2017 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 13-16, 2017, Miami, FL, USA

Toward a Cyber-Physical Quadrotor: Characterizing Trajectory Following Performance

Ajay Shankar, Seth Doebbeling, Justin Bradley
University of Nebraska, Lincoln, NE 68588

***Abstract*— An Unmanned Aircraft System (UAS) is a Cyber-Physical System (CPS) in which a host of real-time computational tasks contending for shared resources must be cooperatively managed to provide actuation input for control of the locomotion necessary to obtain mission objectives. Traditionally, control of the UAS is designed assuming a fixed, high sampling rate in order to maintain reliable performance and margins of stability. But emerging methods challenge this design by dynamically allocating resources to computational tasks, thereby affecting control and mission performance. To apply these emerging strategies, a characterization and understanding of the effects of timing on control and trajectory following performance is required. Going beyond traditional control evaluation techniques, in this paper, we characterize the *trajectory following performance, timing, and control* of a quadrotor UAS under discrete linear quadratic regulator control designed at various sampling rates. We develop a direct relationship between trajectory following performance and the real-time task period (i.e. sampling rate) of the real-time control task allowing future designs to trade off UAS performance and cyber resources at the planning and/or guidance layer. We also introduce new metrics for characterizing cyber-physical quadrotor performance, and lay the groundwork for the application of CPS control methods to quadrotor UASs.**

I. Introduction

The consideration of both computational and physical resources in the design and development of control systems is increasingly important [1]. Advances in autonomy and associated increasing demands on computation mean that "black box" thinking about the computational system will no longer adequately meet the requirements of next generation smart autonomous vehicles, such as in small Unmanned Aircraft Systems (UAS), where computational resources are scarce due to weight and size restrictions. Under these scenarios, the consideration of computational and physical demands can lead to improved overall system and mission performance by dynamically allocating computational and physical resources according to computational and physical performance.

The design, consideration, and intersection of computational, or cyber, resources with communication and physical resources is the objective of Cyber-Physical Systems (CPS) research [2]. To this end, an emerging research area is the design of real-time controllers wherein the time instances of the control input are considered alongside the control input itself. Event-triggered control [3], optimal sampling control [4], and time-varying control [5] are all relatively new methods being developed toward this goal. Similarly, we have developed a co-regulation mechanism [6], [7] that simultaneously, and in response to system performance, adjusts the control inputs alongside the period of the task implementing the control law.

While these strategies adjust computational resources reactively, they do not consider how to plan for this dynamic allocation to meet mission performance objectives. In Figure 1

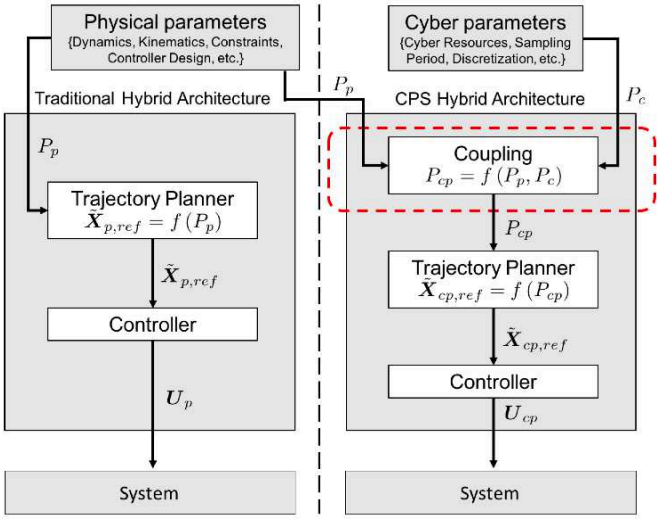


Fig. 1: Traditional Hybrid Architecture (left) and CPS Hybrid Architecture (right). Subscript "p, c" represent "physical" and "cyber" respectively.

we show a traditional hybrid architecture alongside a potential CPS hybrid architecture. In the traditional hybrid robotic architecture (left in Figure 1), a reactive controller provides an input, $\boldsymbol{U}_p$, to the system to command small movements of the controlled object through space and time [8]. To command larger movements, a guidance and deliberative planning layer uses the explicit mathematical relationships between actuators and movement (e.g. kinematics, dynamics, constraints, etc.), P_p, to discretize the desired larger movement into accomplishable reference commands, $\tilde{\boldsymbol{X}}_{p,ref}$, for the reactive controller [9]. Similarly, to develop a CPS hybrid architecture (right in Figure 1), where cyber effectors (i.e. sampling rate) are controlled alongside physical ones, a cyber-physical control input, $\boldsymbol{U}_{cp}$, is given to the combined cyber-physical system to move the physical object through space and time simultaneously with adjustments to the sampling rate. Commanding larger adjustments is accomplished by

A. Shankar and S. Doebbeling are graduate students in the Department of Computer Science and Engineering, and the Department of Mechanical Engineering and Applied Mechanics, respectively, at University of Nebraska-Lincoln, USA. ashankar@cse.unl.edu, sdoebbeling2@unl.edu

J. Bradley is an Assistant Professor in the Department of Computer Science and Engineering, University of Nebraska-Lincoln, USA. jbradley@cse.unl.edu

978-1-5090-4494-8/17/$31.00 ©2017 IEEE

issuing cyber-physical reference commands, $\tilde{\boldsymbol{X}}_{cp,ref}$, from a trajectory planner that requires a corresponding relationship between both cyber and physical effectors and the movement through physical and cyber space and time, P_{cp} (red dashed line around "Coupling" block in Figure 1). This enables a cyber-physical guidance and planning layer to optimize physical *and* cyber trajectories where physical and cyber performance is coupled.

Toward the goal of directly coupling cyber and physical resources to mission performance, we characterize the relationship between control task period (or sampling period), control gain, and reference trajectory following performance of a quadrotor UAS. A careful characterization of the imposed sampling rate of the controller influences stability margins [10], and schedulability [11]. Although the relationship between sampling rate and control performance is understood [12], we take the important next step of characterizing the relationship between sampling rate and *trajectory following performance* - the relationship required to optimally trade off cyber and physical resources at the planning layer. We also introduce new metrics that explicitly measure trajectory following performance of a cyber-physical vehicle system, going beyond traditional controller performance metrics.

II. Background and Related Work

The area of Cyber-Physical Systems research arose out of the area known as "cybernetics," an ancestor of research areas such as control, real-time systems, optimization, autonomy, artificial intelligence, and others [13]. Although the meanings of "cyber" and "physical" have not reached consensus, in this paper we consider "cyber" to be the computation, algorithms, software, and execution of instructions, and the "physical" to be items that occupy physical space and move in the world.

One specific CPS research aim is to investigate new methods, models, and integrations that bridge the divide between discrete computation and continuous control and movement through space [14]. We briefly discuss the role of computation in control and then present related work that considers computation and control simultaneously, including within UAS.

A. Real-Time Computing and Control

In a cyber-physical UAS (hereafter, just UAS), a tight feedback control loop is used to provide reactive behavior to the vehicle [9]. This loop is composed of physical components and cyber components. Initially, physical sensors representing system properties are read, and translated into the digital signals fed into a computer. A controller, modeled in software and executed as a real-time computational task, reads these sensed values and computes a digital control input [11]. The digital control input is then converted into a continuous signal and fed to actuators. This control input is "held" (a zero-order hold) until the control task executes again, restarting the cycle. The question of how often the control task should be executed is governed by the sampled-data assumption, and chosen by the control engineer according to various rules of thumb typically involving noise bandwidth, eigenvalues, and the nyquist frequency [15], [16].

The implementation of the chosen sampling rate is typically represented by assigning a periodicity value to the real-time computational control task. Ideally, either an offline real-time task schedule is then designed, or an online scheduling algorithm is selected [11] and implemented on a Real-Time Operating System (RTOS) to guarantee timing deadlines will be met. Typically, however, because of the complexity of implementation on an RTOS and since certifications and performance guarantees for small UAS are not regulated, a best-effort round-robin architecture (or similar) on a Linux distribution or other microcontroller may be relied upon to provide timing with significant variability. In this scenario, time redundancy is employed to try and mitigate the consequences of missed deadlines [17].

The implications of this cooperation between software execution, real-time progression, and control performance are highly consequential. For system performance and margins of stability, the on board control software must meet deadlines, usually over-designed for worst-case contingency management. For cyber performance, devoting fewer resources to control implies available resources for other computing activities. As a result, on a constrained system the design is a resource allocation and performance trade off.

Several related areas have investigated and leveraged this tradeoff. In [12] an exploration of the impact of sampling rate and control gain on step response is given for a system of inverted pendulums. Very high sampling rates typically result in better performance as the discrete controller approaches its continuous counterpart, though with some caveats [18]. Quality of Service (QoS) research investigates tradeoffs between cyber resource allocation and system performance (including controllers) for various discrete "service" intervals in the cyber system. This research confirms the trend that allocating more cyber resources generally results in better performance [19], [20]. Networked Control Systems has traditionally sought to identify conditions under which stability and performance can be guaranteed for a system wherein sensing, control, and actuation occur on networked computers [21], [22]. More recently, in [23], an optimization strategy is used to identify communication and control inputs simultaneously while taking into account packet loss. Event-triggered control research seeks to maximize cyber resource allocation [3], [24] while maintaining control performance guarantees. While successful in some instances, a fully-developed theory similar to digital control has still eluded the community [25]. A few mechanisms utilizing optimal control techniques to generate control trajectories and sampling instants form a time-varying sampling rate controller [5], [4]. These have been successful theories although with increased computational complexity.

For quadrotors, control systems and their real-time requirements have been studied [26], [27]. Others have implemented an event-triggered control system for attitude stabilization, which is more resource aware [28]. In related work, the authors examined the response-time constraints for a real-time controller implemented onboard the quadrotor [29]. They analyze the response rate of actuators at different

operating conditions in order to design a controller that has an update rate of at least as much as the sampling rate of the various sensors. Seghour et al. [30] implemented a real-time embedded control system for stabilizing a quadrotor, however, they do not reason about the response time or the control rate chosen.

These methods demonstrate and leverage traditional control analysis techniques by assessing controller response to step inputs – the generally accepted strategy in controller design [15]. However, assessing trajectory following performance as a function of cyber resource allocation provides another trade off to exploit in the pursuit of dynamic resource allocation for the holistic cyber-physical system. Here, we extend traditional controller analysis by investigating trajectory following performance and providing the mathematical relationship needed to apply a full cyber-physical control and planning architecture for a UAS. This architecture will enable a more dynamic UAS that can adjust computation in response to performance at both a low, reactive control level, as well as a higher, deliberative planning level.

III. Quadrotor Control Framework

A quadrotor is an under actuated system requiring active control which can be provided by autopilot software. The software periodically generates control signals to ensure stability and drive the quadrotor to a commanded reference in a timely fashion.

Software timing is critical in this process. A controller and planner implemented in software must consider the dynamics and limitations of the vehicle and on-board sensors as well as the timing and scheduling of software tasks in the computer. A low control task period (high sampling rate) can better approximate a continuous model, potentially offering better performance at the expense of computation. Conversely, a high task period (low sampling rate) is easier to achieve computationally amongst many competing autonomy-related tasks, but this may be detrimental to performance. This is compounded in a digital system by sensor values and control inputs that are "sampled and held" until the next time the control task is executed [15]. In this duration, the vehicle continues to react based on its dynamics, possibly becoming unstable.

A. State-Space Model

The nonlinear dynamic motion of a quadrotor UAS can be derived using Newton's and Euler's equations in $\mathbb{R}^3$, [31],

$$\begin{aligned} \boldsymbol{a} &= \frac{\boldsymbol{F}}{m} + \frac{\boldsymbol{F}_{drag}}{m} - \boldsymbol{g} \\ \boldsymbol{\alpha} &= \boldsymbol{I}^{-1}\left[\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}\right] \end{aligned} \tag{1}$$

where $\boldsymbol{a}$ is linear acceleration, $\boldsymbol{F}$ is the net thrust, $\boldsymbol{F}_{drag}$ is forces due to drag, $\boldsymbol{g}$ is the gravitational vector, $\boldsymbol{\alpha}$ is angular acceleration, $\boldsymbol{I}$ is the inertia matrix, $\boldsymbol{\tau}$ represents external torques, and $\boldsymbol{\omega}$ is angular velocity. $\boldsymbol{\alpha}, \boldsymbol{I}, \boldsymbol{\tau}$, and $\boldsymbol{\omega}$ are all calculated about the principle (roll, pitch, and yaw) axes of the quadrotor (i.e. $\boldsymbol{I} = \text{diag}\left(I_{xx}, I_{yy}, I_{zz}\right)$, etc.). Here, the nonlinear equations of motion are written in the frame of reference of the vehicle with respect to an inertial world frame and are used to simulate the quadrotor's motion in response to control inputs. The state of the quadrotor is represented as

$$\begin{aligned} \boldsymbol{X} &= \left(x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}\right)^T \\ (x, y, z) &= \text{position in } \mathbb{R}^3 \\ (\phi, \theta, \psi) &= \text{Euler rotation angles.} \end{aligned}$$

We use a linearized state-space model of the nonlinear equations to design and implement our control strategy. We linearize Eq (1) about a stationary hover, and as a result, the input is biased to account for a vertical component of gravity which does not appear in the system matrix. We have

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{U} \tag{2}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are system and control input matrices respectively, constructed as in [31], and $\boldsymbol{U}$ is the control input. These matrices are functions of m (the mass of the vehicle), g (gravity), D_x, D_y, and D_z (coefficients of drag force acting in each of the coordinate axes), and I_{xx}, I_{yy}, and I_{zz} (inertial moments of the quadrotor's body about the pitch, roll, and yaw axis respectively).

We augment the system matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$, to allow the system to follow different waypoints by adding three integrator states corresponding to the x, y, and z location states of the quadrotor system. This allows for an input waypoint in the form of a positional three vector. The modified state-space equation using the augmented matrices, $\boldsymbol{A}_{aug}$ and $\boldsymbol{B}_{aug}$, is now given as,

$$\begin{aligned} \dot{\boldsymbol{X}}_{aug} &= \boldsymbol{A}_{aug}\boldsymbol{X}_{aug} + \boldsymbol{B}_{aug}\boldsymbol{U} + \boldsymbol{B}_r\tilde{\boldsymbol{X}}_{ref} \\ \boldsymbol{X}_{aug} &= \left(x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, 0, 0, 0\right)^T \\ \boldsymbol{A}_{aug} &= \begin{bmatrix} \boldsymbol{A}_{12\times12} & \boldsymbol{0}_{12\times3} \\ \mathbb{I}_{3\times3} & \boldsymbol{0}_{3\times12} \end{bmatrix} \\ \boldsymbol{B}_{aug} &= \begin{bmatrix} \boldsymbol{B}_{12\times4} \\ \boldsymbol{0}_{3\times4} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{B}_r = \begin{bmatrix} \boldsymbol{0}_{12\times3} \\ -\mathbb{I}_{3\times3} \end{bmatrix}. \end{aligned} \tag{3}$$

The input vector in Eq (3) is $\boldsymbol{U} = \left[\tau_\phi, \tau_\theta, \tau_\psi, F\right]^T$ where τ_ϕ, τ_θ, and τ_ψ are the torques about roll, pitch, and yaw axes respectively, and F is the net thrust exerted by the propellers. For a quadrotor UAS, these can be controlled independently of each other as long as motor saturation is not reached, a common assumption for this simplified model. The additional input vector in Eq (3) contains the location of the target waypoint, $\tilde{\boldsymbol{X}}_{ref} = \left[x, y, z\right]^T$.

B. Real-Time Requirements

If the control signals are not generated in a timely sequence, the quadrotor may become unstable [16]. As a result, flight control code must be executed correctly and completely before a specified deadline. The deadline itself may be categorized as a "hard deadline," since failure to meet it may result in instability [11].

In computer-based control, the sampled-data assumption is often employed to construct digital controllers that are executed periodically according to a real-time schedule. The sampled-data assumption presumes that sensors are read, and the control input is calculated and sent to the actuators at a single, and periodically recurring, instant of time, t. It assumes there is no delay in the states or control input, only

that the control input is then held by the actuators for the entire sampling period, T_d, called a *zero-order hold* (ZOH) [15]. The sampling period of the discrete system is, ideally, matched by the control task execution period enforced by the real-time computing schedule [11]. However, in a real-time system, the only timing guarantee is that deadlines will be met, not that the period between task completion times is consistent. This means that control inputs may not be given at regular time intervals, and there will be state delay in the control input calculation (thus violating the sampled-data assumption) [6].

Consider an implication following this assumption. From the simplified model in Eq (2), the angular roll acceleration at a given time t can be written as:

$$\ddot{\phi}(t) = \frac{\tau_\phi(t)}{I_{xx}}.$$

We compute the rotation angle by integrating $\ddot{\phi}$ twice over the $(k+1)^{th}$ discrete time-interval, where $k \in [0, t/T_d]$, as follows:

$$\phi = \frac{1}{I_{xx}} \int_{kT_d}^{(k+1)T_d} \left(\int_{kT_d}^{(k+1)T_d} \tau_\phi(t) dt \right) dt.$$

In this case the input torque is held constant throughout each sampling period and is only recomputed at the end of the step. Therefore,

$$\tau_\phi(t) = K_{T_d,\phi} \tau_\phi^{max}, \quad kT_d \leq t < (k+1)T_d,$$

where τ_ϕ^{max} is the maximum torque that can be applied, and $K_{T_d,\phi}$ is the roll gain constant chosen suitably for a given T_d. Plugging $\tau_\phi(t)$ into the relationship for ϕ, we have,

$$\begin{aligned} \phi &= \frac{1}{I_{xx}} \int_{kT_d}^{(k+1)T_d} \left(\int_{kT_d}^{(k+1)T_d} K_{T_d,\phi} \tau_\phi^{max} dt \right) dt \\ &= K_{T_d,\phi} \frac{\tau_\phi^{max}}{I_{xx}} T_d^2, \end{aligned} \tag{4}$$

which implies a quadratic relationship between the angular displacement and the amount of time the input is applied for. Using Eq (4) we can compute the maximum amount of time the full input can be applied to the system while keeping the angular rotations within vehicle limitations.

Because of the zero-order hold behavior of discrete systems, Eq (4) has significant implications. The control system has a strong real-time dependency - if a deadline for a new commanded input is missed, and since $K_{T_d,\phi}$ is a constant, the rotation angle may become unbounded. It follows that if the discrete sampling period, T_d, is too large, then the gain, $K_{T_d,\phi}$ needs to be reduced to prevent input saturation.

In a cyber-physical UAS, the same processor executes a multitude of computing tasks, and CPU cycles may become a contended resource that must be allocated appropriately. Small and infrequent delays in meeting the deadlines imposed by the choice of sampling period may lead to a degraded quality of service and instability, but also may be accounted for by time redundancy [19]. While repeated or consistent timing misses may cause an unbounded response on the UAS, committing a larger amount of CPU-time to UAS flight performance may degrade the performance of another important service task (e.g. sensing, data collection). As a result, understanding the limits and implications of sampling rate for the holistic cyber-physical system allows us to balance these resources over the course of a mission.

Finally, it is critical to note that a given digital control strategy is a function of both controller design and selected sampling period. That is, changing the sampling period, even while holding control design variables constant, results in a different value of $\boldsymbol{K}_{T_d}$, effectively re-characterizing the controller itself [15]. The implication is that intelligently trading cyber and physical resources requires us to develop new control techniques that account for the nonlinear relationship between digital, linear control design and sampling rate.

IV. Experimental Setup

Our simulation experiments are designed to demonstrate the effect of varying the software control task period, or sampling period, of flight control tasks as the UAS executes a mission trajectory.

The nonlinear equations discussed in Section III are used to model and simulate the flight of the quadrotor. The controller utilizes a discrete-time linear quadratic regulator (DLQR) strategy on the augmented *linear* system in Eq (3), allowing it to drive the vehicle to a given reference point in space and time. By varying the reference with respect to time, a trajectory is generated for the UAS to follow as the simulation progresses.

We choose LQR control for two reasons: 1) LQR has a closed-form solution and is a stabilizing optimal control algorithm with good margins of stability, thus making it easier to compare controllers with similar performance at different sampling rates; and 2) in finding the optimal gain, LQR minimizes the error on the state vector, $\boldsymbol{X}_{aug}$, and the control input, $\boldsymbol{U}$. This is particularly useful for UAS applications where a large control input may be undesirable. Because our objective is to isolate the relationship between sampling rate and trajectory following performance, we use a MATLAB-based nonlinear equation simulation with full state feedback and do not model external disturbances or sensor noise.

In order to simulate a discrete controller, we first discretize the continuous-time augmented state-space equations into their discrete-time counterparts using a fixed sampling period T_d. The new state-space matrices $\boldsymbol{\Phi}_{T_d}$ and $\boldsymbol{\Gamma}_{T_d}$ are computed from the continuous model in Eq (3), given as,

$$\boldsymbol{\Phi}_{T_d} = e^{\boldsymbol{A}_{aug} T_d} \quad \text{and} \quad \boldsymbol{\Gamma}_{T_d} = \boldsymbol{B}_{aug} \int_0^{T_d} e^{\boldsymbol{A}_{aug} t} dt,$$

such that the equation governing the evolution of system state in the controller at discrete time steps, k, is now,

$$\boldsymbol{X}[k+1] = \boldsymbol{\Phi}_{T_d} \boldsymbol{X}[k] + \boldsymbol{\Gamma}_{T_d} \boldsymbol{U}[k] + \boldsymbol{B}_r \tilde{\boldsymbol{X}}_{ref},$$

where $\boldsymbol{U}[k]$ is the input vector to the system [15]. Note here that $\boldsymbol{X}[k+1]$ and $\boldsymbol{X}[k]$ stem from the augmented state vector, $\boldsymbol{X}_{aug}$.

Once the system matrices are computed for a given sampling period, we design a DLQR controller to maintain a

Parameter	Value	Parameter	Value
g	$9.80665\,\mathrm{m/s^2}$	m	$0.515\,\mathrm{kg}$
D_x	$0.0075\,\mathrm{kg/s}$	I_{xx}	$0.0040\,\mathrm{kg\,m^2}$
D_y	$0.0075\,\mathrm{kg/s}$	I_{yy}	$0.0040\,\mathrm{kg\,m^2}$
D_z	$0.015\,\mathrm{kg/s}$	I_{zz}	$0.0044\,\mathrm{kg\,m^2}$
$\boldsymbol{Q}$	$10\mathbb{I}_{15\times15}\boldsymbol{q}$	$\boldsymbol{R}$	$2\mathbb{I}_{4\times4}$
$\boldsymbol{q} = [100\ 100\ 100\ 100\ 100\ 100\ 10\ 10\ 10\ 1\ 1\ 1\ 50\ 50\ 50]^T$			

TABLE I: System Constants

stable hover at a given reference. Note that in the DLQR, the gain matrix, $\boldsymbol{K}_{T_d}$, is specific to the sampling period T_d used to generate the $\boldsymbol{\Phi}_{T_d}$ and $\boldsymbol{\Gamma}_{T_d}$ matrices [15]. This implies that a real-time system designed with the control task executed at a different periodicity, or under jitter or missed deadline conditions, will result in poor performance of the controller.

In order to study the effect of a changing sampling period, we perform a sweep over a range of values for T_d, generating a corresponding set of $\boldsymbol{\Phi}_{T_d}, \boldsymbol{\Gamma}_{T_d}$, and $\boldsymbol{K}_{T_d}$ matrices. Because we are interested in a highly accurate relationship between sampling period and trajectory following performance, we use a fourth-order Runge-Kutta ordinary differential equation solver, `ode45` in MATLAB, to simulate the system using the nonlinear equations in Eq (1). However, `ode45` is a continuous-time solver, unsuited to the zero-order hold paradigm. As a result, we leverage it as part of a larger simulation technique designed to simulate both the correct zero-order hold behavior and corresponding transients of the system response for each time period during which the input is held, $0 \leq j < m$ where j represents an internal `ode45` time step on $kT_d \leq t < (k+1)T_d$. These modifications are described as follows.

An outer loop iterates over discrete time steps, k, computing and holding $\boldsymbol{U}[k] = -\boldsymbol{K}_{T_d}\boldsymbol{X}[k]$ for the sampling period duration T_d. That is, $\boldsymbol{U}(t) = \boldsymbol{U}[k]$, where $kT_d \leq t < (k+1)T_d$. Within each sampling period, $\boldsymbol{U}[k]$ is passed and held as an input to the nonlinear system model, which is simulated using `ode45`. The initial system state for each discrete step is the final state propagated by `ode45` in the previous iteration. Because `ode45` is a one-step solver, the output from each execution of `ode45` can be appended to the previous one to put together a complete continuous system response. This ensures that the control system follows a discrete-time sample and hold behavior, but we also obtain the transient response for each sampling interval. A pseudocode for the algorithm that simulates the system for a specific sampling period is shown in Algorithm 1.

The quadrotor model used for the simulations is based on an Ascending Technologies Hummingbird [32], which is a general purpose medium-sized research UAS used in the NIMBUS Lab[1]. All experiments are run with the system constants shown in Table I for the Hummingbird quadrotor design.

Algorithm 1: Algorithm to simulate the control of quadrotor flight at one discrete sampling rate.

Data: Nonlinear system model, $f(X)$, $sampling\_period$
`// initialize system constants`
$T_d \longleftarrow sampling\_period$
$lin\_model \longleftarrow$ `linearize`$(f(X), T_d)$
$Q \longleftarrow 10 \cdot \mathbb{I}_{15\times15} \cdot q$
$R \longleftarrow 2 \cdot \mathbb{I}_{4\times4}$
begin
  $\Phi_{T_d}, \Gamma_{T_d} \longleftarrow$ `discretize`(lin_model, T_d)
  $K_{gain} \longleftarrow$ `dlqr`$(\Phi_{T_d}, \Gamma_{T_d})$
  $k \longleftarrow 0$
  $X_{all} = [\,]$
  $X_{init} \longleftarrow$ `initial_state()`
  while $kT_d \leq simulation\_length$ **do**
    $U_k \longleftarrow$ `input_vector`(X_{init}, K_{gain})
    $[X_1 \dots X_m]$ = `Simulate`$(f(X), T_d, U_k\ X_{init})$
    $X_{all} = [X_{all}; X_1 \dots X_m]$
    $k \longleftarrow k+1$
    $X_{init} \longleftarrow X_m$ `// new initial state`
  end
end

V. CPS METRICS

To quantify the relationship between sampling rate and trajectory following performance, we introduce several different CPS metrics. Each of these metrics can be computed for a specific sampling rate and simulation, which, can then be combined to form an explicit mathematical relationship between sampling rate and trajectory following performance. Although we characterize these metrics in the context of a quadrotor UAS, they are, in principle, more generally applicable to a broad class of cyber-physical vehicle systems.

1) Cumulative Time-Weighted State Error

In our experiments, the reference point, $\tilde{\boldsymbol{X}}_{ref}$, is given at specific time intervals throughout the simulation. Therefore, by changing the position of the reference waypoint with respect to time, the UAS can be commanded through a desired trajectory. We define the state error at any given time step, k, as the squared Euclidean distance between $\tilde{\boldsymbol{X}}[k]$ (the position states from the current state vector $\boldsymbol{X}[k]$) and the corresponding reference state vector at that time, $\tilde{\boldsymbol{X}}_{ref}[k]$. Given the reference state to which the controller must drive the system, a cumulative time-averaged state error (CSE) can be defined for the position state vector over the entire mission. Weighting each state error by the simulation time yields the following equation for cumulative time weighted average of state error,

$$\text{CSE} = \frac{1}{n}\sum_{i=0}^{n} t_i \left|\tilde{\boldsymbol{X}}(t_i) - \tilde{\boldsymbol{X}}_{ref}(t_i)\right|^2 \tag{5}$$

where $|\cdot|$ represents Euclidean distance, $\tilde{\boldsymbol{X}}$ is the vehicle's position in $\mathbb{R}^3$, t_i is `ode45`'s discretization of continuous time (t), n is the total number of internal simulation steps, i.e. $i \in [0, n]$. Weighting by time has the advantage of more aggressively penalizing the state error as time progresses, while having a smaller weight associated with initial offsets in

[1] http://nimbus.unl.edu

the system. This definition of state error as a metric for system design captures the effectiveness of a cyber control system in driving the state of the physical system to the reference state within a short amount of time, and with minimal overshoot.

2) Translational Bounds

The ability to place bounds on maximum offsets from a desired trajectory is useful in planning mission objectives and helps to identify worst case flight envelopes and failure states. As the controller responds to commanded target waypoints, this metric of maximum state error (MSE) determines the farthest point the UAS reached from the ideal desired trajectory line connecting two successive target waypoints, $\boldsymbol{L} = \tilde{\boldsymbol{X}}_{ref,next}(t_i) - \tilde{\boldsymbol{X}}_{ref,prev}(t_i)$,

$$\text{MSE} = \max\left(\frac{|\boldsymbol{L} \times (\tilde{\boldsymbol{X}}_{ref,prev}(t_i) - \tilde{\boldsymbol{X}}(t_i))|}{|\boldsymbol{L}|}\right). \quad (6)$$

This metric is used to set a standard for mission success which hinges on whether or not the UAS remained within a desired maximum distance from the given path throughout execution and could be used to determine possible failure states in order to invoke a contingency control strategy.

3) Control Effort

For a quadrotor UAS, minimization of control effort is essential for decreasing power and energy demands thereby preventing possible damage to components and potentially increasing vehicle endurance.

To analyze the control effort of the system, we compute the time-averaged control effort (CE) over all simulation time as follows:

$$\text{CE} = \frac{1}{n}\sum_{i=0}^{n} |\boldsymbol{U}(t_i)|^2 \, t_i, \quad (7)$$

where $\boldsymbol{U}(t_i) = \boldsymbol{U}[k] = \text{const.}$ on $kT_d \leq t_i < (k+1)T_d$. As before, weighting the value of the control effort with the simulation time rewards the natural response of the system to a reference step, which, generally would require less control effort as error is reduced. This metric is also proportional to the energy consumed for propulsion and is approximately proportional to total energy consumed in a system where propulsion dominates energy resources.

In part, this metric is motivated by the mathematical realization that a controller with higher gains may more quickly converge to the reference state by generating larger control inputs for a shorter amount of time (without saturating). This metric favors such a controller, as compared to one which applies smaller inputs for a longer duration, thus, taking longer to converge.

Intuitively, we expect each of these metrics to increase as the sampling period is increased. That is, with longer sampling periods there should be higher state error, a higher control effort (CE), and a typically larger maximum deviation from the ideal trajectory.

VI. Results

We now present the results of several important test cases representing various scenarios we regularly find in our UAS missions.

A. Traditional Disturbance Rejection Experiment

We begin by first characterizing the performance of the controller in rejecting a disturbance represented by an initial non-zero attitude angle. This represents a traditional control system performance metric - evaluating a step response. At time $t = 0$, the system is initialized at the origin of the inertial space $\boldsymbol{p} = (0, 0, 0)$ with $\phi = 0.2$ rad and other components of the state vector set to zeros.

Figure 2a shows the progression of ϕ for different sampling rates as the controller brings the system to a stable hover at the origin. As expected, a higher sampling period results in a longer settling time and larger overshoot.

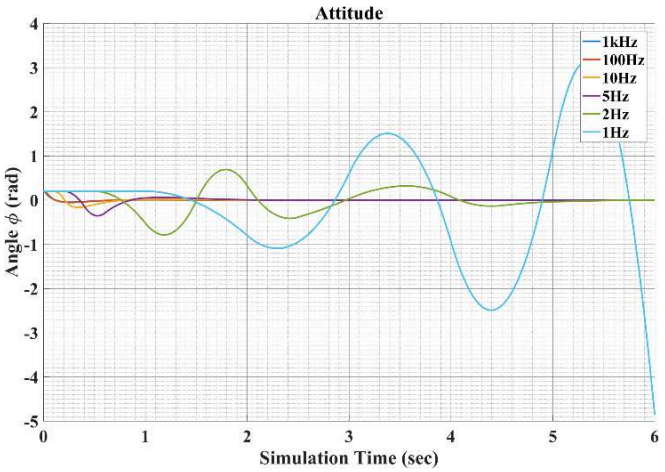
Although DLQR is a stabilizing controller, it stabilizes the linear approximation of the nonlinear system. However, the stable region of a closed-loop nonlinear system changes with the sampling period [33]. As a result, there are states that may exceed the bounds on disturbances from which the DLQR controller can recover. We hypothesize this is the case for the DLQR controller designed and operated at a 1.0 s sampling period in Figure 2a. The initial condition 0.2 rad exceeds the region of stability for the DLQR controlled nonlinear system. The control input required to reject the disturbance is shown with different sampling rates in Figure 2b. We note that for high sampling rates, as anticipated, the control input changes in a much smoother fashion, but the maximum control effort required is higher. For lower sampling rates, however, the maximum control effort is smaller in magnitude, and the system takes longer to settle.

B. Trajectory Following Experiments

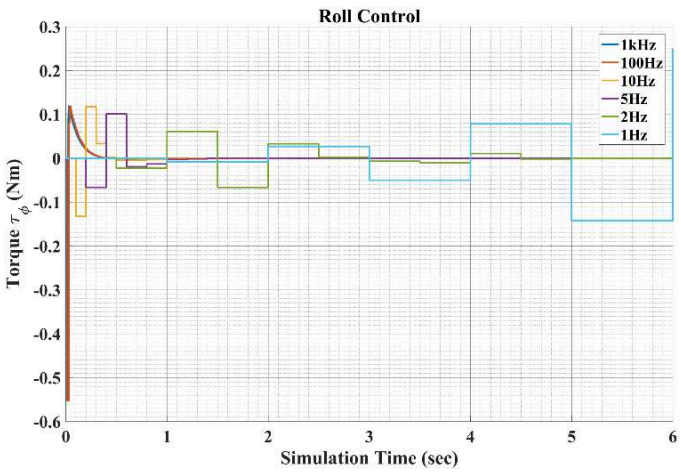
We now assess the controller's performance in following a single, straight line trajectory by driving the vehicle to a point $\boldsymbol{p} = (x_1, y_1, z_1)$ in space, starting from a stable hover at the origin. The commanded reference is held constant throughout the length of the simulation so that the controller causes the vehicle to go to, and hover at, $\boldsymbol{p}$. In the following subsections, we use this test to analyze the various metrics defined previously.

Finally, to asses complex trajectory following performance we develop trajectories consisting of reference waypoints and issue commands to the vehicle to follow. We design the framework such that a new waypoint might be made available at any time instant, whether the vehicle has reached its current waypoint or not. Therefore, if several distant waypoints arrive in quick succession, it is not necessary that the vehicle would ever reach any single one of them. This design decision was based on a cyber-physically co-regulated and co-optimized UAS a mission planner that could decide that reaching each waypoint in a complex trajectory may be subjugated by the desire to conserve resources by allowing for less aggressive control at the expense of precision.

Figure 3 shows the path taken by the UAS as it follows five commanded waypoints in space. The effect of a low sampling rate is clear for certain course legs, most notably the first and the last ones. This becomes less predictable as the simulation progresses. For instance, for three consecutive waypoints $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$, if the angle between the two consecutive waypoints, $\boldsymbol{p}_i\boldsymbol{p}_j$ and $\boldsymbol{p}_j\boldsymbol{p}_k$, is obtuse, then it is possible



(a) Rejecting an initial roll angle of $\phi = 0.2$ radians for different sampling rates.



(b) Rolling torque, τ_ϕ, generated by the controller at different sampling rates in order to bring the vehicle to a stable hover.

Fig. 2: Disturbance rejection on the roll axis at different sampling rates. A higher sampling rate results in higher controller gains and more aggressive control. Lower sampling rates result in a more narrow control input operating range, but the response also takes longer to converge to the reference state. Note that the response for 1 kHz (blue) is nearly identical to the 100 Hz plot, and is obscured by it.

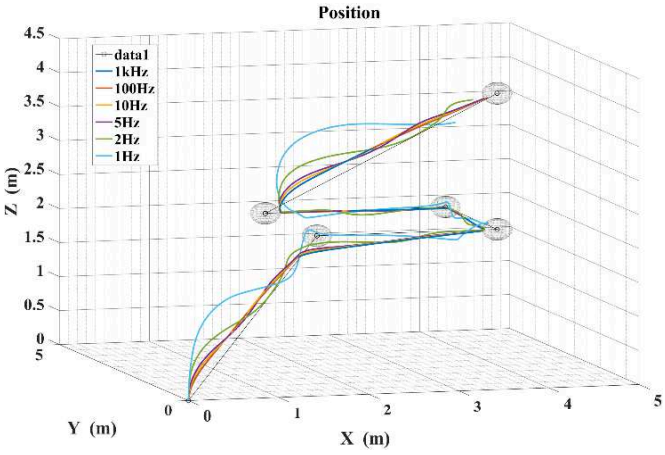


Fig. 3: The paths taken by the UAS as it follows five commanded waypoints in space at different sampling rates of the controller.

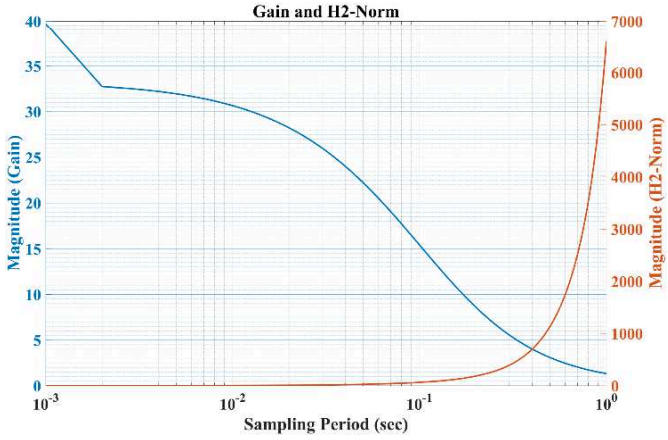


Fig. 4: Variation in the controller gain and the H_2 norm of the discretized system across different sampling periods.

that the vehicle undershoots the waypoint $\boldsymbol{p}_j$ and is then better poised to reach $\boldsymbol{p}_k$. Because DLQR is a stabilizing controller, as long as the states of the vehicle remain within the region of stability of the closed-loop nonlinear system, the controller will always recover. As a result, contrary to the step response in Figure 2a, where initial conditions were beyond the disturbance limits that ensure stability, in Figure 3 the vehicle successfully navigates the trajectory, although with reduced performance. If design of the control system includes similarly large sampling period, a more rigorous mathematical characterization of the bounds on disturbances and regions of stability, similar to [33], is needed.

C. Characterizing the Relationship Between Trajectory Following Performance and Sampling Period

We now demonstrate the relationship between trajectory following performance of our UAS and the sampling period of the software control task using the metrics we discussed in Section V.

Variation of Gain and H_2-norm

We noted in Eq (4) that for a larger sampling period, the gain of the system should decrease. Since we have multiple elements in the input vector, we quantify the control gain here as the L2-norm of the $\boldsymbol{K}_{T_d}$ matrix. Another useful analysis tool is the H_2-norm, which represents the energy of the output of the system [18]. This tool can be used to identify potentially destabilizing intermediate sampling periods of the system if the H_2-norm is infinite at a given sampling period. We perform a sweep on a wide range of sampling periods and plot the variation in the controller gain and the H_2-norm of the system in Figure 4. Much like the analysis in [12], gain decreases with sampling period.

In our case, where the DLQR design parameters remain constant as we change sampling period, analysis of the gain vs. sampling period curve in Figure 4 can be used to select the lowest feasible sampling period of control as long as motor saturation is not reached. In practice, however, sampling rate will most likely be limited by constraints in the cyber system (i.e. how much processing time can be devoted to control computation).

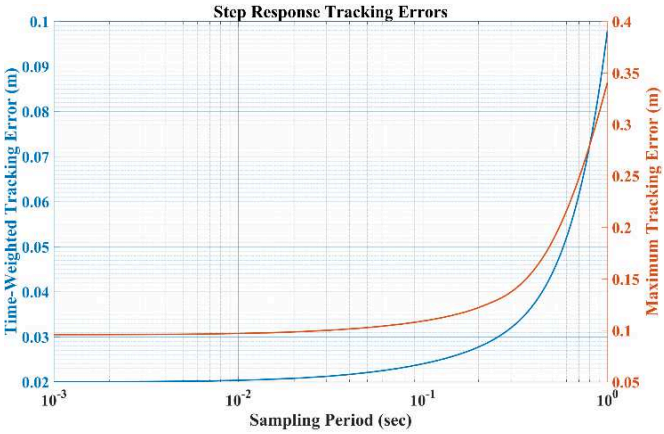


Fig. 5: The change in average trajectory tracking error and the maximum deviation from the trajectory for a single leg (step response) as the sampling period changes.

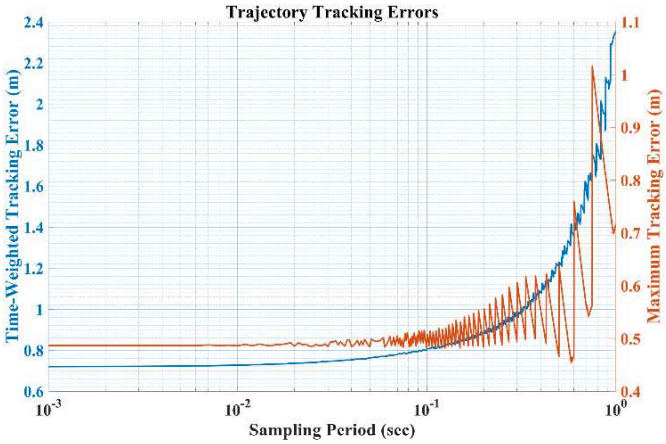


Fig. 6: The change in average trajectory tracking error and the maximum deviation from the trajectory in Figure 3 as the sampling period changes.

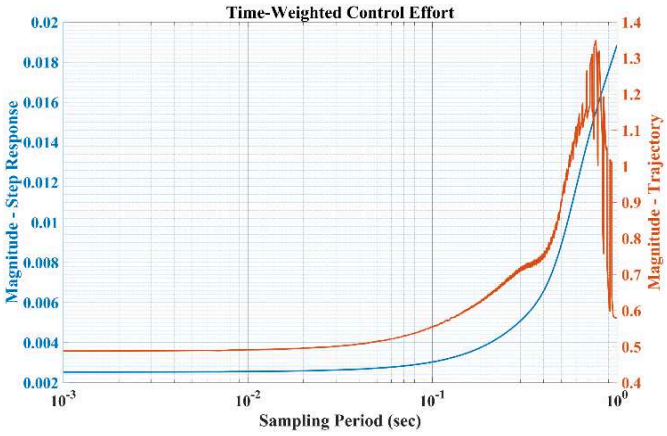


Fig. 7: The increase in time-weighted average control effort metric against increasing sampling periods for a step response and for the trajectory in Figure 3.

The high-gain operation of the controller at small sampling periods can potentially saturate the actuators, thereby violating the assumption that the thrust and torques on each of the axes are independently controllable. Knowing the matrix $\boldsymbol{K}_{T_d}$ designed for a specific T_d, and a given state vector $\boldsymbol{X}(t)$ at time t, we can check for saturation:

$$\boldsymbol{U}(t) = -\boldsymbol{K}_{T_d}\boldsymbol{X}(t) \;\; \leq \boldsymbol{U}_{max}$$

where $\boldsymbol{U}_{max}$ is determined appropriately using maximum rotor thrust from system specifications [32].

State Error

We introduced the cumulative time-weighted state error (CSE) and defined it as a metric to characterize the performance of a controller over a trajectory leg. Observing the traversed paths in Figure 3, we expect this metric to increase in magnitude as the sampling period increases. The maximum deviation from the trajectory leg is also expected to increase, as the sampling period becomes longer, due to the sample and hold nature of control. The trend in these two metrics is captured first in Figure 5 representing a single trajectory leg from the origin to a waypoint (i.e. a step response), as a function of sampling period. In Figure 6 we again show the trend in these two metrics, but this time for the entire trajectory shown in Figure 3. While the tracking errors for a step response follow a smooth trend as sampling period varies, the tracking errors for the trajectory (and similarly control effort) do not. We speculate this is a result of internal resolution changes and numerical error in MATLAB's `ode45` solver coupled with effects previously discussed in regard to path geometry in which the vehicle may find itself poised differently in regards to reaching a newly generated target waypoint.

Control Effort

Using our control effort metric, CE (Eq. (7)), as the sampling period increases, we expect the value of CE to increase since the controller will operate at a lower gain, but for a longer amount of time. However, it is important to note that the magnitude of the maximum control input generated by the controller will now be smaller as in Figure 2b.

We accumulate the time-weighted control effort expended over a given mission for various sampling rates and plot the trend, in Figure 7, against the sampling period for the controller as it drives the vehicle to a stable hover at a single waypoint (blue plot) and also as it drives the vehicle through the entire trajectory (red plot).

VII. Discussion of Results

To develop a cyber-physical UAS, we must consider both limitations in cyber resources and performance expectations of the physical system. Our results capture this trade-off and imply that control analysis must go beyond traditional controller performance assessment and include trajectory following performance in order to trade off resources at each level of the autonomy architecture.

Intuitively, a controller designed to operate at a higher sampling period may cause undesired overshoots in the system state because the dynamics of the system act faster than appropriate control signals are generated. Additionally, the control effort (CE metric) increases, implying the system may need to spend more energy over a longer period of time, though with smaller power requirements. The benefit, however, is the increased availability of computing resources for other tasks (vision, data collection, sensing, etc.).

Choosing lower sampling periods allows for the selection of a higher-gain controller resulting in increased precision and the ability to conduct more aggressive maneuvers. Unfortu-

	Circle						Square						Spiral					
Rate(Hz)	1k	500	100	50	10	2	1k	500	100	50	10	2	1k	500	100	50	10	2
CSE	0.681	0.682	0.686	0.691	0.734	1.057	0.517	0.517	0.520	0.523	0.554	0.827	3.582	3.590	3.608	3.632	3.850	5.433
MSE	0.375	0.375	0.374	0.374	0.373	0.390	0.310	0.310	0.310	0.309	0.303	0.296	0.985	0.985	0.985	0.985	0.985	0.985
CE	0.0021	0.0021	0.0021	0.0022	0.0027	0.0203	0.0247	0.0247	0.0251	0.0255	0.0309	0.2368	0.1200	0.1201	0.1204	0.1216	0.1548	0.1570

TABLE II: Table summarizing the trend in the proposed metrics for different trajectories.

nately, this trade off results in large control inputs which may adversely affect mechanical actuators. For the cyber system, a smaller sampling period adversely affects the schedulability of additional tasks that the system must perform, particularly aperiodic tasks which are often scheduled in available slack time in the cyber system [11].

However, from the above results, the state error and control effort of the system follow a relatively flat curve as sampling period increases up to a range of approximately 0.02 s - 0.1 s. Therefore, in our idealized, no-noise simulation environment, the sampling rate of control can be lowered to this range without incurring a significant cost in state error or control effort. This illustrates the opportunity for savings in cyber resources while sampling at 0.02 s versus 0.002 s provided we can design appropriate controllers that are robust to noise and disturbances.

Table II summarizes the trend in the above metrics for select sampling periods as the vehicle moves along several more complex trajectories. We consider three additional trajectories (a circle, a square, and a spiral) and compute the same metrics across the entire mission. Once again, large changes in error do not occur until the sampling rate reaches a range of approximately 0.02 s - 0.1 s. Also note that the values calculated for each of these evaluation metrics depend largely on the geometry of the trajectory and how it is defined. That is, following a more complex trajectory, or one defined by a higher number of waypoints, especially those consisting of smooth curves, may result in unique results. This suggests the importance of a high level CPS trajectory planner and CPS controller that is able to dynamically adjust resources, thus enabling higher precision following of complex trajectories and reducing resources for following simpler ones.

Utilization metric

In a UAS there exists a set of computational tasks to which a scheduler must allocate appropriate resources to ensure computing deadlines are met. This task set may contain tasks with non-deterministic execution times, varying logical priority, sporadic, aperiodic, and other periodic tasks [11]. For example, a UAS executing a camera-based surveillance mission might have computationally intensive vision processing algorithms, guidance and navigation tasks, and a top-level planner in addition to the on-board state-estimation, sensor fusion, and attitude stabilization algorithms. To complicate this further, there may also be aperiodic tasks with quick deadlines that are triggered by a user input from a ground station. In such a scenario, it is critical to ensure that task priorities and deadlines in the real-time schedule be set correctly and perform predictably, but it is also an opportunity

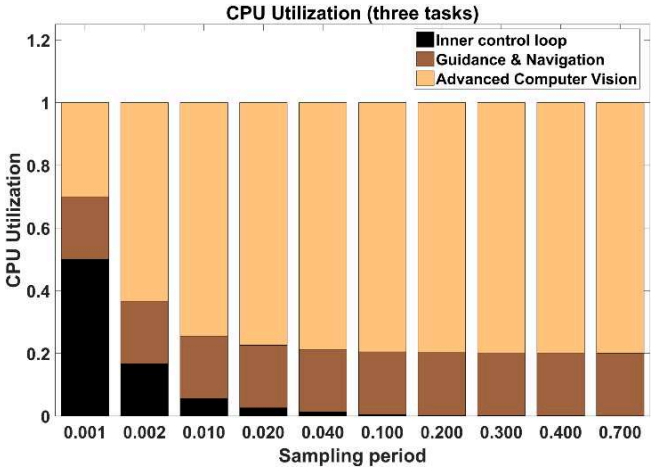


Fig. 8: CPU utilization and availability for different tasks, assuming that the attitude controller has a worst case execution time of 0.2 ms.

to dynamically adjust task priorities and deadlines depending on the environment, system performance, and mission context.

In this context, it is useful to examine resource utilization of the control task in the real-time system as a metric for cyber performance analysis. The utilization of the i^{th} real-time task is computed as $util_i = e_i/p_i$, where e_i and p_i are execution time and the period of the task [11]. The total resource utilization is then the summation over all tasks. Since e_i is difficult to know beforehand, it is usually substituted with the worst-case execution time (WCET) [34]. Given two processes with similar CPU requirements, the one with a larger period will have smaller CPU utilization. This drives efforts towards developing on-demand controllers that can guarantee performance even at higher sampling periods. This frees up cyber resources, which, the scheduler might allocate to other tasks in the system. As an example, in a hover, the likelihood of running into a stationary object is low. This may be an opportunity to turn off a laser scanner and reduce the priority and task period of the corresponding sensor task, thus, freeing up cyber resources for communicating collected data to a ground station. Figure 8 shows the decrease in CPU utilization for the attitude control loop of our UAS as sampling period increases, thereby accommodating other tasks which may have larger WCETs.

VIII. Conclusions

Control and real-time computing are coupled by implementing control laws on a digital device requiring the periodic execution of a task. Characterizing this coupling and the performance of the system allows us to design planning algorithms that trade off cyber and physical resources and ensure predictable performance. In this paper, we have

investigated and quantified the effects of varying sampling periods of a controller on a quadrotor UAS as it follows various trajectories. This provides a mathematical relationship for developing a cyber-physical planning algorithm that trades off cyber and physical resources for improved mission performance.

We also introduced new metrics that quantify both the physical and cyber performance of a quadrotor UAS following a reference trajectory. The results provide us with a means for developing a higher-level CPS planner that computes coupled cyber-physical trajectories and reference commands for a low-level reactive cyber-physical control strategy. The results also serve as a pointer to the awareness for considering timing requirements while designing control laws. This characterization is a step toward the development of a cyber-physical UAS architecture that incorporates cyber and physical resources at every layer of the architecture to improve performance, efficiency, and robustness.

Future work will focus on the development a cyber-physical control strategy robust to noise and disturbances. We are also preparing to conduct flight experiments on our quadrotor UAS with accompanying analysis under similar trajectory tests to verify our results.

IX. Acknowledgments

This work was supported in part by NSF award #1638099 and NSF award #IIA-1539070.

References

[1] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, p. 32703285.

[2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.

[3] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *Automatic Control, IEEE Transactions on*, vol. 57, no. 5, pp. 1291–1297, 2012.

[4] E. Bini and G. M. Buttazzo, "The optimal sampling pattern for linear control systems," *Automatic Control, IEEE Transactions on*, vol. 59, no. 1, p. 7890, Jan. 2014.

[5] K. Kowalska and M. Mohrenschildt, "An approach to variable time receding horizon control," *Optimal Control Applications and Methods*, vol. 33, no. 4, p. 401414, 2012.

[6] J. M. Bradley and E. M. Atkins, "Coupled cyber-physical system modeling and coregulation of a CubeSat," *IEEE Transactions on Robotics*, vol. 31, no. 2, p. 443456, Apr. 2015.

[7] ——, "Toward continuous state-space regulation of coupled cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, p. 6074, Jan. 2012.

[8] R. Murphy, *Introduction to AI Robotics*. MIT press, 2000.

[9] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[10] L. H. Keel and S. P. Bhattacharyya, "Stability margins and digital implementation of controllers," in *Digital Controller Implementation and Fragility*. Springer, 2001, pp. 13–24. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4471-0265-6_2

[11] J. Liu, *Real-Time Systems*. Prentice Hall, 2000, lCCB: 99051522.

[12] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney, "Task scheduling for control oriented requirements for cyber-physical systems," in *Real-Time Systems Symposium, 2008*. IEEE, 2008, p. 4756.

[13] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 1965, vol. 25.

[14] "Cyber-Physical Systems (CPS) (nsf17529) — NSF - National Science Foundation." [Online]. Available: https://www.nsf.gov/pubs/2017/nsf17529/nsf17529.htm

[15] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

[16] N. S. Nise, *Control systems engineering*, 6th ed. John Wiley and Sons, 2011.

[17] D. Gurdan, J. Stumpf, M. Achtelik, K. M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 361–366.

[18] S. L. Osburn and D. S. Bernstein, "An exact treatment of the achievable closed-loop H_2 performance of sampled-data controllers: From continuous-time to open-loop," *Automatica*, vol. 31, no. 4, p. 617620, 1995.

[19] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automated flight control," *Computers, IEEE Transactions on*, vol. 49, no. 11, pp. 1170–1183, 2000.

[20] F. Xia, L. Ma, J. Dong, and Y. Sun, "Network QoS management in cyber-physical systems," in *Embedded Software and Systems Symposia, 2008. ICESS Symposia'08. International Conference on*. IEEE, 2008, p. 302307.

[21] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. of the IEEE*, vol. 95, no. 1, p. 138162, 2007.

[22] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, p. 8499, 2001.

[23] X. Cao, P. Cheng, J. Chen, and Y. Sun, "An online optimization approach for control and communication codesign in networked cyber-physical systems," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 439–450, 2013.

[24] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 4719–4724.

[25] K. J. Åström and B. M. Bernhardsson, "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2. IEEE, 2002, p. 20112016.

[26] P. Castillo, P. Albertos, P. Garcia, and R. Lozano, "Simple real-time attitude stabilization of a quad-rotor aircraft with bounded signals," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 1533–1538.

[27] J. F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare, "Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter," *Control Engineering Practice*, vol. 19, no. 8, pp. 790 – 797, 2011.

[28] J. F. Guerrero-Castellanos, J. J. Téllez-Guzmán, S. Durand, N. Marchand, and J. U. Alvarez-Muñoz, "Event-triggered nonlinear control for attitude stabilization of a quadrotor," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, May 2013, pp. 584–591.

[29] Corona-Sánchez, J. J. and Rodríguez-Cortés, H., "Experimental real-time validation of an attitude nonlinear controller for the quadrotor vehicle," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, May 2013, pp. 453–460.

[30] S. Seghour, M. Bouchoucha, and H. Osmani, "From integral backstepping to integral sliding mode attitude stabilization of a quadrotor system: Real time implementation on an embedded control system based on a dspic μc," in *Mechatronics (ICM), 2011 IEEE International Conference on*, Apr. 2011, pp. 154–161.

[31] R. Beard, "Quadrotor dynamics and control rev 0.1," Tech. Rep., 2008. [Online]. Available: http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2324&context=facpub

[32] "AscTec Research UAVs." [Online]. Available: http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/

[33] T.-T. Lee and S.-H. Lee, "Discrete optimal control with eigenvalue assigned inside a circular region," *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 958–962, October 1986.

[34] R. Wilhelm, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, P. Stenström, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, and R. Heckmann, "The worst-case execution-time problem - overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, pp. 1–53, Apr. 2008.