Why Compute Matters for UAV Energy Efficiency?

Behzad Boroujerdian* Hasan Genc* Srivatsan Krishnan* Aleksandra Faust‡ Vijay Janapa Reddi*‡

***Abstract*—Unmanned Aerial Vehicles (UAVs) are getting closer to becoming ubiquitous in everyday life. Although the researchers in the robotic domain have made rapid progress in recent years, hardware and software architects in the computer architecture community lack the comprehensive understanding of how performance, power, and computational bottlenecks affect UAV applications. Such an understanding enables system architects to design microchips tailored for aerial agents. This paper is an attempt by computer architects to initiate the discussion between the two academic domains by investigating the underlying compute systems' impact on aerial robotic applications. To do so, we identify performance and energy constraints and examine the impact of various compute knobs such as processor cores and frequency on these constraints. Our experiment show that such knobs allow for up to 5X speed up for a wide class of applications.**

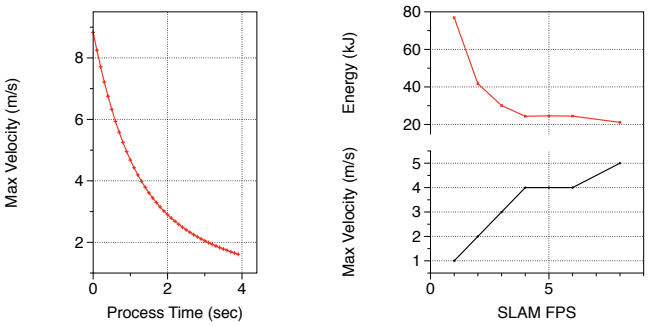
I. Introduction

Unmanned aerial vehicles are becoming an important part of our technological society. With myriad use cases, such as surveillance [1], search and rescue [2], [3], package delivery [4]–[6], and more, these unmanned aerial systems are on the cusp of demonstrating their full potential. But despite their myriad use cases, their limited on board energy pose them with a real challenge manifested in their low endurance. For example some off-the-shelf micro drones have endurance of less than 20 minutes, and a flight range of about 15 miles [5].

To practically deploy drones, both their endurance and range must be improved through not only the use of better algorithms, but also with employment of more powerful on board computers. In this paper we investigate the role of computing given the challenges and demands. More specifically, we demonstrate how a powerful compute subsystem can mitigate the endurance problem by improving how fast a drone can maneuver, fly, and efficiently finish its mission. Without loss of generality we use quad rotor based micro aerial vehicles (MAVs) for our studies due to their popularity.

We start by first establishing the theoretical relationship between processing time and the MAV's ability to fly fast (Section II). Next, we show the measured power results for the difference in power consumption between compute and total system power, including rotors, etc. We intend to establish the point that a naive interpretation of the results can lead to wrong conclusions about the relationship between compute and total system power consumption (Section III). After that, we introduce our experimental simulation infrastructure (Section IV), on top of which we have implemented several different realistic and holistic end-to-end UAV applications. Using the AirSim-based infrastructure, we demonstrate the 5X impact that compute can have total flight time efficiency (Section V). Finally, we close the paper with potential for future work toward developing energy-efficient UAVs (Section VII).

*Department of Electrical and Computer Engineering, The University of Texas at Austin. contact: behzadboro@gmail.com ‡Google Inc, Mountain View, CA, USA



(a) Theoretical max velocity. (b) Measured max velocity.

Fig. 1: *(a)* Theoretical relationship between processing time and maximum velocity. *(b)* Relationship between SLAM throughput (FPS) and maximum velocity and energy of UAVs.

II. Compute vs. Flight-time

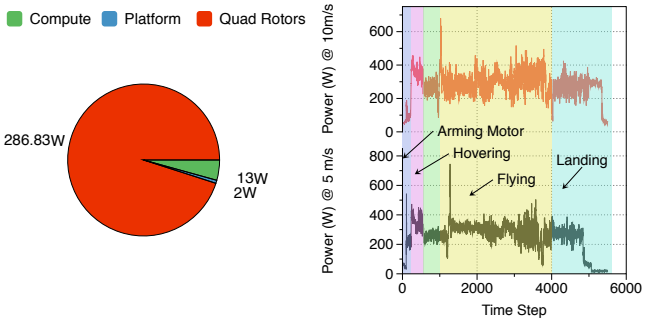
The subtle but critical observation we make is that compute can play an important role in reducing the drone's mission time by increasing the mission's average velocity. Concretely, we identify that the reduction in hover time and the increase in maximum allowed velocity are the two major ways with which more compute can contribute to a higher average velocity.

Hover Time Reduction: Hover time and the average velocity have an inverse relationship, namely, the more drone spends time on hovering, the lower its average velocity. Similar to an idling CPU, a hovering drone is unfavorable since it is not working toward its mission, but yet wasting its limited energy.

Max Velocity Increase, Collision Avoidance Effect: The maximum velocity of the drone is not only mechanically bounded, but also compute bounded since a collision-free flight is only possible if the drone can process its surrounding fast enough to react to it. Equation 1 specifies the components involved in setting this velocity where δ_t, d, a_{max} and v denote process time, required stopping distance, maximum acceleration limit of the drone and maximum velocity [7]. Figure 1a sheds light on this relation.

$$v_{max} = a_{max}\left(\sqrt{\delta t^2 + 2\frac{d}{a_{max}}} - \delta t\right) \quad (1)$$

Max Velocity Increase, Localization Failure Effect: The faster the speed of the drone, the higher the likelihood of its localization failure because the environment changes rapidly around a fast drone. To examine the relationship between the compute, maximum velocity and localization failure, we devised a micro-benchmark in which the drone was tasked to follow a predetermined circular path of the radius 25 meters.



(a) Measured power breakdown. (b) Measured mission power.

Fig. 2: *(a)* Measured power of 3DR Solo. *(b)* Total power while the drone is "Flying" at two different velocities. The power consumption is severely dominated by the quad rotors by 20X.

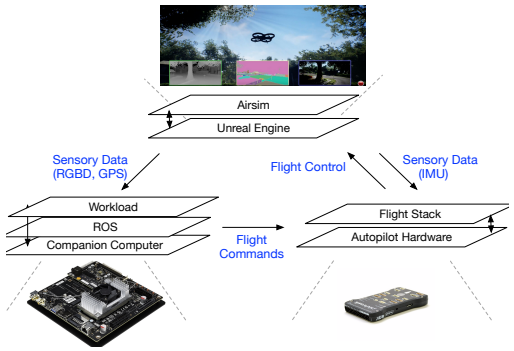


Fig. 3: Overview of our simulation setup, which we use to evaluate a set of simulated applications. These applications include scanning, package delivery, mapping, search and rescue and aerial photography.

For the localization kernel, we used ORB-SLAM2 and to emulate different compute powers, we inserted a sleep in the kernel. We swept different velocities and sleep times and bounded the failure rate to 20 percent. As Figure 1b shows, higher FPS values, i.e. more compute, allows for a higher maximum velocity for a bounded failure rate.

III. Compute vs. Flight-time vs. Energy Efficiency

The compute subsystem can also have a significant role in reducing total MAV energy consumption. To understand this, first we present the power distribution associated with 3DR Solo [8], a popular off-the-shelf MAV. To measure power, we attach a wattmeter known as Eagle Tree Systems eLogger V4 [9] to the 3DR Solo's battery during flight. The wattmeter allows us to collect data over time at 50 Hz while the drone flies. We command the drone to fly for fifty seconds and pull the data off of the wattmeter after the drone lands.

As Figure 2 shows, the majority of the power consumption is dedicated to rotors (locomotion) and the compute only occupies a small portion of the entire pie and its role seemingly trivial. Although small in quantity, compute can have a grand effect on the system's power. This is because by reducing the mission time (as explained in the previous section), more compute power can, in fact, reduce the bigger portion of the pie, namely rotors energy consumption (due to a shorter flight).

We profiled the mission time and the energy associated with the aforementioned microbenchmark. As the bottom plot in Figure 1b shows, higher compute capability results in increased SLAM FPS and hence a reduction in mission time by allowing for faster flights. The reduced mission results in reduced total system energy, as the bottom plot in Figure 1b shows. By increasing processing speed by 5X, we were able to reduce the drone's energy consumption by close to 4X.

IV. Experimental Setup

To conduct detailed experiments, we developed an experimental infrastructure based on existing off-the-shelf hardware and software components. We show how our setup in Figure 3 maps to the components corresponding to a UAV's operation.

Environments, Sensors and Actuators: Environments, sensors and actuators are simulated with the help of a game engine called Unreal [10]. With a physics engine at its heart, it "provides the ability to perform accurate collision detection as well as simulate physical interactions between objects within the world" [11]. Unreal provides a rich set of environments such as mountains, jungles, urban setups, etc. to simulate.

To capture a MAV's dynamics and kinematics through its actuators' behavior and its sensory modules, we used AirSim, an open-source Unreal based plug-in from Microsoft [12]. We limit our sensors and actuators to the ones realistically deployable by MAVs, such as RGB-D cameras and IMUs. Unreal and Airsim run on a powerful computer (host) capable of physical simulation and rendering. Our setup uses an Intel Core i7 CPU and a high-end NVIDIA GTX 1080 Ti GPU.

Flight Controller: AirSim supports various flight controllers that can be either hardware-in-the-loop or completely software-simulated. For our experiments, we chose the default software-simulated flight controller provided by AirSim. However, AirSim also supports other FCs, such as the Pixhawk [13], shown in black in Figure 3 which runs the PX4 [14] software stack. AirSim supports any FC which can communicate using MAVLINK, a widely used micro aerial vehicle message marshaling library [15].

Companion Computer: We used an NVIDIA Jetson TX2 [16], a high-end embedded platform from Nvidia with 256 Pacal CUDA cores GPU and a Quad ARM CPU; however, the flexibility of our setup allows for swapping this embedded board with others such as x86 based Intel Joule [17]. TX2 communicates with Airsim and also FC via Ethernet.

ROS-based Workloads: Our setup uses the popular Robot Operating System (ROS) for various purposes such as such as low-level device control and inter-process communication [18]. All of our workloads run on ROS on top of TX2.

Energy Simulation and Battery Model: In addition to the functional and performance data, energy consumption can also be logged using our setup. To monitor energy, we extended the Airsim simulation environment with an energy and a battery model. Our energy model is a function of the velocity and acceleration of the MAV [19]. The higher the velocity or acceleration, the higher the amount of energy consumption. Velocity and acceleration values are sampled continuously, their associated power calculated and integrated for capturing

the total energy consumed by the agent.

V. Measurement and Evaluation

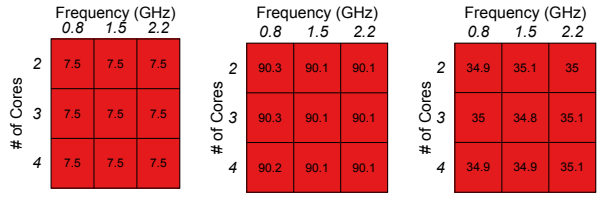
We have assembled 5 representative benchmarks, to examine the effect of compute on MAV systems. Average velocity, mission, and energy values of various operating points are profiled and presented as heat maps (Figure 4—Figure 8). The operating points we consider are changing the number of cores and each core's speed, or clock frequency. This section details the impact of altering the operating points for each application.

Key takeaway: In general, compute can improve mission time and lower energy consumption by as much as 5X.

Scanning: In this simple though popular use case, a MAV scans an area specified by its width and length in a lawn mowing pattern while collecting sensory information about conditions on the ground. We observe trivial differences for velocity, endurance and energy across all three operating points (Figure 4a, Figure 4b, and Figure 4c). This is despite seeing a 3X boost in the motion planning kernel, i.e. lawn mower planning, which is its bottleneck. The trivial effect of compute on this application is because planning is done once at the beginning of the mission and its overhead is amortized over the mission time. E.g., the overhead of planning for a 5 minute flight is less than .001%.

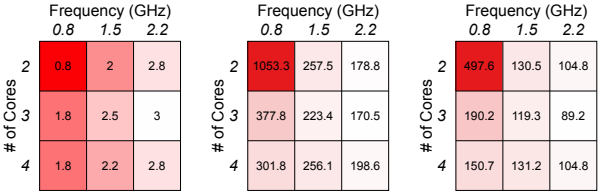
Package Delivery: In this workload, a MAV navigates through an obstacle-filled environment to reach some arbitrary destination, deliver a package and come back to its origin. As compute scales with the number of cores and/or frequency values, we observe a reduction of up to 84% and 82% for the mission time and energy consumption, respectively (Figure 5b, and Figure 5c). The sequential bottlenecks i.e. motion planning and occupancy map generation kernel are sped up by frequency scaling to enable the observed improvements. There does not seem to be a clear trend with core scaling, concretely between 3 and 4 cores. We conducted investigation and determine that such anomalies are caused by the non-real-time aspects of ROS, AirSim and the TCP/IP protocol used for the communication between the companion computer and the host. We achieve up to 2.9X improvement in occupancy map generation and that leads to maximum velocity improvement. It is important to note that although we also gain up to 9.2X improvements for the motion planning kernel, the low number of re-plannings and its short computation time relative to the entire mission time render its impact trivial. Overall the aforementioned improvements translate to up to 4.8X improvement in the average velocity. Therefore, mission time and the MAV's total energy consumption are reduced.

Mapping: With use cases in mining, architecture, and other industries, this workload instructs a MAV to build a 3D map of an unknown polygonal environment specified by its boundaries. We observe a reduction of up to 273% and 64% for the mission time and energy consumption, respectively, as compute scales with the number of cores and/or frequency values (Figure 6a, Figure 6b, and Figure 6c). The amount of concurrency present in this application justifies core scaling performance boost. The sequential bottlenecks, i.e. motion planning and occupancy map generation kernel explain the frequency scaling improvements. We achieve up to 6.3X improvement in motion planning and that leads to hover time reduction. We achieve 6X improvement in occupancy map generation and that leads to maximum velocity improvement. Overall, such improvements translate to an overall 3.2X improvement in average velocity. Therefore, mission time is reduced and consequently so is the MAV's total energy consumption.



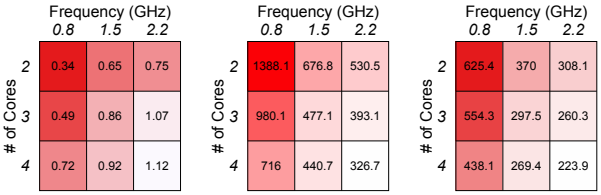
(a) Velocity (m/s) (b) Mission Time (s) (c) Energy (kJ)

Fig. 4: Scanning.



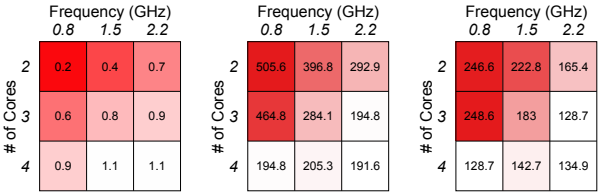
(a) Velocity (m/s) (b) Mission Time (s) (c) Energy (kJ)

Fig. 5: Package Delivery.



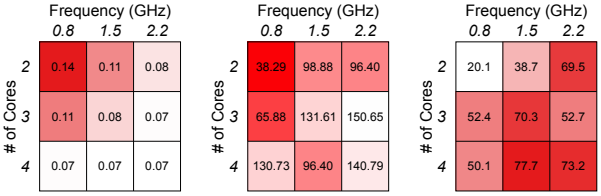
(a) Velocity (m/s) (b) Mission Time (s) (c) Energy (kJ)

Fig. 6: Mapping.



(a) Velocity (m/s) (b) Mission Time (s) (c) Energy (kJ)

Fig. 7: Search and Rescue.



(a) Error Rate(m/s) (b) Mission Time(s) (c) Energy (kJ)

Fig. 8: Aerial Photography.

Search and Rescue: MAVs are promising vehicles for

search-and-rescue scenarios where victims must be found in the aftermath of a natural disaster. We see a reduction of up to 60% and 50% for the mission time and the energy, respectively, as compute scales (Figure 7a, Figure 7b, and Figure 7c). Similar to mapping, more compute allows for the reduction of hover time and an increase in maximum velocity which contribute to the overall reduction in mission time and energy. In addition, a faster object detection kernel prevents the drone from missing sampled frames during any motion. We achieve up to 1.8X, 6.8X, and 6.6X speedup for the object detection, motion planning and occupancy map generation kernels, respectively. In aggregate, these improvements translate to 4.9X improvement in the MAV's average velocity.

Aerial Photography: In this workload, we design the MAV to follow a moving target with the help of computer vision algorithms. We observe an improvement of up to 53% and 267% for *error* and mission time, respectively (Figure 8a, Figure 8b, and Figure 8c). In aerial photography, as compared to other applications, higher mission time is more desirable than a lower mission time. This is because the drone only flies while it can track the person, hence a longer mission time means longer tracking time. In addition to maximizing the mission time, error minimization is also desirable for this application. We define error as the distance between the person's bounding box (provided by the detection kernel) center to the image frame center. Clock and frequency improvements translate to 2.49X and 10X speedup for the detection and tracking kernels and that allows for longer tracking with a lower error. No significant trend is observed in the energy data because energy depends on both the mission time and the velocity, and as opposed to the other applications, there is no need for the drone to minimize its velocity. Instead, it needs to successfully track the person.

VI. The Road Behind and the Road Ahead

Traditionally, the UAV community has almost always, if not exclusively, focused on improving algorithmic efficiency for the sole purpose of getting the task at hand to be computed correctly. However, as these systems become more integrated (i.e., everything moves onboard) and autonomous, there is a need to pay attention to their overall efficiency; The definition of a system here includes the algorithms, software, and underlying digital (i.e. compute platform elements such as processors/co-processors) and mechanical hardware. It is this total system's efficiency that ultimately matters for completing a mission.

We emphasize the need to take a holistic system's driven approach to UAVs since *all* UAVs operate on a tightly constrained power budget, which severely limits their mission time. Even if UAV systems were to work in a collective as a "swarm," improving the efficiency of one UAV at a time by optimizing its compute will result in an overall net effect where the endurance of the swarm will extend.

So, we foresee a future where it will become necessary to understand how we can optimize the algorithms while also improving the total power consumption and energy efficiency of a UAV from algorithms down to the hardware design.

VII. Conclusion

Despite their ample use cases and a rapid progress in the robotic community, UAVs have seen a modest examinations from the software and hardware architects. In this paper, we examined importance of compute for such systems by show casing two compute knobs. We uncover a hidden compute to total system energy relationship where faster computers can allow drones to finish missions quickly, and hence save energy. This is because most of the drone's energy is consumed by the rotors, hence, faster compute can cut down on mission time (by increasing the max velocity and reducing the hover time) and energy accordingly. This insight is examined in 5 applications where we see up to 5X improved due to the aforementioned compute knobs. Going forward, we humbly call for the collaboration between the robotic and computer architecture community to make autonomous drones a reality.

References

[1] Debra R. Cohen McCullough, ""Unmanned Aircraft Systems(UAS) Guidebook in Development"."

[2] A. Qiantori, A. B. Sutiono, H. Hariyanto, H. Suwa, and T. Ohta, "An emergency medical communications system by low altitude platform at the early stages of a natural disaster in indonesia," *J. Med. Syst.*, vol. 36, pp. 41–52, Feb. 2012.

[3] James Rogers, "How drones are helping the Nepal earthquake relief effort." http://www.foxnews.com/tech/2015/04/30/how-drones-are-helping-nepal-earthquake-relief-effort.html, note =.

[4] "Amazon delivered its first customer package by drone." https://www.usatoday.com/story/tech/news/2016/12/14/amazon-delivered-its-first-customer-package-drone/95401366/. (Accessed on 01/22/2018).

[5] Arthur Holland Michel, "Amazon's Drone Patents." http://dronecenter.bard.edu/amazon-drone-patents/, 2017. [Online; accessed 09-Jan-2018].

[6] BBC News, ""Google Plans Drone Delivery Service for 2017"."

[7] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1484–1491, IEEE, 2016.

[8] "3dr robotics." https://3dr.com/solo-drone/. (Accessed on 04/5/2018).

[9] "Eagle tree systems." http://www.eagletreesystems.com/index.php?route=product/product&product_id=54. (Accessed on 04/5/2018).

[10] "Game engine technology by unreal." https://www.unrealengine.com/en-US/what-is-unreal-engine-4. (Accessed on 01/19/2018).

[11] "Physics simulation — unreal engine." https://docs.unrealengine.com/latest/INT/Engine/Physics/. (Accessed on 01/19/2018).

[12] "Microsoft/airsim: Open source simulator based on unreal engine for autonomous vehicles from microsoft ai & research." https://github.com/Microsoft/AirSim. (Accessed on 01/19/2018).

[13] "Home - pixhawk flight controller hardware project." https://pixhawk.org/. (Accessed on 01/19/2018).

[14] "Px4 architectural overview Â· px4 developer guide." https://dev.px4.io/en/concept/architecture.html. (Accessed on 01/19/2018).

[15] "mavlink/mavlink: Marshalling / communication library for drones.." https://github.com/mavlink/mavlink. (Accessed on 01/19/2018).

[16] "Embedded systems developer kits, modules, & sdks — nvidia jetson." https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/. (Accessed on 01/19/2018).

[17] "Iot." https://software.intel.com/en-us/intel-joule-getting-started. (Accessed on 01/19/2018).

[18] "Ros.org — powering the world's robots." http://www.ros.org/. (Accessed on 01/19/2018).

[19] C. D. Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pp. 111–117, April 2015.