and mechanical energy separately. The mechanical energy model can easily be extended to other drones and is built using a Fourier series from a number of training flights. The computational energy model is more advanced as it handles heterogeneous hardware and incorporates a specification that defines the quality-of-service ranges for software components of the robotic system. The computational model is suitable for any mobile robot and is implemented in a modeling tool. The tool automatically generates an energy model from the specification by performing a set of empirical trials for selected configurations while approximating others. Information about the battery State of Charge is also included in the tool, hence allowing the evaluation of how different software configurations impact the battery. This approach can be used for dynamic mission assessment regarding different planning decisions. We here have demonstrated its ability to model the energy of a specific mission performed at varying levels of quality-of-service using a specific drone.**

I. Introduction

Mobile robots, wearable electronics, and in general portable embedded devices are all examples of power-critical systems concerned with energy efficiency. Though this is a common problem in many different domains, in mobile robotics it is one of the major challenges as the level of autonomy of mobile robots is often bounded by a limited power source, typically represented by a lithium-ion battery. In this paper, we propose a systematic approach based on a case study, to estimate the energy consumption of a fixed-wing drone. An accurate estimation of the system within a specific configuration is done by modeling separately the mechanical and computational energy. The latter is part of our analysis as in certain mobile robotic systems more than half of the overall power consumption can be due to computations, with mechanical motion accounting for the remaining [1].

Computational energy side: our approach is automatic in the sense that the developer specifies different components of the system, along with ranges of quality-of-service (QoS) from the lowest admissible to the highest achievable. From this information, a modeling algorithm automatically generates all the possible configurations and profiles a number of them while approximating the others statistically. *Mechanical energy side*: using an Opterra fixed-wing drone, a number of training flights are performed prior to the energy estimation phase. Data from these flights are collected and further analyzed with a weighted statistical regression technique, where a Fourier series is used to describe mechanical energy in time for the three phases of a flight with major variability: take-off, cruise, and landing. Finally, using numerical analysis, a battery model is derived into the battery State of Charge (SoC) to evaluate the battery efficiency of different components configurations.

The information about SoC can be also used to evaluate the energy cost of different components, or components executing in a larger component-based framework, like the one outlined in our previous work [2]. Moreover, SoC is of particular interest for mobile robots where it can be used to assess dynamically energy efficiency of software featuring autonomous tasks, and thus to correlate the battery state to the level of autonomy. The approach can be further extended to address computational energy-aware planning decisions. A drone, for instance, can use the SoC of the current computation to estimate its energy cost. The information can be propagated to dynamically adjust component configurations to meet specific energy targets, as the drone might limit its QoS in order to meet specific mission-dependent criteria. To this extent, we presented a modification of a real-time scheduling algorithm [3] in our previous work [2], that can be used to assess the energy cost of software components running in a larger, constrained network where one software component input might depend from the others. The scheduler is statically generated in this context, which allows meeting real-time requirements of many mobile robot scenarios.

Our approach is implemented in a profiling tool named `powprofiler`.[1] The tool, written in C++ and distributed under MIT license, has been further extended from [2], [4] to support the mechanical energy model described in this paper, as energy estimation of mobile robots often requires an expected energy value as a function of the mission time. The current implementation allows deriving an energy model that includes, but is not limited to, SoC from an arbitrary number of software components and a given flight mission. It allows the evaluation of the computational and mechanical energy consumption against the overall energy budget, and thus to reduce or increase the computations by adjusting QoS to meet specific mission-dependent requirements. In a concrete setup, a mobile robot equipped with `powprofiler` can potentially take advantage of the energy model by adaptively selecting

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

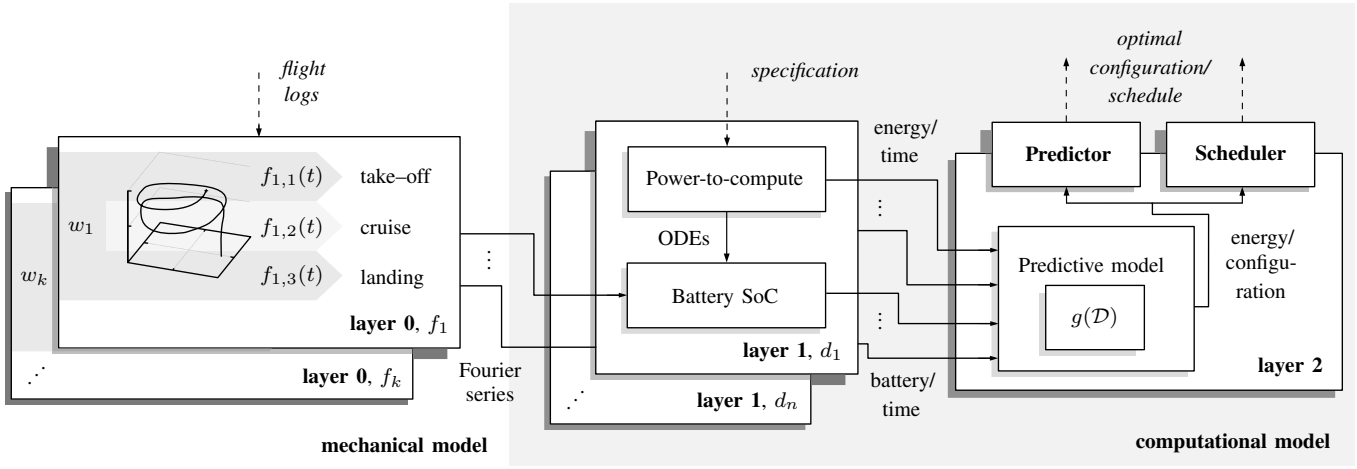


Fig. 1. Overview of the mechanical and computational energy estimation approach

different QoS levels depending on the current battery state.

The rest of this paper is organized as follows. Section II reviews the related work for energy modeling and optimization for mobile robots. Section III examines further the details of the three energy models, mechanical, computational, and battery, while Section IV assesses our experimental setup along with the results. Section V concludes our analysis and presents future work.

II. Related Work

Battery models for energy estimation have been extensively studied in the literature. A broad summary of different models was presented by Rao et al [5]. The contribution shows four classes of battery models, with different levels of detail and complexity. In the robotics domain, battery models used for mobile robots often rely on an equivalent electrical circuit, due to a linear relation between the circuit voltage and SoC, described in the work done by Pang et al. [6]. The contribution presented by Chiasson et al. [7], estimates SoC with such a circuit but does not apply the model to a use-case. Many other contributions extend Kalman Filter for SoC parameters estimation [8], [9], or focus on advanced metrics, i.e., battery state of health, and battery state of function [10]. However, they generally do not provide an energy modeling strategy for mobile robots.

Energy models for mobile robots have been previously studied by Mei et al. [1], [11], [12]. The former contribution [11] proposed energy-efficient motion planning to minimize mechanical energy. The contribution investigates efficient energy plans but does not account for computational energy. Later, Mei et al. [1] introduced the separation between the microcontroller and embedded computer for flexible mobile robot design. The contribution presents energy-conservation techniques such as dynamic power management and real-time scheduling. The last contribution [12] focused on an energy-efficient deployment algorithm in environments with obstacles. Mei et al., however, do not provide a direct battery state estimation technique, and in general, the above three contributions focus little on computational energy. Our approach influence on battery state relative to the computational part for a broad class of mobile robots but also limits the analysis of the mechanical energy for a specific fixed-wing drone. Nevertheless, the mechanical energy approach can be easily extended to a broader class of drones and the computational energy approach to potentially any class of mobile robots.

Of particular interest is the contribution by Berenz et al. [13], which outlined a dynamic recharge approach for battery management based on the mission assessment. They used a wheeled mobile robot capable of self-docking to a battery rechange station but did not focus further on computational energy awareness. Wheeled mobile robots were also investigated by Kim et al. [14], [15]. They, however, do not evaluate modeling, rather examine energy-efficient motion control (in [14]), or include optimal control theory for minimum energy trajectories (in [15]). An energy modeling approach that relies on statistical models in the absence of measurements is developed by Sadrpour et al. [16], [17]. The approach focuses on field robots and does not account for SoC.

Drones energy efficiency specifically have been also studied in the literature, as drones are typically known to be very battery dependent [4]. Most researchers focused in the past on mechanical energy, with little or none contribution on computation. To this extent, Uragun [18] suggests the use of power-efficient components. Kreciglowa et al. [19], research efficient trajectory generation methods, while Kanellakis et al. [20], suggest limiting resources for computing and sensing.

III. Energy Estimation

In this paper, energy estimation is achieved by merging three different energy models, mechanical, computational, and battery. While the first two require direct developer's interaction, the battery model is automatically generated upon the definition of battery-specific parameters and allows the evaluation of different mission dependent configurations. A mobile robot

is often equipped with at least two different computational units: a microcontroller that controls low-level real-time tasks, such as motor control, and a heterogeneous embedded device which takes care of computationally heavy operations, such as path planning, obstacle avoidance, environment mapping, and object detection.

We present a case study of an Opterra fixed-wing drone equipped with Apogee v1.00 microcontroller which uses Paparazzi UAV firmware for the autopilot [21]. Moreover, the drone is equipped with a 3200 mAh battery, suited for agricultural use-case requirements, where a drone flies on a preassigned flying route and collects images for further analysis. An NVIDIA Jetson TX2 embedded board is evaluated separately with two computationally heavy components, object detection, and encryption. Both components allow specifying different levels of QoS, that can be adapted according to the current battery state evaluated through `powprofiler` during the mission.

A. Overview

Figure 1 presents an overview of the mechanical and computational energy estimation approaches presented in this paper. From k test flights $f_1, f_2, \ldots, f_k$, different functions using a Fourier series are derived with a regression technique into layer 0 for the three flight phases. The Fourier series is used since the mission, in general, follows periodic patterns. This mechanical model is later used in the computational model, that derives into layer 2 different software configurations $d_1, d_2, \ldots, d_n \in \mathcal{D}$ (where $\mathcal{D}$ is the set of all the configurations). The resulting predictive function $g(\mathcal{D})$ in layer 2 maps configurations to the overall energy, average power, and battery SoC. The different configurations are profiled in layer 1, where the mechanical energy is merged with computational energy per configuration as it is integrated into SoC using an ordinary differential equation or ODE.

B. Mechanical Energy Model

In this subsection, a regression technique to build a mechanical energy model is described. The model is specific for an Opterra drone, but an equivalent technique can be applied to any fixed-wing drone. Key to the technique is the distinction between three phases of the flight: take-off, cruise, and landing. The distinction between these three phases follows from experimental data, as we observed that different flights present similar behavior and almost identical tendencies. This means that each phase has a different altitude and overall time evolution concerning motor torque and power drain. In particular, less variability in energy evolution is observed during the cruise and take-off. This applies when we analyze a single test flight, and when we analyze a set of flights regarding their phases. In the latter, we observed little variability in the cruise phase of different flights, as this is performed mostly by the autopilot with variability often due to wind conditions, while take-off of a fixed-wing drone usually presents a similar set of controls with little variability. Landing is observed to be the phase that presents the largest variability in the collected data, as the procedure requires specific and conditions-dependent maneuvers. Furthermore, glide slope and ground effect, specific to landing site conditions, also affect the controls sequence.

A weighted average time derived from collected data of 28 seconds, 10 minutes, and 1 minute, for respectively take-off, cruise, and landing, is assumed to model a test mission that we use in this paper. The information is used in the regression technique to map time to the current energy consumption. Regression itself consists of a phase-specific third-order Fourier series, as the data often presents a periodic behavior and tends to diverge for higher orders while not mapping precisely to the power evolution in time for lower ones. The following equation represents the power as a function of time t for each of the three phases:

$$f(t) = \sum_{n=0}^{3} a_n \cos\left(\frac{nt}{\xi}\right) + b_n \sin\left(\frac{nt}{\xi}\right), \tag{1}$$

where a_n, b_n are the Fourier series coefficients, ξ is a characteristic time, and:

$$f(t), t, a_n, b_n, \xi \in \mathbb{R}, \tag{2}$$

The Mechanical energy evolution in time can be expressed through a three-dimensional vector $\mathbf{f}$:

$$\mathbf{f}(t) = \begin{bmatrix} f_1(t) & f_2(t) & f_3(t) \end{bmatrix}^T, \tag{3}$$

where $f_1(t)$, $f_2(t)$, $f_3(t)$ are the energy evolutions in time for take-off, cruise, and landing respectively.

The mechanical energy model from Equations (1–3), can also be expressed in a matrix form:

$$\mathbf{f}(t) = \begin{bmatrix} {}^1a_0 & {}^1b_0 & \cdots & {}^1a_3 & {}^1b_3 \\ {}^2a_0 & {}^2b_0 & \cdots & {}^2a_3 & {}^2b_3 \\ {}^3a_0 & {}^3b_0 & \cdots & {}^3a_3 & {}^3b_3 \end{bmatrix} \begin{bmatrix} \cos\frac{0t}{\xi} \\ \sin\frac{0t}{\xi} \\ \vdots \\ \cos\frac{3t}{\xi} \\ \sin\frac{3t}{\xi} \end{bmatrix}. \tag{4}$$

The constants a_n, b_n, and ξ, for the regressions $f_i(t)$ with $i \in 1, 2, 3$ (and thus for $\mathbf{f}(t)$) were found using the Levenberg-Marquardt algorithm implemented in Matlab. The above procedure, however, just accounts for one flight. A further analysis concerns the generation of power trajectories through the technique for each test flight. An interpolating curve is then built using a probabilistic approach, that simply indicates a weight for a given flight k. This can be particularly useful as some flights can be affected by other conditions such as wind, temperature, or battery state at the beginning of the mission, thus affecting the model behavior unexpectedly. Given k flights and a vector of weights $\mathbf{w}$, the mechanical model can be in this fashion expressed using an equivalent expression to Equation (2):

$$\mathbf{f}(t) = \begin{bmatrix} f_{1,1}(t) & f_{1,2}(t) & f_{1,3}(t) \\ \vdots & \vdots & \vdots \\ f_{k,1}(t) & f_{k,2}(t) & f_{k,3}(t) \end{bmatrix}^T \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix}, \tag{5}$$

where w_i is the associated weight for a specific flight i, that expresses its accuracy according to the set of all the collected flights, and:

$$\begin{aligned} \mathbf{w} &\in \mathbb{R}^k, \\ w_i &\in [0,1], \\ w_1 + w_2 + \cdots + w_k &= 1. \end{aligned} \tag{6}$$

Based on a simple reordering, Equations (4–6) can be also expressed:

$$\mathbf{f}(t) = \left(\begin{bmatrix} {}^{1,1}a_0 & {}^{1,2}a_0 & {}^{1,3}a_0 \\ {}^{1,1}b_0 & {}^{1,2}b_0 & {}^{1,3}b_0 \\ \vdots & \vdots & \vdots \\ {}^{1,1}a_3 & {}^{1,3}a_3 & {}^{1,3}a_3 \\ {}^{1,1}b_3 & {}^{1,1}b_3 & {}^{1,3}b_3 \\ \vdots & \vdots & \vdots \\ {}^{k,1}a_3 & {}^{k,2}a_3 & {}^{k,3}a_3 \\ {}^{k,1}b_3 & {}^{k,2}b_3 & {}^{k,3}b_3 \end{bmatrix}^T \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} \right) \begin{bmatrix} \cos \frac{0t}{\xi} \\ \sin \frac{0t}{\xi} \\ \vdots \\ \cos \frac{3t}{\xi} \\ \sin \frac{3t}{\xi} \end{bmatrix}. \tag{7}$$

Equation (7) can be used to derive all the constants for the three phases of the flight, and thus to describe the power consumption in the function of time for take-off, cruise, and landing. Table I outlines the constants that we obtained on the Opterra fixed-wing drone test flights of the agricultural use-case.

TABLE I
CONSTANTS FOR THE MECHANICAL ENERGY MODEL

	take-off	cruise	landing
a_0	29.97	27.22	27.21
a_1	1.57	0.3737	-0.9512
b_1	-1.963	1.194	1.276
a_2	0.3876	0.6513	0.9357
b_2	-1.552	0.2954	0.965
a_3	-0.2869	0.5039	0.4913
b_3	-0.547	-0.2864	-0.1192
ξ	0.1525	0.1799	0.1296

C. Computational Energy Model

The computational energy model is a hardware-specific abstraction that maps different software configurations to energy consumption. Any software configuration can give a different energy evolution, as software in our analysis is composed of several components. Each component accepts QoS-specific parameters and hence affects the computational energy drain accordingly. As a concrete example, the object detection component accepts a parameter, the time in milliseconds, to express the period between two consecutive detections. The QoS "frames-per-second rate" can be directly mapped to this parameter with `powprofiler` that will automatically generate a computational energy model and therefore show the energy evolution for different "frames-per-second rate" configurations. The energy consumption of the drone can in this fashion be estimated by comparing current against future configuration, since when adjusting the configuration the battery SoC can change considerably (as described later in this section).

Data from the `powprofiler` tool can thus be used to estimate the energy consumption of different software configurations, once the energy model has been built from a set of profiled trials. This is a process that `powprofiler` automatically takes charge of. The tool expects a specification file that specifies the components along with QoS ranges (the specification is further described in our previous work [4]). The tool first generates a layer 1 model, which maps time to energy consumption for each configuration. A simple statistical approximation technique based on a weighted average is performed to produce a layer 2 model, which maps energy consumption to all the configurations within the QoS range. The output is a comma-separated values file, but the model can also be queried through an API.

The computation can be adjusted dynamically in the sense that the system can evaluate how different configurations will affect the energy consumption, and by implementing a scheduling strategy accordingly (we presented an example of a component-based scheduling strategy in our previous work [2]). The computational energy model could furthermore be integrated into a component-based framework, such as the ROS middleware, or the TeamPlay components toolchain [22].

D. Battery Model

Battery requirements are a key aspect of almost any mobile robot as the completion of a mission is often limited by the available amount of energy [14]. We present a battery abstraction in this subsection to: **a)** evaluate the effect of different computational models, and **b)** estimate the impact of mechanical energy on the battery SoC. The final objective of battery SoC estimation is to predict the energy consumption of different software configurations.

Given following general expression for the battery SoC:

$$\frac{d}{dt}\mathrm{SoC}(t) = -\frac{I_{\mathrm{int}}(t)}{Q_c}, \tag{8}$$

where I_{int} is the current load, Q_c the constant nominal capacity, and:

$$\mathrm{SoC}(t), I_{\mathrm{int}}(t), Q_c \in \mathbb{R}, \tag{9}$$

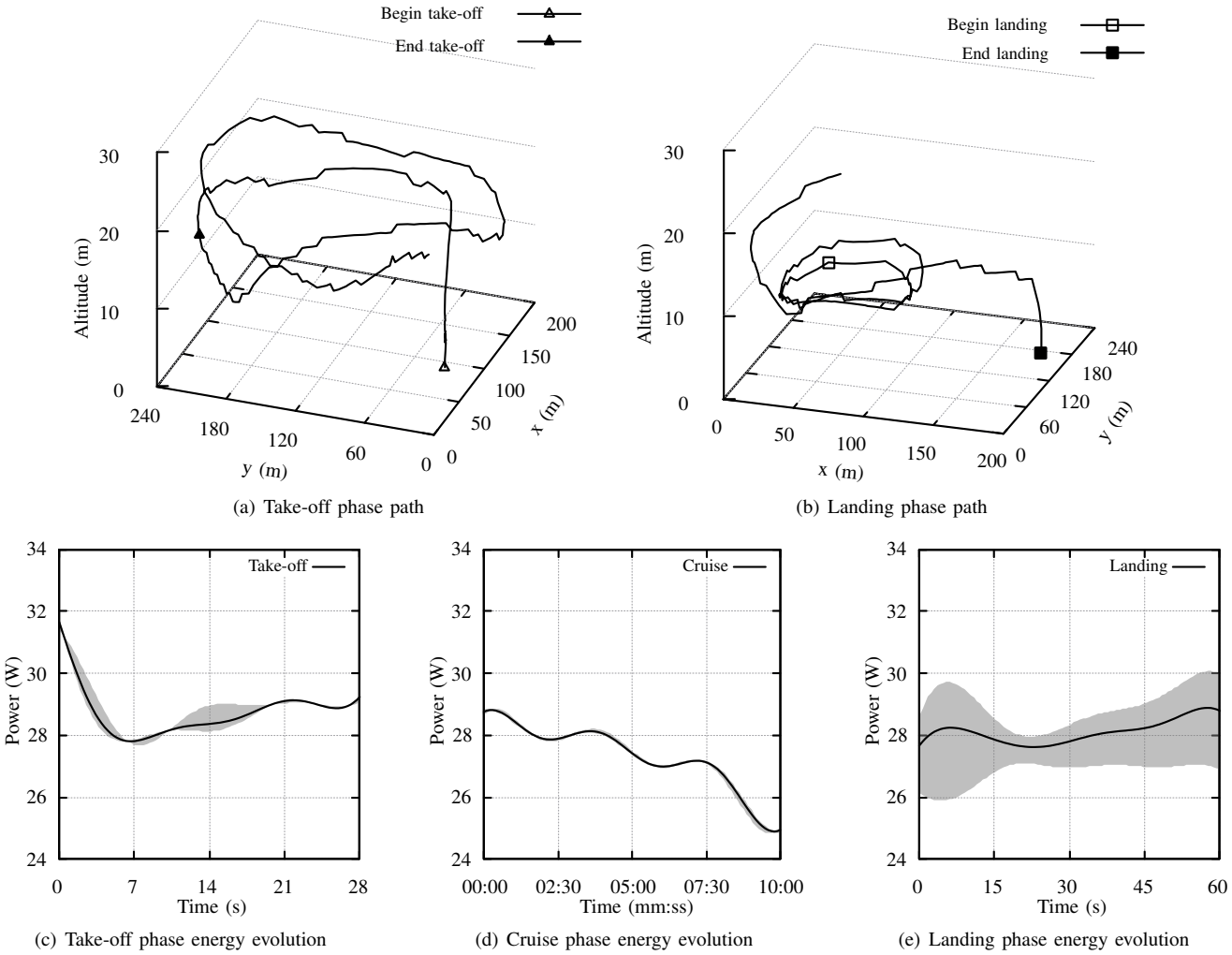
the battery model can be derived from an equivalent electrical circuit, like the one presented by Hasan et al. [9]. Such a model consists of an ODE which describes the evolution in time of battery SoC with the following expression for the current load:

$$I_{\mathrm{int}}(t) = \frac{U_{\mathrm{int}} - \sqrt{U_{\mathrm{int}}^2 - 4 \cdot R_{\mathrm{int}} \cdot U_{\mathrm{sta}} \cdot I_{\mathrm{load}}(t)}}{2 \cdot R_{\mathrm{int}}}, \tag{10}$$

where U_{int} is the internal battery voltage, R_{int} the internal resistance, U_{sta} the stabilized voltage, I_{load} the current required by the load, i.e., the mechanical and computational energy, and:

$$I_{\mathrm{load}}(t), U_{\mathrm{int}}, R_{\mathrm{int}}, U_{\mathrm{sta}} \in \mathbb{R}. \tag{11}$$

All the constants presented above in Equations (8–11) were chosen to satisfy the requirements of an embedded device.



(a) Take-off phase path

(b) Landing phase path

(c) Take-off phase energy evolution

(d) Cruise phase energy evolution

(e) Landing phase energy evolution

Fig. 2. The evolution in time of energy and paths for three different flight phases. take-off, cruise, and landing

A classic Runge-Kutta iterative method is implemented in `powprofiler` to solve the ODE from Equations (8–11). The initial value of 100% is in this way integrated over time with the computational model from different configurations. Specifically, the computational energy drain is transformed into the current drain $I_{\text{load}}(t)$ from Equation (10). Results are stored in a layer 2 model that can be further analyzed to estimate overall energy. Our initial approach towards numerical integration in `powprofiler` tool included Runge-Kutta-Nyström modfication [23], as a second-order ODE was used initially to describe the equivalent electrical circuit (a solution that turned out to be unnecessary for the energy estimation in this paper).

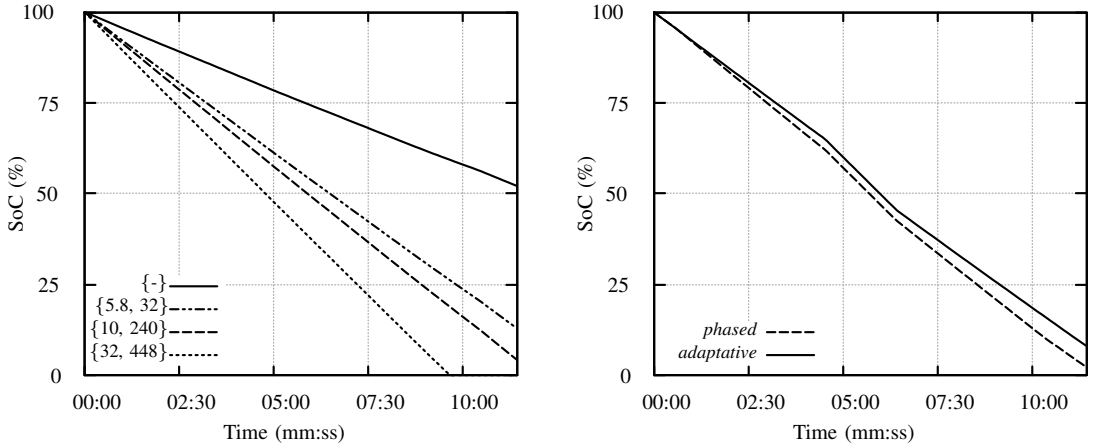
IV. Evaluation

Mechanical, computational and battery energy models are evaluated in this Section using respectively: **a)** the Opterra fixed-wing drone performing an agricultural use-case introduced in Section III, **b)** the Jetson TX2 board for the object detection featuring YOLO neural network [24], and symmetric encryption featuring Blowfish algorithm [25], and **c)** the use of the battery model presented in Equations (8–11) to determine the SoC evolution throughout the mission of the drone.

A. Mechanical Model Evaluation

The mechanical energy model presented in Subsection III-B, was evaluated by analyzing flight logs and deriving the constants from Equations (5–7). Test flights were performed featuring the agricultural use-case in different times, although in the same hardware and software configuration. The flight logs were retrieved through the Paparazzi UAV autopilot firmware and were further analyzed with Matlab. Different flight phases $f_1(t)$, $f_2(t)$, $f_3(t)$ were first defined for each flight, later merged in three Fourier series that map time to energy consumption for take-off, cruise and landing respectively.

Figure 2(a) shows first how the take-off phase for a flight was derived. To obtain this information we analyzed the altitude, motor torque, and power drain. Specifically, the drone takes altitude in the first part of the take-off phase and inserts itself into the circular trajectory later performed during the cruise phase. Figure 2(c) shows the energy evolution (in black)



(a) SoC as a function of time for different configurations

(b) Two examples of the evolution of SoC-aware schedules

Fig. 3. Different SoC evolutions in time

for $f_1(t)$, which means that the Fourier series is calculated using a number of test flights. The energy consumption of those evolves in the gray section of the plot, i.e., the gray section indicates all the test flights, while the black indicates the take-off mechanical energy model. Little variability can be observed during the first part of the take-off phase, as different weather conditions often result in different take-off controls. Very little or no variation is observed in the latter part, corresponding to the insertion into the circular trajectory, as this is the part where autopilot takes control over the human operator.

Similarly, the cruise phase is modeled through $f_2(t)$ as shown in Figure 2(d). There is very little variability in the different flights' evolution, as the cruise phase is always performed by the autopilot in our analysis. An overall descent tendency can be observed, with fewer watts drained from the battery. This is the result of a gradual battery depletion during the flight and is common to all the flights that we analyzed.

The landing phase $f_3(t)$ presents the most variable behavior, as shown in Figure 2(e). A landing trajectory of a flight can be seen in Figure 2(b), where first the drone performs small circles lowering the altitude, and later descends to the ground under control of a human operator. The phase is very dependent on different conditions, and thus requires direct adjustment with regard to the current landing site and/or wind conditions. From the plot, we can observe major variability at the beginning where the power drain is dependent on the current altitude, and at the end, where it is mostly dependent on the landing site conditions.

B. Computational Model Evaluation

The computational energy model has been extensively evaluated for different embedded boards in our previous work [4]. Here we focus on modeling the energy estimation of different software configurations, as our case study is composed of two software components that present different ranges of QoS. The model is built for object detection and symmetric encryption algorithms using the `powprofiler` tool.

The object detection component has been modeled in the range between 5.8 and 32 frames, representing the QoS "frames-per-second rate". The resulting energy model shows the evolution of energy as a function of FPS, such that a higher frames-per-second rate results in higher overall energy consumption. The data were profiled and modeled with `powprofiler`, which automatically evaluated four ranges of frames per second rates, 5.8, 10, 25 and 32, and built a statistical model for the missing ones. The analysis on the encryption component was done in the range between 32 and 448 bits, representing the QoS range "key-size". We used Blowfish symmetric and variable-key encryption algorithm for this purpose [25], and the OpenSSL command-line tool to perform cyclic encryption of a heavy data-file of approximately 150 MB [26]. Both object detection and encryption were further merged arithmetically, to model battery SoC of using the two components concurrently (Figure 4). In a concrete implementation of the computational energy model, a layer 2 model can be automatically derived through `powprofiler` for either separate components that are subsequently analyzed (an approach that we investigated in our previous work [2]) or for components running in parallel. In the latter, the modeling time is higher since components combinations are evaluated empirically.

C. Mission Assesment

The computational energy model can be correlated to the mechanical energy model with the battery SoC using `powprofiler` which estimates the current battery state using Equations (8–11). The analysis holds under the assumption that SoC is approximately linear in the considered interval. Concretely, the mechanical model can be included into a `powprofiler` model that describes the current mechanical energy evolution in time through a discrete set of values (an operation that can be computed by, e.g., using Matlab: first a csv file can be exported from Matlab, this file can then be included into the model). The information is used to correlate the current software configuration to the remaining battery and

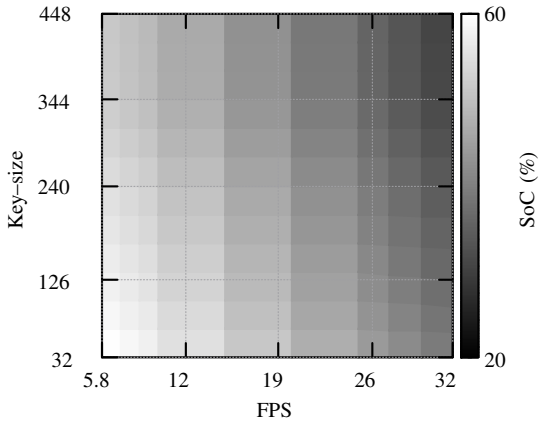


Fig. 4. Estimated SoC as a function of frames-per-second rate and key-size QoS ranges after completing a mission

thus to estimate the future energy cost. Data from Figure 4 shows how the battery SoC changes as a function of different configurations. In general, the higher the FPS rate and key-size, the larger the impact on the battery. SoC information can be evaluated on any initial conditions, allowing the evaluation of a specific configuration on the current battery state. In this paper we assume a fixed dependency on mechanical energy, meaning only the computational energy can be affected by varying QoS ranges.

Figure 3 shows the energy estimation (both computational and mechanical) of SoC using different schedules. First, Figure 3(a) shows an invariant schedule. The solid line indicates the mechanical energy model, with no computations included. The model is almost linear, as the power drain varies just in a small interval at the beginning (cf. Figure 2(c)), and then presents little variability (cf. Figures 2(d), 2(e)). The other lines of Figure 3(a) show the evolution of mechanical and computational models together with configurations of 5.8 FPS and 32 bits, 10 FPS and 240 bits, and 32 FPS and 448 bits respectively.

Figure 3(b), shows the energy estimation (mechanical and computational) of two variable schedules. The *phased* schedule uses 5.8 FPS and 32 bits configuration during take-off and landing, 32 FPS and 448 bits configuration for two minutes during the cruise phase (exactly in the middle), and 10 FPS and 240 bits configuration otherwise. A resulting SoC of 2.05% was determined using our model. The *adaptive* schedule uses 32 FPS and 448 bits configuration for two minutes during the cruise phase, and 5.8 FPS and 32 bits configuration otherwise, with a resulting SoC of 7.86%.

Using a battery model allows the evaluation of some battery-specific behaviors, as the battery evolution in time is non-linear. For instance, in some particular schedules (described in our previous work [4]) the battery SoC can be different even if the overall energy consumption is equal (the battery behaves better with a constant power drain against a spiked one).

V. Conclusion and Future Work

In this paper, we presented a case study for the energy estimation of a fixed-wing drone. Although mechanical energy data are assessed using a number of flights with an Opterra fixed-wing drone, our approach towards mechanical energy modeling can easily be extended to other drones. We modeled mechanical energy using a Fourier series for three phases of the flight with major variability, for take-off, cruise, and landing. Computational energy is estimated using an empirical approach, where software components are evaluated for specific ranges of QoS and approximated for others. Computational and mechanical models are generated using the `powprofiler` tool and automatically integrated into the battery model to obtain battery SoC and thus to estimate the cost of different software components configurations. Our approach allows the evaluation of dynamic scheduling decisions, as the drone might limit computations to be able to complete the mission.

In terms of future work, we aim to define automatic dynamic scheduling options (the schedule is currently statically generated) and evaluate them on a flying drone. A direct ROS interface for `powprofiler` is also being considered, as many mobile robotic systems rely on ROS middleware. A variable definition of the mechanical model will be included later in our analysis since dynamic path planning against current computation can considerably extend our approach, enabling a complete energy awareness for both motion and computation. Such a model should also take into account the effect of the surrounding environment, such as weather conditions.

Acknowledgment

This work is supported and partly funded by the European Unions Horizon2020 research and innovation program under grant agreement No. 779882 (TeamPlay). The authors thank Amit Ferenz Appel for the contribution to the training flights used in the mechanical energy model in this paper.

References

[1] 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.

[2] 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.

[3] B. Rouxel, R. Julius, A. Sebastian, and G. Clemens, "A time, energy and security coordination approach," in *10th International Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS 2019)*, 2019.

[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] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery modeling for energy aware system design," *Computer*, vol. 36, no. 12, pp. 77–87, 2003.

[6] S. Pang, J. Farrell, J. Du, and M. Barth, "Battery state-of-charge estimation," in *Proceedings of the 2001 American control conference.(Cat. No. 01CH37148)*, vol. 2. IEEE, 2001, pp. 1644–1649.

[7] J. Chiasson and B. Vairamohan, "Estimating the state of charge of a battery," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 4. IEEE, 2003, pp. 2863–2868.

[8] M. Partovibakhsh and G. Liu, "An adaptive unscented kalman filtering approach for online estimation of model parameters and state-of-charge of lithium-ion batteries for autonomous mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 357–363, 2014.

[9] A. Hasan, M. Skriver, and T. A. Johansen, "Exogenous kalman filter for state-of-charge estimation in lithium-ion batteries," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 1403–1408.

[10] F. Zhang, G. Liu, and L. Fang, "Battery state estimation using unscented kalman filter," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1863–1868.

[11] 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.

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

[13] V. Berenz, F. Tanaka, and K. Suzuki, "Autonomous battery management for mobile robots based on risk and gain assessment," *Artificial Intelligence Review*, vol. 37, no. 3, pp. 217–237, 2012.

[14] 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.

[15] 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.

[16] 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.

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

[18] B. Uragun, "Energy efficiency for unmanned aerial vehicles," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 2. IEEE, 2011, pp. 316–320.

[19] 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.

[20] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, 2017.

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

[22] "Public deliverables of the TeamPlay Horizon2020 project," https://teamplay-h2020.eu/index.php?page=deliverables, 2019, accessed: 2019-08-25.

[23] B. Paternoster, "Runge-kutta (-nyström) methods for odes with periodic solutions based on trigonometric polynomials," *Applied Numerical Mathematics*, vol. 28, no. 2-4, pp. 401–412, 1998.

[24] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[25] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *International Workshop on Fast Software Encryption*. Springer, 1993, pp. 191–204.

[26] J. Viega, M. Messier, and P. Chandra, *Network security with openSSL: cryptography for secure communications*. O'Reilly, 2002.