The Sky Is Not the Limit: A Visual Performance Model for Cyber-Physical Co-Design in Autonomous Machines

Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi

Abstract—We introduce the "Formula-1" (F-1) roofline model to understand the role of computing in aerial autonomous machines. The model provides insights by exploiting the fundamental relationships between various components in an aerial robot, such as sensor framerate, compute performance, and body dynamics (physics). F-1 serves as a tool that can aid computer and cyber-physical system architects to understand the optimal design (or selection) of various components in the development of autonomous machines.

Index Terms—Cyber-physical system, system architecture, performance model, aerial autonomous machines, robot learning

1 Introduction

HIGH-SPEED autonomous aerial robot navigation [1], [2], [3], [4] in cluttered environments is a highly sought after feature as it can enable many potential applications. Examples include autonomous drone racing, search and rescue operations in damaged buildings, forests, etc. However, high-speed aerial robot navigation is challenging as the control algorithms, that generate actions for the robot, must run onboard with limited computing, sensing capability, and battery.

There is a need to understand the role of computing on the performance of aerial robots. To enable onboard computation of a control algorithm, aerial roboticists often choose either general-purpose Intel NUC [5] platforms or embedded GPU platforms such as Nvidia Jetson TX2 [6] or Nvidia Xavier AGX [7]. Similarly, there are several choices for learning-based control algorithms, such as the use of standard neural networks (e.g., VGG-16 [8]) or customized ones (e.g., DroNet [9]). However, it is unclear which combination of the control algorithm and computing platform will result in optimal performance (e.g., velocity) for a robot while also considering the type of sensor and its other physical aspects such as thrust-to-weight ratio (i.e., body dynamics). Therefore, there is a fundamental need for a systematic and comprehensive model that both the computer architects and roboticists can use to achieve an optimal cyber-physical co-design of the various components of an aerial robot.

Computer architects have long benefited from performance models to guide the design of optimal systems for a given workload. For instance, the roofline model [10] proposed a simple yet powerful model to visualize the performance for various workloads on a given multicore CPU system. It provides a relation between peak compute performance (ops/s), peak memory bandwidth, while modeling the application using operational intensity (Flops/Byte). Recently, the Gables [11] model extended the roofline to SoCs consisting of heterogeneous IPs. Both models generate a visual plot that defines the upper bound on maximum attainable performance for multicore systems. Performance models by design are not perfect, rather they are intended to provide the first-level of insights to understand the various effects and bottlenecks in a system before its actual design.

Inspired by such models, in this paper, we answer the following question: *What is a useful performance model for cyber-physical system design, specifically in the context of autonomous machines?* To this end, we propose 'Formula-1 (F-1): a roofline model for the cyber-physical co-design of autonomous aerial machines. *The goal of the F-1 roofline is to provide a visual model to understand the impact of the onboard computing platform's performance on the velocity of aerial autonomous machines, while taking into consideration the effects of sensor and physical parameters (e.g., thrust-to-weight ratio).* F-1 establishes different regions: a region where the velocity of the robot is only affected by the computing system's performance, and others where it is only bounded by the type of sensor or the body dynamics. F-1 can be used to co-design the entire cyber-physical space for the aerial robots: in the compute-bounded region, the computer architects should optimally design/select the control algorithm and computing hardware while also considering the various physical aspects, and when the velocity is sensor/body dynamics bound, the roboticists can optimize the sensor/mechanical components, while keeping in mind the available computing options.

To show the utility of the F-1 model, we evaluate the impact of commonly-used learning-based control algorithms, running on real-world computing platforms, on a given aerial robot's velocity. We use three algorithms: VGG-16 [8], DroNet [9], and TrailNet [12]. Four common drone compute platforms are considered: Nvidia Xavier, Nvidia TX2, Intel NCS, and Ras-Pi. With these algorithms and platforms in conjunction with the F-1 model, we show how running a control algorithm on a high-performance system does not necessarily achieve high velocity due to the following relationship: *high-performance compute ⇒ high power ⇒ higher TDP ⇒ more-weight (due to more complex hardware and heatsink) ⇒ reduced-acceleration ⇒ lower-velocity*. Conversely, we also show that a low-power computing platform also does not necessarily achieve high velocity due to the following relationship: *low-power compute ⇒ lower compute performance ⇒ lower algorithm performance⇒ more compute-bounded execution ⇒ lower/slower flight speed or velocity*.

Our observations suggest that attaining high-velocity on an aerial-robot requires computer architects' attention to design computing platforms that are capable of achieving high performance-per-watt, not due to traditional thermal constraints, but rather due to weight constraints. The algorithms running on a platform need to achieve high-performance under a low power envelope to avoid the need for heavy cooling solutions. F-1 serves as a useful tool in systematically navigating the design of computer systems for aerial robots.

2 Cyber-Physical Model of Autonomous Aerial Robots

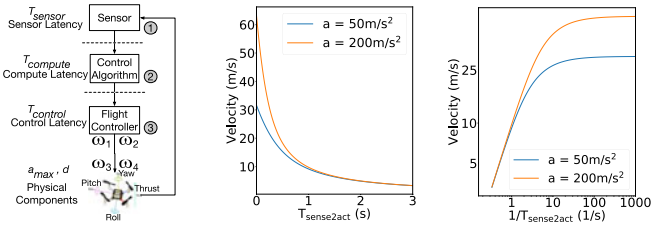
Aerial robots are cyber-physical systems that involve an interplay of the sensor, compute platform, and other mechanical units such as rotors. The impact of these components on velocity is discussed next.

2.1 Sense-to-Act Components

A typical interaction of different components in an aerial autonomous machine is shown in Fig. 1a. ① The first component is the sensor to capture the state of the environment. An example of a

- *S. Krishnan, Z. Wan, K. Bhardwaj, G. Y. Wei, D. Brooks, and V. J. Reddi are with Harvard University, Cambridge, MA 02138. E-mail: srivatsan@seas.harvard.edu, {zishenwan, kbhardwaj, gywei}@g.harvard.edu, {dbrooks, vj}@eecs.harvard.edu.*
- *P. Whatmough is with Harvard University, Cambridge, MA 02138, and also with ARM Research, Waltham, MA 02451. E-mail: paul.whatmough@arm.com.*
- *A. Faust is with Robotics, Google, Mountain View, CA 94043. E-mail: sandrafaust@google.com.*

Manuscript received 15 Feb. 2020; accepted 6 Mar. 2020. Date of publication 16 Mar. 2020; date of current version 29 Apr. 2020.
(Corresponding author: Srivatsan Krishnan.)
Digital Object Identifier no. 10.1109/LCA.2020.2981022



(a) sense-to-act Flow. (b) CPS Model. (c) Roofline.

Fig. 1. The sensor-compute pipeline, CPS relationship, and the roofline model in log-log scale for aerial autonomous machines.

sensor includes a monocular camera which is usually mounted on the aerial robot. ② The second component is the onboard compute system, which is the compute hardware executing the control algorithms. The compute system generates action command depending upon the sensor input. ③ The third component is the flight controller. which takes the high-level command generated by the compute system to generate low-level actuation signals (e.g., angular velocities). The latencies of these components are described next.

Sensor Latency (T_{sensor}). It is the time to fetch data from the sensor is known as sensor latency. For example, if the aerial robot has 30 FPS camera, it means that we can sample one image at 33.3 ms interval, which becomes the sensor latency (T_{sensor}).

Compute Latency ($T_{compute}$). It is the time to process the control algorithm to estimate the high-level action commands. The algorithm running on the computing system feeds on the sensor data. Compute latency is a function of the algorithm as well as the underlying system architecture of the onboard computing platform.

Control Latency ($T_{control}$). It is the time to generate the low-level actuation commands. The flight controller operates upwards of 1 kHz [13], and hence the $T_{control}$, which is inverse of controller frequency, is negligible compared to the sensor/compute latency.

Total Sense to Act Latency ($T_{sense2act}$). Since $T_{control}$ is negligible, the total sense to act latency in this pipelined system is

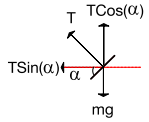
$$T_{sense2act} \approx max(T_{sensor}, T_{compute}). \tag{1}$$

Based on Equation (1), one can observe that the choice of sensor framerate and compute platform plays a role on $T_{sense2act}$.

2.2 Physical Components

The physical components play an important role in the performance of the aerial robots. Physical components in the aerial robot are the intrinsic and mechanical properties of either the sensor, compute, and body (frame + rotors). Examples of mechanical properties include the mass of the sensor, compute, rotors, and body frame, thrust-to-weight ratio, the maximum range of the sensor, etc. We need to understand these to capture the cyber-physical model and its relationships.

We abstract the effect of various physical component in the aerial autonomous machine using two terms, namely, acceleration of the aerial robot (a_{max}) and sensor range (d). Acceleration (a_{max}) models all the effects that arise by adding extra weight to the aerial robot. For instance, adding a powerful compute platform or sensor will add extra weight to an aerial autonomous machine. This added weight will, in turn, reduce the thrust-to-weight ratio, which reduces the acceleration (a_{max}) of the drone. Likewise, we can abstract the sensing quality using sensor range (d). A powerful time-of-flight sensor can provide a higher sensing range, whereas a camera array based depth sensor can provide only a limited range. Hence, the sensing range parameter can be used to abstract the type and quality of the sensor used.



$$T\cos(\alpha) - mg = ma_{max}$$
$$a_{max} = \frac{T\cos(\alpha) - mg}{m} \tag{2}$$

Fig. 2. Free body diagram and estimation of acceleration.

In particular, Fig. 2 uses the free body diagram to show the effect of a robot's weight on its maximum acceleration (a_{max}).[1,2] The four rotors in the quadroptor produce thrust (T) perpendicular to the plane of the aerial robot. The value of the thrust is usually referred to as "Pull" in the motor specification. For the aerial robot to move forward, it has to pitch at an angle α. The vertical component of the thrust ($T\cos(\alpha)$) needs to overcome the weight (mg) component to fly upwards. Using the Newtons' second law, the resultant net force equals the product of mass times the acceleration. Re-arranging the terms leads us to Eq. (2), using which we can model the acceleration to account for the mass of the compute, sensor, or any other payload. Likewise, we can model the other effects such as air drag in the free-body diagram (not shown currently).

2.3 Cyber-Physical Relationship

The velocity of an autonomous aerial robot depends upon the sense-to-act and physical components. Prior work [5] established and validated this relationship as defined by the equation below

$$v = a_{max}\left(\sqrt{T_{sense2act}^2 + 2\frac{d}{a_{max}}} - T_{sense2act}\right), \tag{3}$$

a_{max} is the maximum attainable acceleration that the body dynamics of an aerial robot can allow, d is the sensing range, and $T_{sense2act}$ is the time from sensor to action as defined in Equation (1).

The Eq. (3) makes a few assumptions about the Cyber-Physical relationship. First, it uses short-range planners where the planning is done only for a limited horizon instead of extended horizon. Second, the model limits the maximum velocity with the assumed physical parameters (a_{max} and d) to ensure it can safely stop without colliding with an obstacle. Third, the planning algorithm can generate a stopping policy at each step.

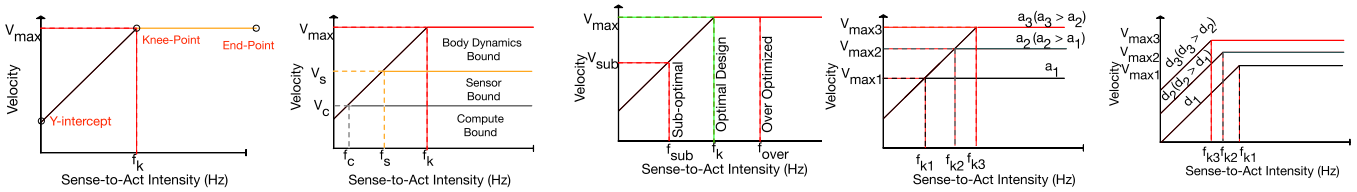
Using Eq. (3), we can understand the role of compute in the robot at a fundamental level. As it turns out, the relationship closely resembles a traditional computer system roofline model [10] as seen in Fig. 1c.

To get there, we sweep the $T_{sense2act}$ from $0 \rightarrow 3$ seconds along with two values of accelerations (a_{max} = 50 m/s^2 and a_{max} = 100 m/s^2) and the sensor range (d = 10m). The resulting plot is shown in Fig. 1b. We can observe that the Equation (3) has asymptotic relation between velocity and $T_{sense2act}$ such that as $T_{sense2act} \rightarrow 0$, the velocity $\rightarrow \infty$. Likewise, as the $T_{sense2act} \rightarrow \infty$, the velocity $\rightarrow 0$.

We also plot the inverse of $T_{sense2act}$ on the x-axis and velocity on the y-axis in Fig. 1c. Both the x-axis and y-axis are plotted on a log-log scale. We see that there exists a point beyond which decreasing $T_{sense2act}$ does not increase the velocity, showing a saturation or a *roofline*. Decreasing the sense-to-act latency (e.g., faster computing platform, faster sensor etc.) beyond a certain point will yield no improvement in the velocity of the aerial robot.

The resulting output seen in Fig. 1c is that similar to the traditional roofline model, where increasing operational intensity beyond a certain point does not improve the overall throughput of the system, over-optimizing any one component can result in an unbalanced system.

1. https://catsr.vse.gmu.edu/SYST460/QuadcopterDynamics.pdf
2. Diagram is an illustration to visualize the net forces acting on a body.



(a) Roofline construction. (b) Different bounds. (c) Optimal Design. (d) Effect of a_{max}. (e) Effect of d.

Fig. 3. The proposed F-1 roofline model for aerial autonomous machines.

3 F-1 Roofline Model

The F-1 model provides a simple visual model to help (computer) system architects determine the role of the computing platform in the overall cyber-physical design of aerial robots. We discuss how to visualize the F-1 model to determine the impact of different robot components (compute performance, sensor framerate, and body dynamics) on its velocity. We also show how varying the physical properties, such as its acceleration, can impact the choice of onboard computing requirements.

3.1 Visualizing the F-1 Roofline Model

The F-1 roofline model in Fig. 3a is constructed by approximating the original cyber-physical relationship between a robot's velocity and its $T_{sense2act}$ (Fig. 1c). The F-1 model is a log-scale plot between velocity and "*Sense-to-Act Intensity (Hz),*" where the latter is the throughput of the sense-to-act pipeline, defined as

$$Sense - to - Act\ Intensity = 1/(T_{sense2act}). \quad (4)$$

To visualize the F-1 model, we need to show two regions (borrowed from Fig. 1c): (i) where a robot's velocity depends on $T_{sense2act}$, and (ii) where the velocity is saturated and independent of $T_{sense2act}$. The former is approximated by a slope while the latter is a horizontal line. The slope begins at the "*Y-intercept,*" which is the velocity when the sense-to-act intensity is ≈0, and ends at the "*Knee-Point.*" The knee is the intersection of the slope and the horizontal saturation line. It is determined using the data from Fig. 1c and by searching for the sense-to-act intensity after which point the velocity becomes constant. The saturation continues until the "*End-Point,*" whose x-coordinate is greater than the knee-point.

The resulting Fig. 3 is similar to the original roofline model, but it has a different meaning. The peak horizontal line in the F-1 model denotes the maximum velocity, whereas the horizontal line in the roofline model [10] corresponds to the peak floating-point performance. Likewise, the slope in the F-1 model is a function of sensor range (d), whereas the slope in the original roofline model [10] signifies the peak memory bandwidth of a multicore system.

3.2 Different Bounds and Sensor/Compute Optimality

Using the F-1 model, we show that the maximum attainable velocity by an aerial autonomous robot can be (theoretically) upper bounded by either the body dynamics (physics), sensor framerate or the compute performance. These bounds can help us choose or design optimal computing platforms that can maximize the robot's velocity.

Body Dynamics Bound (The Ultimate Upper Limit). For a given aerial robot with its set of physical properties, such as body frame, rotors, and thrust to weight ratio, the ultimate bound on velocity (v_{max}) is determined by its body dynamics (Fig. 3b). We call the region to the right of the knee-point (i.e., when sense-to-act intensity is greater than or equal to f_k) as *body dynamics bound*. In this region, unless the physical components are improved (e.g., increasing thrust-to-weight ratio), the velocity cannot exceed the current peak velocity.

Sensor Bound. As in Fig. 3b, a robot's velocity can be said to be sensor-bound if its sense-to-act intensity is equal to the sensor's frame rate (f_s or $1/T_{sensor}$) but less than the knee-point intensity (f_k). In more detail, the sensor-bound case is when the latency of the compute ($T_{compute}$) is less than or equal to the sensor latency (T_{sensor}) (i.e., sense-to-act intensity is equal to f_s according to Equation 1), and $f_s < f_k$. In this scenario, the sensor adds a new ceiling to the roofline model, bounding the velocity under V_s.

Compute Bound. As also shown in Fig. 3b, a robot's velocity is compute-bound if its sense-to-act intensity (f_c) is less than the sensor's frame rate (f_s) and the knee-point intensity (f_k). That is, when $T_{compute} > T_{sense}$ and $f_s \leq f_k$. In this case, the computing platform adds a new ceiling to the roofline model, bounding the velocity under this limit (V_c).

Optimal Computing Platform and Sensor Design. Fig. 3c shows how understanding the bounds on velocity through the F-1 roofline model can help us design an optimal cyber-physical system for aerial autonomous machines. For a given aerial robot with fixed mechanical properties, changing the sensor type or onboard computing impacts the $T_{sense2act}$ (or sense-to-act intensity). Consequently, the optimal design point is when the sensor latency and compute latency result in a sense-to-act intensity that is equal to the knee-point intensity (f_k). If the sense-to-act intensity is f_{over} such that $f_{over} > f_k$, then either the sensor/computer is over-optimized since any value greater than f_k yields no improvement in the velocity of the aerial robot. Likewise, if the sense-to-act intensity is f_{sub}, such that $f_{sub} < f_k$, then the sensor/computer is under-optimized, which signifies that there is scope for improvement through a better algorithm design or via better selection (or design) of the compute.

3.3 Effect of Increasing Acceleration and Sensor Range

The F-1 roofline model can be used to visualize the effect of varying maximum acceleration (a_{max}) and sensor range (d). As shown in Figs. 3d and 3e, a higher acceleration (a_{max}) or increased sensor range (d) signifies an increase in the attainable velocity for the aerial robot and pushes the roofline upwards. When different a_{max} are considered in an increasing order, the knee point shifts towards the right, this trend shows that for an aerial robot with high a_{max}, achieving maximum velocity will require a higher sense-to-act intensity (i.e., using high-performance compute/sensor). When sensor range increases, the knee-point shifts towards the left, suggesting for a high sensor range (d) a nominal computing performance can achieve the sense-to-act intensity required to achieve the maximum velocity.

4 Architectural Insights From The F-1 Model

In this section, we demonstrate the effectiveness of our F-1 roofline model. In particular, using the F-1 model, we determine the impact of the different real-world computing platforms, executing the commonly-used learning-based control algorithms, on the velocity of an autonomous aerial robot. Our analysis shows that the velocity in several cases is limited by the onboard compute systems, even though the robot's mechanical components can allow for it to move faster.

Experimental Setup. Without any loss in generality, we consider an aerial robot that has a thrust-to-weight ratio of 6 [14]. The aerial

TABLE 1
Targeted Computing Platforms and Learning Based Control Algorithms

Platform	TDP (W)	Weight (g)	Heatsink Weight (g)	Aerial Robot Base Weight (g)	Maximum Acceleration (m/s^2)	Control Algorithm
No-Acc	<500mW	-	-	535	~60	Human teleoperated
Xavier	<30W	280 [17]	162	535	~23	Vgg16, DroNet, TrailNet
TX2	<15W	85 [18]	81	535	~35	Vgg16, DroNet, TrailNet
Ras-Pi	<1.5W	18 [19]	8.1	535	~46	DroNet, TrailNet
Intel NCS	<1W	42 [20]	5.4	535	~43	Vgg16, DroNet, TrailNet

robot is equipped with a camera sensor, and weighs 585 g, including the weight of the sensor, body frame, and battery. The baseline robot is human teleoperated; it comes with a micro-controller unit, but has limited computing and memory capacity to run autonomous control algorithms other than the flight controller stack. Since this onboard compute system does not use a hardware accelerator, we refer to this baseline configuration as "*No-Acc*." Such an aerial robot can achieve a max acceleration of 60 m/s^2, based on its thrust-to-weight ratio.

We augment the baseline robot configuration with four different accelerators that have varying compute capabilities: Nvidia Xavier, Nvidia TX2, Intel NCS, and Ras-Pi 3b. These systems are selected as they are used in real aerial robots [6], [7], [15], [16]. Therefore, in addition to the "No-Acc" baseline, we create four other robot configurations: each using a different accelerator, while the rest of the mechanical parameters (e.g., sensor) remain the same as the "No-Acc" baseline. A diverse set of realistic control algorithms are run on these four configurations: Vgg16 [8], DroNet [9], and TrailNet [12].

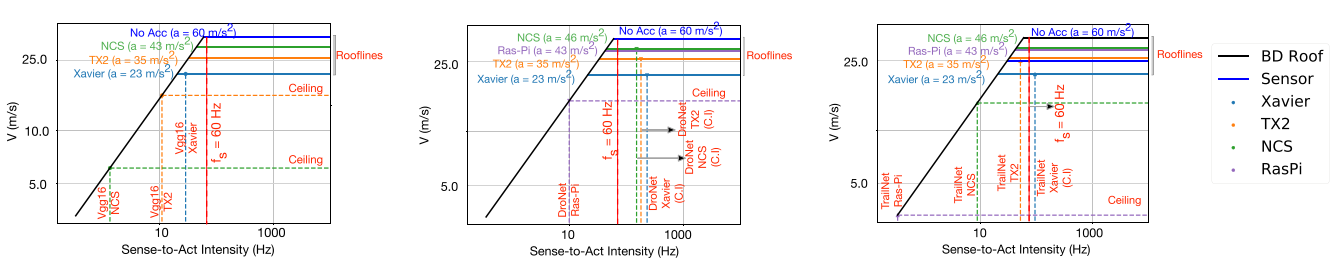
Cyber-Physical System Co-Design. Table 1 shows the maximum acceleration for each of the four robot configurations when using the different accelerator-based computing platforms. For the maximum acceleration calculation, we use Eq. (2), where weight (mg) includes the robot's base weight, and the computing platform's and its corresponding heatsink's weight. To estimate the heatsink weight, we use the Thermal Design Power (TDP) of a platform to determine the volume of the heatsink [21] required for cooling and multiply the volume with the density of aluminum (commonly used material for heatsink). Since Xavier is the heaviest of the four, it achieves the minimum acceleration, while Rasp-Pi/Intel NCS achieve the highest. However, these peak acceleration values are still lower than the "No-Acc" baseline acceleration of 60 m/s^2, thus implying it is important to consider the effect of compute weight on a robot's max acceleration.

Furthermore, we show how the choice of the (1) onboard computing platform and (2) control algorithm affect the maximum velocity of the robot. We investigate running a computationally-intensive (Vgg16) and computationally-light control algorithms (DroNet and TrailNet) on high-performance (Nvidia Xavier and Nvidia TX2) and low-power (Intel NCS and Ras-Pi) computing platforms. Fig. 4 shows that a high-performance computing platform (e.g., Xavier) has the lowest roofline due to the added weight. These platforms trade-off power for performance, which translates to higher TDP (e.g., 30W), thus requiring a bigger (and heavier) heatsink. We also show that a low-power platform (e.g., Intel NCS), that trade-offs performance for low power, achieves a higher roofline since it is lighter.

Computationally-Intensive Control Algorithm. To determine the velocity ceilings when using the four computing platforms, the sense-to-act intensity for these systems is computed using Eq. (4), where compute latency is determined by running VGG-16 on the system, and sensor latency is assumed to be 16.6 ms (f_s = 60 Hz). Fig. 4a shows ceilings obtained by three platforms (Ras-Pi 3b runs out of memory for VGG-16). The sense-to-act intensities of Xavier, NCS, and TX2 are dominated by their compute latencies as they are higher than the sensor latency. Xavier achieves higher sense-to-act intensity of 28 Hz compared to TX2 (10 Hz) and NCS (1.3 Hz). For Xavier, the velocity is not bounded by compute as its sense-to-act intensity is higher than its roofline's knee point. But for TX2 and NCS, the velocity is compute-bound that adds ceilings to their rooflines. Therefore, Xavier results in a higher maximum velocity, even though its a_{max} is lowest.

Computationally-Light Control Algorithm. Similarly, Fig. 4b and 4c show the roofline models when running DroNet and TrailNet on the four computing platforms. These two control algorithms are less computationally-intensive than Vgg16. For DroNet, the velocity of the drone using Ras-Pi 3b is bounded by the computing platform (as shown by its ceiling), while the other three platforms achieve a higher maximum velocity, which is not bounded by the compute. Interestingly, NCS (running DroNet) can enable higher velocity than Xavier, even though the latter is faster than the former, as the velocity for NCS is not compute-bounded and its lower weight allows for higher acceleration of the drone. Similarly, for TrailNet, the slowest is Ras-Pi which has the lowest ceiling followed by NCS, while for TX2 and Xavier, velocity is not bounded by the computing platform. A relatively lightweight TX2 achieves higher velocity than Xavier.

Interestingly, a computationally-light control algorithm, such as DroNet, running on low-power NCS is able to achieve a higher maximum velocity than VGG-16 running on high-performance Xavier.



(a) VGG-16 (b) DroNet (c) TrailNet

Fig. 4. The F-1 roofline plots for three end-to-end learning models running on Nvidia Xavier/TX2, Intel NCS, and Ras-Pi. *C.I* is the throughput of running a control algorithm on a given hardware, shown only when it is greater than f_s (sense-to-act intensity is equal to f_s in these cases).

Based on these observations, a computing platform that is optimal in both performance and power will be ideal for applications targeting autonomous aerial robotics. While high performance ensures that velocity is not compute-bound, low power dissipation translates in lower weight (smaller heatsink), hence able to support higher a_{max} (higher roofline). Given that the sense-to-act intensities of all the different combinations of control algorithms and computing platforms in Fig. 4 are far from their optimal knee points in their respective roofline plots, there is a need for algorithm-hardware co-design to achieve optimal compute systems for aerial robots.

The F-1 roofline plots can also be used by cyber-physical system architects (or roboticists) to understand the design (or selection) of the sensor and other mechanical properties starting from a fixed computing platform. For instance, from Fig. 4a, a sensor framerate of 60 FPS is over-designed for VGG-16 running on Xavier. Instead, a 30 FPS sensor framerate can achieve nearly the same velocity as the 60 FPS camera since the sense-to-act intensity for VGG-16 running on Xavier is around 35 Hz (f_c = 35 Hz). These insights can allow for a better design of the overall cyber-physical system for aerial robots.

5 Conclusion

F-1 is a model to understand the role of computing platforms in aerial autonomous robots. The model plots the relationship between the onboard compute system's performance and the robot's velocity, while considering the sensor and body dynamics properties. Future work can readily extend the F-1 model to autonomous cars.

Acknowledgments

The authors would like to thank Mark Hill, Sophia Shao, Divya Mahajan, and the reviewers for their valuable feedback that improved this paper. This work was supported in part by the U.S. Government, under the DARPA DSSoC program. The work was also supported in part by the SRC, NSF IIS award #1724197, and Intel.

References

[1] Darpa fla. Accessed: Mar. 28, 2020. [Online]. Available: https://www.darpa.mil/program/fast-lightweight-autonomy
[2] K. Mohta *et al.*, "Experiments in fast, autonomous, GPS-denied quadrotor flight," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 7832–7839.
[3] G. Loianno, D. Scaramuzza, and V. Kumar, "Special issue on high-speed vision-based autonomous navigation of UAVs," *J. Field Robot.*, vol. 1, no. 1, pp. 1–3, 2018.
[4] S. Li, M. M. O. I. Ozo, C. De Wagter, and G. C. H. E. de Croon, "Autonomous drone race: A computationally efficient vision-based navigation and control strategy," *CoRR*, vol. abs/1809.05958, 2018. [Online]. Available: http://arxiv.org/abs/1809.05958
[5] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1484–1491.
[6] News center. Oct. 2019. [Online]. Available: https://news.developer.nvidia.com/skydio-2-jetson-tx2-drone/
[7] Airr. Accessed: Mar. 28, 2020. [Online]. Available: https://thedroneracingleague.com/airr/
[8] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Proc. Robot.: Sci. Syst.*, 2017.
[9] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.
[10] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.
[11] M. Hill and V. J. Reddi, "Gables: A roofline model for mobile socs," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2019, pp. 317–330.
[12] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 4241–4247.
[13] W. Koch, R. Mancuso, and A. Bestavros, "Neuroflight: Next generation flight control firmware," *CoRR*, vol. abs/1901.06553, 2019. [Online]. Available: http://arxiv.org/abs/1901.06553
[14] Rtf eachine wizard (68mph)-complete parts list—FPV drone reviews. Accessed: Mar. 28, 2020. [Online]. Available: https://fpvdronereviews.com/reviews/rtf-eachine-wizard-68mph/
[15] L. Upton and V. Gupta, "Autonomous drones (only slightly flammable)," Sep. 2018. [Online]. Available: https://www.raspberrypi.org/blog/autonomous-drones-only-slightly-flammable/
[16] News & announcements. May 2017. [Online]. Available: https://www.movidius.com/news/intel-movidius-myriad-2-vpu-enables-advanced-computer-vision-and-deep-learn
[17] Nvidia xavier agx. Accessed: Mar. 28, 2020. [Online]. Available: https://devtalk.nvidia.com/default/topic/1048372/jetson-agx-xavier/-wei gth-of-jetson-agx-xavier-module-only/
[18] Nvidia jetson TX2 delivers twice the intelligence to the edge—nvidia developer blog. Accessed: Mar. 28, 2020. [Online]. Available: https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge /
[19] Raspberry pi. [Online]. Available: https://www.pololu.com/blog/598/new-product-raspberry-pi-3-model-b
[20] Amazon.com: Intel neural compute stick 2: Computers & accessories. Accessed: Mar. 28, 2020. [Online]. Available: https://www.amazon.com/Intel-Neural-Compute-Stick-2/dp/B07KT6361R
[21] Heat sink size calculator. Accessed: Mar. 28, 2020. [Online]. Available: https://celsiainc.com/resources/calculators/heat-sink-size-calculator/