Trade Offs Between Energy Consumption and Control Quality in Unmanned Aerial Vehicles

Paulo H. Vancin*, Rafael F. Garibotti†, Luciano C. Ost‡, Ney L. V. Calazans†§, Fernando G. Moraes*
* School of Technology, Pontifical Catholic University of Rio Grande do Sul – PUCRS – Porto Alegre, Brazil
† PGMICRO, Federal University of Rio Grande do Sul – UFRGS – Porto Alegre, Brazil
‡ Wolfson School, Loughborough University – Loughborough, U.K.
§ Computing Department - DEC, Federal University of Santa Catarina – UFSC – Araranguá, Brazil
phvancin@hotmail.com, rfgaribotti@inf.ufrgs.br, l.ost@lboro.ac.uk, nlvcalazans@inf.ufrgs.br, fernando.moraes@pucrs.br

***Abstract*—Unmanned Aerial Vehicles (UAVs), particularly autonomous versions, are increasingly relevant in several fields, including human activity, nature observation, and actuation. This work explores an often overlooked question: the impact of flight control algorithms and their parameterization on the energy efficiency of UAV electronics. Using a highly configurable and comprehensive simulation environment, it evaluates the energy required to run specific control algorithms with varying parameters in a processor, while other UAV tasks are accomplished in other processors of a multiprocessor core. Experiments in energy saving mode show it is feasible to decrease the energy taken to run the control algorithm alone by 16.67%, while a maximum performance mode allows for more precise UAV operation, whenever required.**

***Index Terms*—Unmanned Aerial Vehicles, UAVs, Robotics, Energy efficiency, Flight control algorithms.**

I. Introduction

Interest in Unmanned Aerial Vehicles (UAVs) has surged in recent years due to their ability to perform difficult or dangerous tasks with high mobility, safety, and low cost in services such as real-time monitoring, search and rescue, packet delivery, and precision agriculture, among others [1, 2]. UAVs, particularly autonomous ones, are complex devices that rely on the integration of advancements from several fields, such as precision mechanics, electromechanical components, and advanced electronics. Integrating these advanced components into an efficient device is challenging, especially since the design of any one part can affect the efficiency of others.

Advanced control algorithms are at the heart of efficiency, since autonomous UAVs are obviously more complex to operate than autonomous surface devices. Besides, efficient implementations of control algorithms in advanced electronics are a complex task [3]. Separately addressing control algorithms and electronic design for UAVs may get device designs stuck in local optima at any or at both these design aspects.

Energy efficiency is crucial in UAVs, mainly due to the critical nature of UAV power supplies, which are often severely limited in weight and size [4]. Obtaining more work from less available energy is crucial to achieving higher levels of autonomy and ensuring the success of UAV missions. Accordingly, this work addresses the particular question of UAV flight control algorithms energy efficiency regarding the device electronics. It investigates the influence of UAV control algorithm parameterization, the related achieved operation precision levels and the trade offs of such parameterization on the UAV electronics energy consumption.

This work was financed in part by FAPERGS (grant 21/2551-0002047-4) and CNPq (grants 309605/2020-2, 317087/2021-5, 311587/2022-4 and 407477/2022-5).

II. Related Work

UAV energy efficiency is a major concern in several research works [5–7]. Most works usually address higher levels of abstraction and modeling, overlooking aspects such as the impact of energy on UAV electronics during missions. Consequently, the existing literature lacks comprehensive coverage of the relationship between control algorithms execution parameters and energy consumption on UAVs.

Boukoberine *et al.* [8] provide a comprehensive review of power supply and energy management solutions, including in-flight recharging, battery swapping, and tethering schemes, with a focus on hybrid power supply UAVs, i.e., those combining multiple power sources like batteries and solar fuel cells. The authors demonstrate that an optimal energy management system and hybridized power source architecture are crucial for efficient UAV operation. The paper concludes that power supply hybridization is the best architecture for powering UAV propulsion systems, though specific energy-control trade-offs are not addressed.

Prasetia *et al.* [9] propose a method to predict energy consumption for multirotor UAV missions. The UAV training power model includes data collection, preprocessing and regression phases. Authors claim their UAV black box method displays more than 98% average accuracy for missions, from take-off to base return. Furthermore, the authors show that the proposed method achieves an accuracy exceeding 10% compared to other works in the literature. No optimization for energy consumption is proposed, a given UAV setup is used only to reach a precise energy consumption estimation.

Similar to Prasetia *et al.* [9], Beigi *et al.* [10] are concerned with energy models for predicting power consumption. However, they diverge in their approach, with the latter committing to explore four main factors that influence UAV energy consumption: (*i*) UAV design; (*ii*) operating environment; (*iii*) drone dynamics; and (*iv*) delivery operations. The study limits itself to delivery mini drones operating on lithium polymer (LiPo) batteries, and their associated energy models.

Chodnicki *et al.* [11] propose an energy-efficient method for controlling fixed-wing UAVs through algorithms for route planning, in-flight control, and trajectory correction, tested on an Arm processor-based architecture using hardware-in-the-loop techniques. The solution uses two computers: one for flight control with a real-time operating system (RTOS) and another for mission control with Linux and the Robot Operating System (ROS). The main contribution is to integrate planning and monitoring methods to minimize UAV energy consumption, considering wind conditions.

In contrast to previous work, this paper *contributes* by assessing flight control algorithms in terms of both energy efficiency and performance, as well as delving deeper into specific control parameterization aiming at saving power consumption.

III. A Drone Computing System Architecture

The Robot Operating System (https://www.ros.org/) has emerged as the *de facto* standard for developing robotic systems. ROS-based systems implement a publish-subscribe pattern [12], organized into *nodes* and *topics*. A *node* is a software capable of subscribing to (or publishing on) data channels known as *topics*. In typical robot simulation setups, ROS is often combined with Gazebo (http://gazebosim.org/), a robot simulator capable of simulating environmental physics, e.g., gravity, wind, terrain, and sunlight. Robots interact with the environment through sensors, capturing data based on the stimuli of the simulated environment. Gazebo can be extended to feed data into ROS, enabling applications to access sensory information. This allows programming and validation of robots' behavior as if the software run on a real robotic platform. Without loss of generality, this work addresses quadrotor UAVs, often used as autonomous UAVs.

A. Target Architecture

Figure 1 shows the system's architecture used in this work. This is a simulation environment that runs on Linux platforms. The system is divided into two main components: ROS and the URSA simulator [13]. ROS is responsible for simulating the quadrotor and establishing communication with the ORCA Multiprocessor system-on-a-chip (MPSoC). URSA is a dedicated simulator for the ORCA MPSoC, which operates the quadrotor's control system within its processing cores [14].

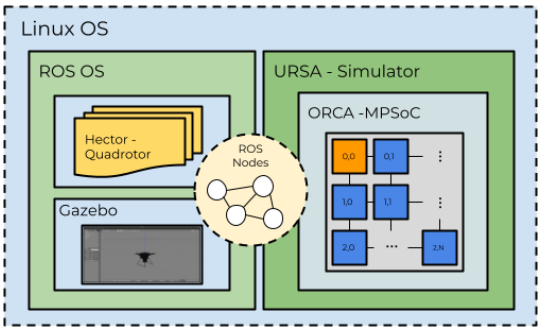

Fig. 1: Overall representation of the target architecture.

The target architecture experimental setup was designed using two workstations (WSs), refer to Figure 2. The first is a ROS Master, which hosts the URSA simulation and the ROS communication nodes, while the second is a ROS Client that runs the Hector quadrotor simulation, supported by Gazebo. This allows maximizing the simulation processing time, mostly due to the heavy computation demands of the Hector quadrotor and URSA simulations.

Fig. 2: The target architecture experimental setup.

For the experimental setup to run ROS on multiple workstations (WSs), the following rules must be applied: (*i*) all WSs, including the ROS Master and all ROS Clients, must be on the same network; (*ii*) in a multi-WS system, a single WS should be designated as the Master, with multiple client WSs allowed; (*iii*) all WSs must identify the Master using the ROS_MASTER_URI configuration, which informs nodes of the Master's location; (*iv*) all WSs should be capable of running multiple ROS nodes.

The experiments focus on two control algorithms. The first is the Extended Kalman Filter (EKF), which, using quaternions, determines the quadrotor's attitude by providing a recursive method for state estimation in a dynamic system with noise. The second algorithm is the proportional-integral-derivative (PID) control, which computes control laws based on the Euler angles of the quadrotor's state.

B. The ROS-URSA Interface

Figure 3 illustrates the ROS node structure present in the experimental setup, composed by *network* and *processing* nodes. This shows how Hector communicates with the ORCA MPSoC. According to [13], networking nodes include hardware from the processing node. However, these have a network bridge connected to the network interface (NI) rather than a processor core and main memory. The network bridge translates raw user datagram protocol (UDP) packets, including UDP/IP headers and additional information, into the MPSoC network-on-chip (NoC) protocol, by adding required headers. Translation occurs when transmitting data from the NoC to the UDP network and vice-versa.

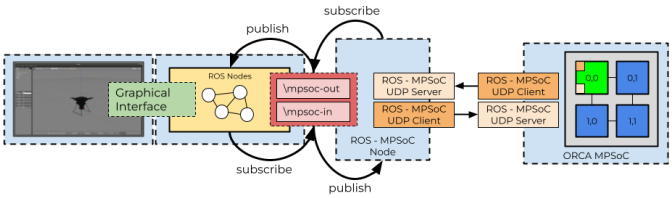

Fig. 3: The experimental setup network and processing nodes and their interaction.

Note that the ROS-MPSoC node exists to establish a communication link between ORCA and ROS. This node wraps the communication with the MPSoC and exposes the MPSoC as a resource to the entire system through topics. This node operates with two concurrent threads. The first handles messages from the ROS systems to the MPSoC, and the second manages UDP packets from the MPSoC.

IV. Energy-Control Experiments

Energy estimation here covers three primary MPSoC components: the processor, the NoC, and memory. Processor energy and power characterization rely on executed instruction types, with measurements segmented into leakage and dynamic power. Netlist simulations enable a calibration process obtained from switching activity. NoC router energy characterization considers two scenarios: idle power consumption and per-port activity during data transmission. The CACTI-P tool [15] allow assessing memory energy during reads and writes. The MPSoC internally monitors individual processing

cores consumption, periodically sending data to a manager core for real-time computation of system power and energy.

This Section proposes an energy management technique software implementation. It does not require that the user change any hardware settings, only dealing with data input frequency changes. As observed in previous sections, the cores only run tasks once new data is incoming. Meanwhile, a core is kept idle, running only a function that detects if new data have arrived. Therefore, it is possible to control the overall energy consumed by the core, by spacing its data injection frequency. Figure 4 shows the overall MPSoC use.

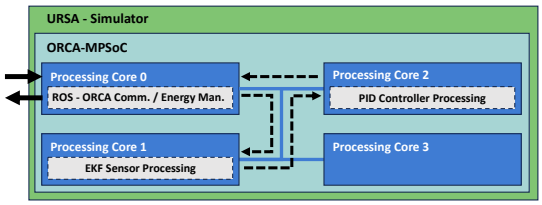

Fig. 4: MPSoC configuration for the experiments.

Experiments rely on the fact that running the system at full power is not always needed. Consider the use of EKF: in hover mode, changes on the Euler angles are minimal; therefore, running the attitude estimation at its maximum performance is unnecessary. With the quadrotor in motion mode, attitude estimation becomes more relevant, demanding faster processing speed and thus more energy.

Experiments evaluate two modes of operation: energy saving and maximum performance. In energy saving mode, the data injection frequency into the EKF is reduced, which minimally impacts estimation quality during tasks like hovering. In maximum performance mode, the data injection frequency is maximized. Four energy management experiments are proposed: (*i*) system is in energy saving mode while the quadrotor is hovering; (*ii*) system is in maximum performance mode while the quadrotor is hovering; (*iii*) system is in energy saving mode while the quadrotor is in motion; and (*iv*) system is in maximum performance mode while the quadrotor is in motion.

A. Hovering in Energy Saving and Maximum Performance

In a first experiment, the data injection rate in the EKF is kept at 12Hz. This means that the core running the EKF is idle longer, consuming less energy. Such low frequency naturally reduces the control precision. Figures 5 and 6 show the "QX" and "QY" (resp. the Roll and Pitch) quaternion components output while the system runs on saving mode while hovering.

An examination of Figures 5 and 6, reveals that significant latency is observed in the roll and pitch dynamics, manifesting as substantial jitter, which serves as an indicator of marginal stability. Results pertaining to height control and yaw are omitted, as they yielded satisfactory outcomes, demonstrating stability in those.

The next experiment consists in increasing the data injection rate in the EKF to 40Hz. This means that the core running the EKF is kept idle for shorter times, taking up more energy. This higher frequency increases the control precision. Figures 7 and 8 show the quaternion components "QX" and "QY" (representing Roll and Pitch) output while the system runs on this maximum performance mode in hovering.

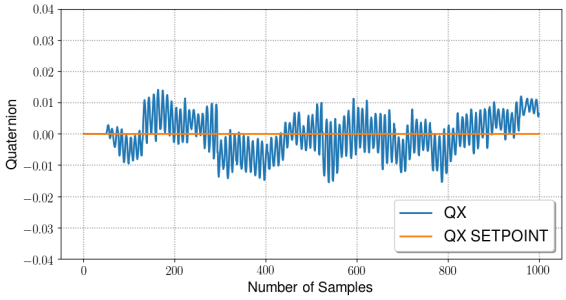

Fig. 5: The Roll component resulting from the PID control running in the **energy saving** mode while **hovering**.

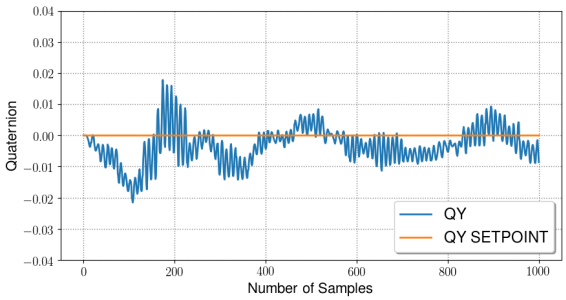

Fig. 6: The Pitch component resulting from the PID control running in the **energy saving** mode while **hovering**.

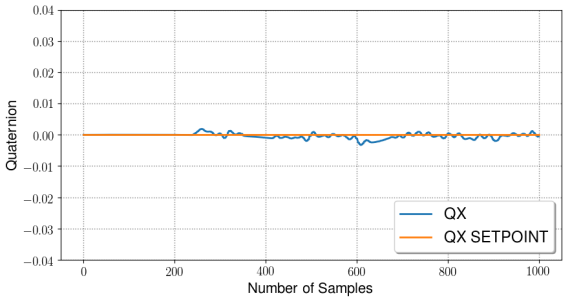

Fig. 7: The Roll component resulting from the PID control running in the **maximum performance** mode while **hovering**.

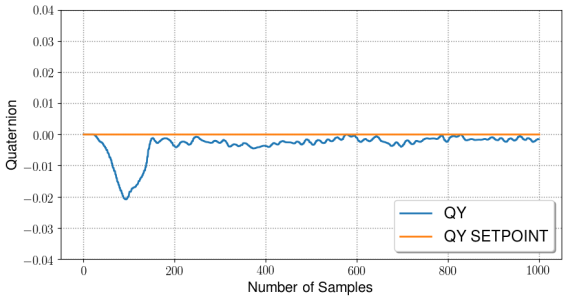

Fig. 8: The Pitch component resulting from the PID control running in the **maximum performance** mode while **hovering**.

Upon scrutiny of Figures 7 and 8, a noticeable enhancement is discernible when comparing the test results depicted in Figures 5 and 6. The reduced latency within this system proves instrumental in achieving superior stability for roll and pitch

angles, with minimal to negligible variations. Nonetheless, in the sample range 0 to 200 a conspicuous error in component QY is perceptible, which is due to the fast quadrotor ascent. Results of height and yaw control are omitted, since they exhibit satisfactory performance and display no discernible improvements between the two tested modes.

B. Energy Saving and Maximum Performance in Motion

This section evaluates the system's ability to maintain the quadrotor in a fixed position during flight, both in energy-saving and maximum performance modes. The two experiments aim to stabilize the quadrotor at a height of 1 meter, followed by hovering at a fixed XY point. The quality of the results is measured by the maximum error in both axes. Figure 9 provides a graphical representation of these experiments.

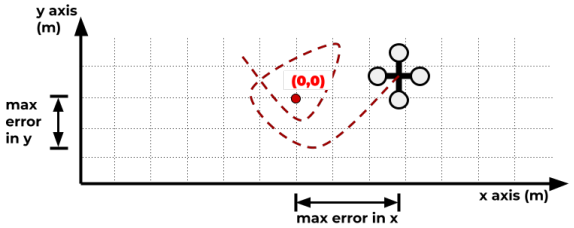

Fig. 9: Overall representation of the motion experiments.

Table I shows the maximum errors observed in the experiments. Notably, the quadrotor's overshoot error decreased from 0.75 m and 0.6 m to 0.1 m and 0.05 m. It's worth mentioning that the quadrotor's behavior was similar in both experiments; it attempted to stop at the set point but continued to overshoot, with the only difference being the distance of this overshoot in the X and Y directions.

TABLE I: Maximum error regarding the position of the quadrotor relative to the setpoint (X,Y), for both operation modes.

Operation Mode	Max Error in X axis (m)	Max Error in Y axis (m)
Energy Saving	0.75	0.6
Maximum Performance	0.1	0.05

C. Energy Consumption Analysis

Consider the energy estimation for both flight modes: energy saving and maximum performance. Table II shows the accumulated energy estimated in the core running the PID task in the energy saving and maximum performance modes. The energy saving mode consumed 2.5 mJ along the whole experiment. The maximum performance mode implied an overall consumption of 3 mJ. This represents a 16,67% drop in energy consumption only by changing the data injection rate in one of the MPSoC cores.

TABLE II: Accumulated energy estimated in the core running the PID task in energy saving and maximum performance modes.

Operation Mode	Energy Consumed (mJ)
Energy Saving	2.5
Maximum Performance	3.0

In practical terms, this shows the possibility for UAV systems to operate in several modes, depending on the specific application, on environmental conditions (wind gusts, etc.), as well as on power, performance requirements or on battery charge conditions. The experiments show that it is safe to operate in saving mode while in take-off and hovering. When it is needed to move, the maximum power setting is triggered, given the conditions allowing it are present.

V. Conclusions and Future Work

This work exemplifies how the joint consideration of control algorithms and electronics can be beneficial to the energy-efficient operation of UAVs. The use of a powerful modeling and simulation environment, comprising ROS, Gazebo and URSA enabled the collection of detailed runtime information about specific UAV control and electronics settings and their effects on operation and energy consumption.

The employed modeling and simulation environment allows several more experiments. Examples are: (*i*) evaluating the effect of using different control algorithms such as Linear Quadratic Regulators (LQR) or Model-Predictive Controller (MPC); (*ii*) examining alternative scheduling of control algorithms in the MPSoC (e.g. parallel versus distributed algorithm distributions); (*iii*) exploring fault tolerance in UAVs, by replicating controllers in multiple available cores with several strategies; (*iv*) assessing scenarios where switching modes during flight leads to improved performance and energy savings.

References

[1] N. K. Yang *et al.*, "A Novel Approach for Real Time Monitoring System to Manage UAV Delivery," in *IIAI-AAI*, 2016, pp. 1054–1057.

[2] P. K. Reddy Maddikunta *et al.*, "Unmanned Aerial Vehicles in Smart Agriculture: Applications, Requirements, and Challenges," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 608–17 619, 2021.

[3] N. Passalis *et al.*, "Efficient Camera Control using 2D Visual Information for Unmanned Aerial Vehicle-based Cinematography," in *ISCAS*, 2018, pp. 2346–2350.

[4] F. Wang *et al.*, "Research on Energy Optimal Control Strategy of DC PV-Energy Storage System for Unmanned Aerial Vehicle," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 3, pp. 2643–2651, 2021.

[5] X. Dai *et al.*, "Energy-Efficient UAV Communications: A Generalized Propulsion Energy Consumption Model," *IEEE Wireless Communications Letters*, vol. 11, no. 10, pp. 2150–2154, 2022.

[6] Y. Zeng and R. Zhang, "Energy-Efficient UAV Communication With Trajectory Optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[7] Y. K. Tun *et al.*, "Energy-Efficient Resource Management in UAV-Assisted Mobile Edge Computing," *IEEE Communications Letters*, vol. 25, no. 1, pp. 249–253, 2021.

[8] M. N. Boukoberine *et al.*, "A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects," *Applied Energy*, vol. 255, p. 113823, 2019.

[9] A. S. Prasetia *et al.*, "Mission-Based Energy Consumption Prediction of Multirotor UAV," *IEEE Access*, vol. 7, pp. 33 055–33 063, 2019.

[10] P. Beigi *et al.*, "An Overview of Drone Energy Consumption Factors and Models," Jul. 2022, Available at https://arxiv.org/abs/2206.10775v2.

[11] M. Chodnicki *et al.*, "Energy Efficient UAV Flight Control Method in an Environment with Obstacles and Gusts of Wind," *Energies*, vol. 15, no. 10, pp. 1–31, 2022.

[12] G. Coulouris *et al.*, *Distributed Systems: Concepts and Design*, 5th ed. Pearson, 2011.

[13] A. Domingues, "URSA: A framework to the simulation of multiprocessor platforms," 2019, https://github.com/andersondomingues/ursa.

[14] P. H. Vancin *et al.*, "Towards an Integrated Software Development Environment for Robotic Applications in MPSoCs with Support for Energy Estimations," in *ISCAS*, 2020, pp. 3963–3967.

[15] S. Li *et al.*, "CACTI-P:Architecture-Level Modeling for SRAM-based Structures with Advanced Leakage Reduction Techniques," in *ICCAD*, 2011, pp. 694–701.