Impact and Influence of Cyber-Physical Systems Research on Autonomous Aerospace Systems

Justin Bradley∗
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Cody Fleming† and Kristin Y. Rozier‡
Iowa State University, Ames, IA 50011, USA

Amy Pritchett§
Penn State University, University Park, PA 16802, USA

Cyber-Physical Systems, as a discipline, is relatively new, appearing prominently between 2000-2010, but has rapidly made contributions in many disciplines. The aerospace industry has been a primary application domain for CPS from the beginning, with much focus on autonomous systems, particularly Unmanned Aircraft Systems. Aerospace systems provide an opportune, complex, safety-critical system on which to prove out CPS research, techniques, and strategies where robustness, agility, and provable performance guarantees are important. In this paper we survey and discuss the synthesis of core CPS research areas into four key areas of autonomous aerospace systems. We illuminate how these have been applied across aviation and space systems that interface with human operators and/or bystanders. We also discuss how computing and select CPS concepts more prominently applied in other applications could be applied in aerospace systems. Finally, we elaborate on existing shortcomings in cyber-physical aerospace systems and propose future avenues of exploration and application to pursue.

I. Introduction

Cyber-physical systems (CPSs) “combine[s] cyber systems (computational systems such as microprocessors and digital communication networks) with other physical systems (electromechanical, chemical, structural, and biological systems)” [1]. Generally, this applies to a large swath of engineered systems many of which have their own niche research areas. Likewise, the topics of research that contribute to CPS span a wide array of disciplines. This may lead some to conclude that CPS research is a catch-all term attempting to supersede or capture individual, more targeted areas of research. We distinguish between CPS research and CPSs and remark that many existing research disciplines contribute to the advancement of engineered CPS but may not be thought of as “CPS research.” We define “CPS research” as research focused on advancing the “fundamental scientific and engineering principles that underpin the integration of cyber and physical elements across all application sectors.”∗∗. As a matter of practice, a few research areas have been identified as CPS research: control, data analytics, and machine learning including real-time learning for control, autonomy, design, Internet of Things (IoT), mixed initiatives including human-in- or

∗Associate Professor, School of Computing, AIAA Associate Fellow
†Associate Professor, Mechanical Engineering, AIAA Senior Member
‡Black & Veatch Associate Professor, Department of Aerospace Engineering, AIAA Associate Fellow
§Professor and Department Head of Aerospace Engineering, AIAA Fellow
human-on-the-loop, networking, privacy, real-time systems, safety, security, and verification; this list fluctuates as different topics emerge at prominent CPS venues.

To avoid yet another CPS survey paper (of which there are many), in this paper we focus on surveying the synthesis of control, autonomy, design, human-in- or human-on-the-loop, real-time systems, safety, and verification into autonomous aeronautical and astronautical systems (see Figure 1). While an individual survey paper could be written about each of these individual research areas applied to aerospace systems, we target the integration and synthesis of these concepts into the holistic design and deployment specifically targeted at aerospace systems and their attributes that include real-time operation of flight dynamics and strict requirements for safety. The intent is to give the reader a view of how CPS research is impacting integration efforts in deployed aerospace systems, the limitations of those efforts, and what areas may need more consideration.

II. CPS Origins, Surveys, and Challenges

Conceptually, cyber-physical systems stem from the field of cybernetics – the scientific study of control and communication in the animal and the machine [2, 3] – and hence has been around in various forms for many years. Of more relevance, the term “Cyber-Physical Systems,” as a discipline, was coined in 2006 by Helen Gill at the National Science Foundation [4]. Since then the field has grown quickly and rapidly as researchers and practitioners wrestle with the science of cyber and physical integration in engineered systems. In the aerospace sector, the area of most rapid uptake of CPS concepts has perhaps been autonomous Unmanned Aircraft Systems (UASs) wherein significant constraints on Size, Weight, and Power (SWaP) have demanded novel methods for engineering capable, safe, resilient flight.

A challenge in the broader community appears to be how best to understand what constitutes a CPS and CPS research. For such a young discipline, a relatively large number of related surveys seek to summarize the state of the art and describe how CPS research applies to various disciplines, and the challenges that lie ahead in advancing the field. Here, we discuss existing CPS surveys and describe how our work here differs.

A. CPS Surveys

A large number of CPS survey papers review existing work in a broad array of research fields (as opposed to application sectors). This speaks to the power of CPS research to impact many research areas as well as the excitement of researchers in applying CPS concepts to their core areas. The areas of security and safety have the largest number of surveys [5–12], and includes surveys of CPS concepts in blockchains [13, 14], secure control [5], and even a survey of surveys in security in CPS [9]. These surveys discuss the unique challenge of tightly integrating cyber and physical components to detect both traditional computation-based attacks, as well as physical infrastructure attacks, ensure privacy, and maintain robust safety.

A number of surveys examine a cornerstone of CPS research – the development of novel modeling schemes [15–19] spanning mixed discrete/continuous systems, verifiable models, and models for model-based design paradigms [20–23]. Other central research areas such as control [24–26], data [27, 28], and networking [29, 30] have had multiple CPS-based surveys published. Testing has received less attention [31], suggesting perhaps that the community has yet to develop many strong CPS testing techniques. Related to this is a single survey on test case generation for software in safety-critical CPS [32], indicating opportunity for software engineering and CPS researchers to collaborate.

CPS surveys have also addressed application domains such as the medical [33–35], and power/smart grid sectors [11, 36–38]. Finally, researchers have recognized the importance of integrating fundamental CPS research concepts into education and have compiled surveys in this important area [39, 40].

B. Work Discussing Future CPS Challenges

Accompanying surveys on CPS research in application domains, and core concepts are publications discussing the various challenges that can be addressed by taking a CPS research approach. Several of these appeared early on as CPS research was just taking off [41–47], but discussions of such challenges have continued even as CPS has matured [19, 48, 49]. Addressing technological challenges via CPS research has also been a critical topic in several President’s Council of Advisors on Science and Technology (PCAST) reports [50–52]

In this work we contribute a survey of CPS research to the aerospace domain. However, we focus particularly on the unique challenges in the aerospace industry and synthesize CPS research contributions applied to those challenges. We

1NSF CPS Program https://beta.nsf.gov/funding/opportunities/cyber-physical-systems-cps
aim to show how more holistic considerations of complex aerospace systems can help propel the industry toward a more robust, safe, and integrated future, particularly important in an industry in which so many legacy systems remain.

III. Real-time Systems & Control

Real-time systems and control have played an outsized role in the CPS community, starting with its origins in the DARPA and AFRL sponsored “Software Enabled Control” program [53–56]. Much of the contents of the seminal book [53] have gone on to become foci of the CPS community. Naturally, through practitioners and academics, much of this has led to advances in commercial aviation, aerospace defense, as well as the emerging small Unmanned Aircraft System (UAS) community. In this section we divide the impact of CPS research on aviation into two thrusts: 1) the influence of computing, its theory, concepts, and algorithms on aerospace control, and 2) the impact, and intersection of computing and control in aviation.

Influence of Computing on Aerospace Control

High speed computing has enabled many new control paradigms. For example, the X-33 pushed computing and control boundaries significantly, in part because vehicle dynamics changed profoundly across flight conditions. More computationally intensive adaptive control strategies [57, 58], including neural nets (NN) [59] were leveraged to create full-flight envelope controllers avoiding the more traditional gain scheduling which was not feasible due to the explosion of possible operating points. NNs have continued to grow significantly in size and complexity requiring a mechanism to build in safety measures from the start that prevent adverse learning in the case of non-linear/discrete/hard-limit effects such as control saturation [60].

Much of the influence in the CPS community on control has stemmed from the growth of hybrid systems [53, 61], leveraging the ability of automata to model both continuous and discrete dynamics of a complex system [62, 63]. Hybrid systems were, and still are, prominent in many research efforts spanning everything from low-level control [64], to formation flight [65], to air traffic management [66].

Hybrid system modeling paradigms and extensions have expanded rapidly and many extensions have emerged allowing designers to model other cyber and physical phenomena holistically. Simultaneously, modeling tools, themselves a part of the cyber-physical design system, have grown significantly in capability and allowed for rapid design, iteration, model checking, and model-based design [67]. Model-based design [68, 69], wherein models are used to design, verify [70], synthesize [71], and produce executable code [72] for a portion of an aerospace system, ideally provide a direct coupling between implementation and design (see Figure 2). This reduces the opportunity for mismatches between implementation and design [73] and allows for designers to update, modify, and version designs, rather than implementation. The document “Software Considerations in Airborne Systems and Equipment Certification,” DO-178B and DO-178C [74], provide acceptable means through which software may comply with federal regulations, and with which MBD tools (e.g., Matlab, Simulink) comply [21]. While not fully clear how much MBD and synthesis are used in large, real-world projects, NASA famously employed MBD on the Orion project about which they have written extensively [20, 75–78].

The complexity of modern controllers composed of discrete, continuous, and neural net (NN) components require extensive analysis, testing and verification to ensure performance. As a result, formal methods [70] occupy a significant place in the CPS community and have been applied liberally in aerospace [79–81] (discussed more in Section VI). However, MBD, synthesis, verification and similar tools are costly in both time and money to build into production and research processes. For small businesses and research groups in the smaller, less costly aerospace sector (e.g., small UASs), this means these tools have not yet found widespread adoption despite the safety-critical nature of those systems. Indeed, ArduPilot‡, an extremely popular open-source autopilot for autonomous systems, has relatively few formal models and primarily develops autonomous systems directly in software [82].

Aside from the interconnected thread of hybrid systems, synthesis, formal methods, and machine-learning, traditional discrete algorithms have also found extensive use in autonomous aerospace systems including graph search [83], gridding/mapping [84], and automata such as Markov Decision Processes [85, 86].

‡https://ardupilot.org/
**Fig. 3** Classes of Computer Controlled Systems.

**Intersection of Computing and Control** Computer control is ubiquitous and hence so is the struggle of ensuring assumptions about software execution, timing, and computer memory match reality – an exceptionally difficult task. Most relevant to safety-critical aerospace systems is the consideration of timing, and the enforcement of deadlines on hard real-time control tasks necessitating the employment of a real-time operating system (RTOS) [87–89], including assessing the schedulability of computing tasks [90].

While presumably standard practice in the commercial and defense aerospace sector, in the small UAS and satellite sector RTOSs may still be scarce owing to the complexity of writing RTOS software and deployment on inexpensive hardware. Most small autonomous autopilot packages (e.g., ArduPilot, PX4) utilize an RTOS (e.g., ChibiOS§), however, interacting software on companion computers are typically still Linux-based “best effort” implementations. The ubiquitous Robot Operating System¶ is a microcosm of the current state of affairs. While ROS2 employs numerous improvements on the computational side (e.g., QoS, asynchronous messaging) ROS1 is often still the first choice for small businesses and researchers due to its prevalence in the community and the difficulty of switching to ROS2.

Assuming implementation on an RTOS, for the control engineer the difficulties start with selecting a fixed sampling period enabling the controller to be robust to anticipated worst-case scenarios. Over the years exploration of the topic has been applied in both aviation [91, 92] and spacecraft [93], though often sample rate selection remains based on “rules of thumb” [94].

More recently, as chip manufacturers have hit physical limits on clock rates, and advanced algorithms and neural nets have increased computational demands, a renewed effort to consider computation as a scarce resource have also seen a resurgence. On the control side, in direct contrast to time-triggered control, event-triggered and self-triggered control represent the pinnacle of computation utilization – only executing control inputs when needed, as triggered by some condition in the system. Such methods have primarily been applied in the research space to small UAS [95]. Limitations in analytical methods, performance, and perhaps more significantly, implementation, have hampered their wider deployment in safety critical systems. A primary disadvantage is the difficulty of ensuring computational timing guarantees for aperiodic tasks, as well as differentiation between failures and the lack of a triggering event [96].

From the control perspective, one view of the computational control paradigm can be seen in Figure 3 with traditional time-triggered/periodic control systems on the left, and event-triggered/aperiodic control systems on the right [97]. In the middle, Category II, in Figure 3, are the newest computer control paradigms wherein resources, and hence performance, are adjusted online according to the system’s needs and objectives. Figure 4 shows how this type of system might work. Traditional fixed-periodic control systems select a single operating point typically based on worst-case anticipated scenarios (red line). Time-varying periodic control strategies allow the operating point to move along the blue line to adjust performance and resources on the fly.

Co-regulation [98] is an example of this strategy whereby the periodic sampling rate is dynamically adjusted in response to vehicle performance. This requires a new class of controllers and real-time schedulers [99, 100]. Co-regulation in both UAVs [101] and multi-agent UAV systems [102] show similar performance to time-triggered

---

§https://www.chibios.org/dokuwiki/doku.php

¶https://www.ros.org/
controllers, but has significant resource savings that can be put to use where the system needs it. Complementary work in adaptive resource management adjusts computing resources to algorithms that can make use of freed computation cycles [103]. Both of these strategies envision control as a service [104], the quality of which can be adjusted, and anytime algorithms [105, 106] that can be employed to leverage available resources.

**Challenges** An overarching challenge in the CPS control community has been to elevate computation to a first class citizen in the design, testing, and deployment of CPSs. In a modern aerospace vehicle computers physically impact the design of the structure, its power/energy budget, and computationally impact nearly every facet of the onboard controls, avionics, autonomy, and communication. Here we describe two main challenges for the cyber-physical aerospace control community to consider: 1) support for computational and performance adaptability; and 2) safety-critical control software.

To make aerospace systems more adaptable, their computation and performance must also be adaptable. Above, we listed several solutions to developing controllers able to dynamically adjust their performance and computational requirements. On the computing side, while a few supporting adaptive real-time computing task models have been developed, they have primarily targeted the automotive industry, and seen limited real-world deployment [107–109]. Task models that can allocate compute cycles to where they’re most needed are required to allow existing anytime algorithms to leverage resources and produce improved results.

Along these same lines, machine-learning based control is rapidly emerging, and while still not widely deployed in safety-critical software, they likely will be in the future [110–112]. However, often the pursuit in ML-based control has been controllers that demonstrate excellent performance even at the expense of complexity. This limits their deployability on Size, Weight, and Power (SWaP) constrained systems. ML-based components of aerospace systems should be designed in conjunction, and in context with computation and other components of the system to ensure they’re not overdesigned [113]. Additionally, to realize adaptable performance, ML-based perception and control strategies would benefit from design principles used in anytime algorithms since their “best” performance may not be needed at all parts of the flight envelope.

Software has become a critical aspect of the aerospace community in recent years. Most often, research in aerospace control software has focused on mismatches between the intended design and the execution of the software [114, 115], or conversely, sought to verify that software conforms with specifications [116]. Strategies to ensure this have included annotations in the software that correspond with desirable properties, essentially “integrating proofs within the code” [117]. Ultimately, efforts along these lines, like model-based design (discussed above), intrinsically assume that implementation may not conform to design, that decoupling between them will occur, and attempt to rectify it. We take the view that despite advances in MBD or synthesis, unless every aspect of a system can be modeled to the finest detail, control software will interact with handwritten software in some way in the holistic system. This suggests the need to study the control software itself, how it changes [73], how it interacts with other libraries, how bugs propagate [118] and get fixed, how unmodeled aspects of the implementation are changed, likely by hand, to make the system “work.” This means expanding the view of control software to be considered a part of the CPS in its own right [119]. A more complete view of aerospace control software might have established metrics by which we could show how robust a particular software controller is to bug fixes, changes in linked libraries, improper units, or computational exceptions (e.g., segmentation fault).

**IV. Autonomy**

Different definitions and notions of autonomy abound, and some form of automation has been part of aerospace systems for decades. To provide focus and scope in the context of CPS, as well as the relationship and significance of autonomy to other topics of this paper, we leverage a recent definition from [120], which asserts that autonomy involves self-governance and self-directed behavior, relying on “artificial intelligence (AI)-based capabilities that allow it to respond to situations that were not preprogrammed or anticipated in the design”. The key term for this review is
*artificial intelligence*, but because the term itself is fraught with ambiguity and dissension, we will instead focus on a related but arguably more clear term of *machine learning*. Furthermore, to provide scope to our review and analysis, we will borrow from a conceptual framework familiar to autonomous driving, which breaks the autonomy stack into (1) perception, (2) planning, and (3) control [121]. We will therefore review techniques that use machine learning in one of these three areas in the context of aerospace engineering applications. We will then extend the discussion and conclude with several of the unsolved challenges facing the aerospace CPS community, including architectures, safety, and multi-agent systems.

**Perception** Navigation in aerospace has typically been assisted with (human) vision in aviation, inertial measurement units, and ground-based infrastructure such as radar. The rise of neural network-based machine vision, coincident with the increasing number of operational use cases for unpiloted aerial systems (UAVs or UAS), has resulted in a new class of techniques in perception for aerospace CPS. Examples also abound in space exploration [122–124].

Using vision also as a source of redundancy with respect to other navigation and localization aides that are traditional to aerospace (e.g. GPS, radar) is a primary motivator in [125]. Leveraging a notion of synergy and redundancy is also significant in [126, 127], which builds multirotor onboard sensing platform that uses inertial measurement units and altitude sensors in concert with an RGB camera in order to perform a complex UAV landing task without any additional infrastructure. Achieving reliable relative inertial measurements, and vision-based relative pose updates in UAV navigation applications is the focus in a related line of work [128, 129]. Vision is used in a multi-agent setting in [130], where micro-aerial vehicles coordinate their motions in order to track a (ground) person’s trajectory.

In addition to state estimation and navigation, image processing is leveraged in the context of search and rescue or related applications with UAVs [131]. To provide increased situational awareness in marine search and rescue, several approaches use neural network-enabled machine vision for detection of people [132]. Perception is particularly important in the detection and prediction of fires, an increasingly important and relevant UAV application [133, 134].

This discussion has so far focused on the use of neural networks for vision-based perceptions, particularly as this relates to UAV applications. As machine vision and perception performance continues to advance [135–138], these applications will grow but come with the attendant questions about the reliability, trustworthiness, and explainability of these neural network architectures. Furthermore, the notion of perception is not limited to just vision, and more comprehensive and integrative localization and sensing suites are both necessary and are advancing [139–142].

**Planning** The planning literature is rich, vast, and varied, and we again emphasize that we are primarily focused on learning-enabled planning as it applies to aerospace CPS. Reinforcement learning is increasingly being used in the context of planning, including applications to hopping rovers [143], planetary rovers [144], and other spacecraft GNC applications [145].

There is also a growing line of work using supervised learning and neural networks for planning. Trajectory generation and optimization is a focus in [146], where the output of a pre-trained neural network is used as an initializer for a numerical optimization scheme. In [147], a neural network generates motion primitives of trajectories for dynamically-feasible quadrotor trajectories, and recurrent neural networks are seeing increased use for path planning and collision avoidance applications [148, 149].

Much like the perception tasks, machine learning (and in particular, the use of neural network architectures) is seeing a rapid rise in planning applications; and like perception – and any other safety-critical task associated with machine learning – there are still open questions about reliability, trustworthiness, and explainability. As we will discuss momentarily, the power and expressiveness of these learning architectures lend themselves to increasingly complex applications such as multi-agent trajectory planning [150–152] and prediction [153, 154].

**Control** We refer the reader to Section III for a discussion of machine learning-based control but note that much of this work blurs the line in our distinction between perception, planning, and control. For example, several papers investigate closed-loop systems with neural networks in the loop, which span planning and control applications [155–157] but have clear applicability to aerospace CPS.

**Challenges** There is practically an unlimited set of open questions and future research directions when it comes to autonomy broadly, and machine-learning enabled autonomy more specifically. We mention just a few and note that many other directions are related to the other sections of this paper, for example human-autonomy teaming, safety, and of course controls. This review focuses on three challenges. First, what are the appropriate architectures for integrating
As we will discuss momentarily, even state-of-the-art performance levels could have disastrous consequences in safety-critical CPS. However, aviation has achieved one of the great safety records in engineering history not because the individual components of an aircraft (or the air traffic control system) are 100% reliable. Rather, the system is safe in large part due to the architecture; that is, the individual components achieve certain performance characteristics, but it is their interactions and interfaces that lead to the total system behavior. In autonomy, and in particular for machine learning, it is therefore vital to think about how and where a learning-enabled system might fit. Several works have put forth a notion of composition, where individual parts are analyzed and reasoned about, and then with careful attention to interfaces, a total system property can be obtained [158–161]. In addition, there are recent control architectures that explicitly try to separate out traditional mathematical models (e.g., state space control models) and black box learning models in a unified control framework [162–164]. Despite these and other attempts to frame autonomy as a potential question of systems architecture, it is still unclear where and when these learning-enabled systems should be.

Though safety and verification are discussed more thoroughly in Section VI, it is important to think about risk in a slightly different way as it relates to autonomy. If future systems are to be truly endowed with self-governance and high-level decision-making, it is important to assess whether the systems themselves have the ability to reason about risk. In the context of robotics, recent work tries to develop appropriate risk metrics [165] or constrain the learning system itself according to risk [166, 167]. Other work attempts to consider risk at operation time [168]. The question remains: will these systems exhibit the same behavior towards risk as their human counterparts? This question then gives rise to whether they should; though somewhat philosophical in nature, such questions very quickly come to bear on engineering techniques and methods.

Testing has been a vital aspect of assuring aerospace systems for decades. However, many of the tasks being proposed in the autonomy literature have historically been performed by humans. Indeed, humans get “tested”, but this type of testing is both qualitatively and quantitatively different than how we have historically tested software [169, 170]. Furthermore, the way that machine learning is currently tested – in the sense of train / validate / test splits [171, 172] – is likely inadequate for safety- or mission-critical aerospace CPS. For example, an even 96% test accuracy in terms of precision or recall may result in a catastrophic number of accidents or lives lost. This suggests the urgency that machine learning components should be architected within the broader systems context. Recent results suggest different ways of testing neural networks in more rigorous, transparent ways [173–176]. However, much more work is needed, not only in terms of techniques and metrics of testing machine learning algorithms, but also towards an overarching paradigm for test and validation of autonomous systems particularly in the context of aerospace CPS.

V. Humans Working With Cyber-Physical Aerospace Systems

Humans, and their activities, are integral to aerospace CPS in two ways. First, the definitions of automation and autonomy frame the machine as performing functions normally attributed to humans. Thus, the human operator’s activities have been, and continue to be, the central point of definition of operations that CPS contribute to. While machine capabilities have increased over the century since the first demonstration of automatic control of an aircraft, to the point that air transport aircraft now can, in theory, almost completely fly themselves, this automatic functioning is allowed as long as nothing fails or changes in the intended route of flight, at which point the pilot needs to intervene (which is estimated to occur in roughly 90% of flights) [177].

Second, whenever CPS systems are assigned functions, the human operator is impacted. While no longer needing to execute some functions, the human operator still needs to perform any that haven’t been automated. This leads to several concerns [178]. For example, while a higher-level of automation suggests that the machine is taking on more importance in system functioning, the human may consistently override or second-guess high levels of automation; this both lowers the actual operational value of the automation, and can be intensive for the human in terms of effort and information required. Likewise, so-called lower levels of automation that alert or suggests decision may effectively dictate a course of action to the human [179, 180], driving the system behavior even if the human is the actuator.

Function allocations driven by what the machine can do implicitly apply a function allocation strategy termed by Bailey as leftover allocation: automate as many functions as technology will permit, and assume the human will pick up which-ever functions are leftover [181]. In aerospace, commonly automated functions drive standard processes within clearly-defined conditions. However, this provides little support in off-nominal conditions, which is generally when
the human needs the most support [182]. In such cases, the number of required human operators is no longer driven by workload in nominal conditions; instead, following the description of modern pilotage as “hours/weeks/months of boredom interrupted by seconds of sheer terror,” the number of human operators required (somewhere, connected to the vehicle somehow) is driven by the worst-case, when they must be able to detect and take-over situations beyond the machine’s capabilities.

Historically, humans have been framed as being better at cognitive tasks, and machines at the higher-bandwidth sensing-control-actuation type tasks, based on early attempts to frame what “Men are better at / machines are better at” [183]. This has been an explicit or implicit justification for a supervisory control paradigm, in which the human is expected to perform fairly cognitive tasks based on abstraction and problem solving, framed by Sheridan as principally serving as inputs to the computer/machine to determine its control activity [184]. This substantially changes the human’s task from “needing to know the vehicle dynamics well enough to control them” – a skill-based task easily transferable between aircraft – to “needing to know the details of how the automation/autonomy works sufficient to program, command, confirm and intervene in its activities” – which, outside normal operations, can be an intensive task requiring knowledge specific to that piece of autonomy and not transferable to other systems or situations.

Typically, discussions of function allocation only consider which agent within a human-machine team will execute which action – here, defined as the authority for an action. However, the human remains legally responsible for the outcome. Thus, as automation and autonomy take on more, the human’s role in a supervisory control paradigm is arguably harder. The human’s tasks are “monitor” and “intervene,” even as s/he has been pushed out of the sensing-control-actuation performance of the task. This demands that the operator do something that may look easy and passive given its lack of externally-observable activity: analyze whether the machine is doing the right activity by monitoring. This monitoring needs to occur frequently enough that it can catch any problem within the performance of the task quickly enough for successful intervention – for continuous control tasks such as autoflight, this monitoring needs to occur almost as continuously. This monitoring may need to consider all the inputs used by the machine as well as its outputs – the monitoring may trigger on simple determinations of machine actions that look to be out of range, but it may also need to perform a parallel assessment of what the operator thinks the actions should be, or even reverse-engineer why the machine outputs are being executed and decide whether that implied machine process model is correct. Only recently are studies starting to analyze the demands of this monitoring, and the effort, information and knowledge it requires [185–187].

Thus, in aerospace the decision of whether to engage and use the machine should not be viewed merely as a matter of “trusting” it: with aerospace’s high standards for safety, a system that is correct 90% of the time, or 99% of the time, needs to be monitored and corrected for. Instead, human operators need to “trust but verify” at all times. Without a concrete basis for assessing if the automation is correct, humans often over- and under-trust the automation; either way, incorrect trust is viewed as human error, despite its basis in the function allocation [188]. Further, a human’s operator choice to rely on the machine also considers the “interaction overhead” associated with programming it, invoking it and, in when thinking ahead to potential interventions, the difficulty and delay in intervening. Thus, a human may invoke a system even when they do not believe in its functioning if it is easy to command, monitor and intervene; conversely, a human operator may not invoke a system that they believe is 99% correct if it is hard to command, monitor and intervene; conversely, a human operator may not invoke a system that they believe is 99% correct if it is hard to command and the difficulty and delay in intervening can be safety-critical.

It is common now to describe intelligent machines, such as those sought by CPS, as members of human-machine teams, an extension from the supervisory control one-human/one-machine model. However, where autonomy is viewed as a team member, its ability to communicate with others, in a form comprehensible to them, is of significant interest. This transcends the autonomy being “explainable” in a passive sense by an active human operator: It is important that team members are able to anticipate each other’s information needs and provide information at useful, non-interruptive times [189, 190]. Timing must be considered as well: inter-leaved tasks can leave one agent waiting on another, and poorly timed communication can disrupt performance. An interruption may be demanded by circumstances, in which case it can spur knowledge acquisition [191] and facilitate decision-making performance [192]. Unfortunately, too often automation is “clumsy:” it unduly interrupts its human team members because, whereas humans can implicitly sense information about whether other team members would benefit from an interruption, automation historically cannot [193], for an example of efforts to improve this aspect of automation). With human-automated teams, predefined sets of function allocations may serve as more explicit coordination strategies, such as the playbook proposed by Miller and Parasuraman [194] and such as the function allocations that the pilot of a modern aircraft may invoke. Likewise, roles for the autonomy such as critiquing systems that monitor the operation and only interrupt if the human operator’s activities or judgements do not match models of “good performance” can define effective machine roles [195].

Finally, the literature in cognitive engineering also examines the meta-cognitive activities of humans in safety-critical
systems such as aerospace. These operations require the choice of which activities to perform when, balancing many simultaneous actions. Dynamic analysis is often required to identify situations where the interleaving of functions assigned to disparate agents requires significant co-ordination or idling as one waits on another, or where workload may accumulate, or where one agent will be unduly interrupting another.

Proposed notions of resilience describe a team’s ability to work in unexpected situations as maintaining control of, or regulating those situations [196–198]. Resilience enables organizations to cope with the unexpected in a “robust yet flexible” manner [199]. However, the brittleness of machine functions (i.e., safe operation only within some defined set of operating conditions defined, for example, by the bounds of the control authority or the training set used in machine learning) reflects how allocation of key functions to the machine may degrade the team’s resilience in unexpected situations. Likewise, resilience is fostered when a human agent may select strategies (courses of action) appropriate to the state of the environment and their own capabilities [200]. For example, Hollnagel’s concept of cognitive control describes how humans adapt their activities (and sequence them) in response to their competency and their perception of resources available to them (such as information availability) and demands on them (such as subjective available time) [201, 202]. However, such adaptation can be constrained where the machine in a human-machine team dictates a specific sequence of activities from the human [203]. The adverse effects of such overly prescribed function allocations have been found to manifest in work-arounds or dis-use of machines [188, 200, 204].

These difficulties highlight the value of understanding the complete set of human activities in aerospace operations when designing CPS. Such understanding can serve several purposes. First, it can inform the design of important machine functions to better imitate what the human does. Second, it can highlight which machine functions would truly support the operations, versus which would potentially disrupt. Third, it highlights the need for machine behaviors that not only act independently – autonomously – but also interact with their team, including a human supervisor who is legally responsible and other agents outside their immediate vehicle, such as air traffic control or command and control.

**Challenges**  
Several important questions must be answered when fielding CPS into quintessentially human-driven operations – and CPS research and developments needs to be steered to ensure these answers are available. The first recognizes that CPS are envisioned to take over functions (or type of functions) normally attributed to humans: So, what-all do the humans actually do when performing these functions? This section has articulated a wide-range of activities normally attributed to humans. Some, commonly called the “taskwork,” are obvious and clear-cut, but others are typically described as implicit or intangible: the meta-cognitive activities of constantly predicting and organizing the entire set of activity that is occurring, of re-designing the team allocation, and of preparing for likely future activities.

Second, no agent truly gets to act “autonomously” in aerospace operations: all must coordinate their activities with others, sometimes loosely and sometimes tightly integrating their activities with others. This section has articulated a wide range of teamwork activities. Unfortunately, machines are too often not good team players; CPS systems must not only understand how to perform their functions, but also how to coordinate these functions with the other agents’ activities, and how (and when) to communicate about them in a manner useful to others in the operation who should not need to learn inner details of the CPS to understand its impact on the operation.

Finally, the choice to give CPS certain functions impacts others, who must adjust their pattern of activity around them. CPS should not be developed just around which functions it is easy or convenient for the technology designers, but instead need to be purposefully be constructed towards a role that supports the entire team. This criteria is particularly profound when a human operator is responsible for the outcome of the entire operation. In this case, it is advisable to the human to not employ CPS whose functions may not be appropriate to the situation, or where the difficulty or risk in commanding, programming and invoking the CPS – and then needing to monitor and potentially correct or intervene the functions of the CPS – are not worth the effort and disruption.

**VI. Safety & Verification**

Verification of CPS requires an integrated approach that weaves together a safety case from both traditional and formal verification techniques. The U.S. FAA, for example, documents verification requirements for flight certification in DO-178B [205], DO-178C [206], DO-333 [207], and DO-254 [208]. Intuitively, we need to validate that our system design meets the objectives of the given operational concept, and validate the set of system requirements, e.g., so that we know they are internally consistent, faithfully represent properties like safety and security, and sufficiently cover the range of system behaviors. We then need to use the validated system design and requirements in our effort to verify that the system does what we think it should do and nothing else. For example, we can verify that a system sufficiently often produces an acceptable output given certain inputs using traditional verification techniques like testing and simulation.
For another example, we can verify that our system does not spontaneously exhibit an unsafe behavior (i.e., exhibit the behavior on any input or no input at all) using formal verification techniques like model checking and theorem proving. Our objective in the end is to produce a cohesive argument or safety case for flight certification.

**Highlights of Formal Methods in Autonomous Aerospace Systems** Though they are still not as broadly understood or widely adopted as traditional verification techniques, formal methods have significantly impacted all areas of autonomous aerospace systems. Often, specific challenges from the development of new aerospace systems have served as the impetus for new areas of formal methods research and development. We highlight a few examples of advancements in formal methods driven by developments in aerospace autonomy.

*Model Checking* contributed to the early-design-stage of NASA Ames’ Automated Airspace Concept [209], including finding rare scenarios where aircraft could execute commands that were not from their current controller [210], and compare multiple design options to prove which design parameters provide the lowest probability of an unresolved conflict between aircraft [211]. The impact of formal analysis of one, initial, partial design led to the challenge of formal design space exploration: comparing several possible system designs that differ in terms of design choices, capabilities, and implementations. We can apply formal techniques ranging from model checking to model-based fault-tree analysis to rigorously analyze the safety of different design solutions, comparing how different functional allocations impact the overall reliability of the automated air traffic control system [212]. Next, we faced the challenge of considering a design space with more than 20,000 designs for the NextGen air traffic control system. We introduced a compositional, modular, parameterized approach combining model checking with contract-based design to automatically generate large numbers of models from a possible set of components and their implementations, identifying novel as well as known problematic configurations [213]. The need to scale model checking to explore large design spaces faster drove the creation of a new algorithm, FuseIC3, learning and reusing information from solving related models to check the design space of 20K models over five times faster than before [214]. Utilizing logical techniques to discover design-space dependencies and dynamically order intermediate analysis steps sped up the verification of NASA’s full design space by another 9.4x [215]. We have continued advancing algorithms for scalable design-space model checking, with impact both on automated air traffic control and on industrial CPS analysis, such as at IBM [216].

*Runtime Verification* (RV) can efficiently analyze the current mission, checking adherence to mission safety parameters in real time, on-board, in order to account for, e.g., safety violations caused by unexpected environmental occurrences or other off-nominal conditions. Like simulation, RV analyzes single runs of the system instead of the entire design space, but RV uses formal verification techniques to provide real-time checks of complex-temporal properties while the system is operating [218]. The need to assure automated flights of NASA’s wildfire surveillance aircraft, the Swift UAS drove the creation of the Realizable, Responsive, Unobtrusive Unit, an RV engine designed to satisfy those verification parameters [219, 220]. Unexpected emergent behavior by Robonaut2 on the International Space Station inspired a novel revision of RV via R2U2 in order to disambiguate between failure modes using only the space left over in the FPGA controlling Robonaut2’s knee joint [221]. After an extensive survey of all currently-available verification tools, NASA’s Lunar Gateway Vehicle System Manager (VSM) team selected R2U2 for operational verification [222]. The need to improve usability and trust in this real-time verification of a large-scale, complex safety-critical system drove advancements including easy-to-validate methods for set-based reasoning, transparent representations of assume-guarantee-contract requirements, and automated checks to answer the question of how do we check the checker [223].

**Fig. 5** The (growing) tree of formal methods; solid lines represent direct variations whereas dashed lines represent related derivations [217].

**Challenges** While traditional verification techniques benefit from broad understanding and adoption in the final stages of system design, CPS require a more integrated assurance approach incorporating both explicit validation and verification, both traditional and formal verification techniques, and application of assurance techniques across all stages of the system life-cycle. This need lays the groundwork for future challenges in safety and verification of autonomous...
aerospace systems. Recently, [217] defined a framework for certification of future reliable autonomous systems; we highlight some elements of that framework along with other aerospace-specific challenges here.

One major barrier to adoption of integrated CPS V&V techniques stems from the success of the "V-model" of system engineering. This "V-model" lays out the recommended stages of system design in the shape of the letter "V" with verification appearing exclusively as the last step of the design, and consisting only of traditional techniques. While this model may be ideal for the design of physical systems, where verification does not require formal methods and where testing does not make sense until there is a completed structure capable of responding to a test, it does not apply to software. As aerospace systems evolved into autonomous cyber-physical systems, the "V-model" no longer applies. Therefore, we need to address the challenge of social barriers to adoption of state-of-the-art CPS V&V techniques at every stage of the system lifecycle, from early system design to system runtime and maintenance. This challenge involves broad, industry-wide adoption of techniques including initial requirements validation, using model checking to check early-design-stage partial (not yet complete) designs for safety properties, generating software implementations from formally-verified designs, and incorporating on-board runtime verification for real-time checks that the current mission upholds safety requirements.

Technical challenges to safety and verification of autonomous aerospace systems include developing new formal methods to address verification needs of new types of (sub-)systems, scaling existing techniques to be capable of reasoning about ever-more-complex CPS, and automating the application of verification tools and techniques to make their use feasible. For example the Achilles heel of model checking is the state-space explosion problem: because model checking exhaustively checks all possible system behaviors to provide a provable assurance claim, analyzing a system with a very large behavior space can take an unacceptable amount of time, or elude the model checker entirely. New algorithms, heuristics, compositional verification techniques, and other technological advances can address this challenge; these are all active areas of research. For another example, applying rigorous verification techniques early in the design lifecycle requires significant time and expertise, but this investment can massively benefit later design stages if we design automation to enable this. Early design stage formal verification can additionally contribute to automated test-case generation and automated generation of verified software implementations, thus saving the system designers from performing these tasks manually while providing a higher level of assurance.

There is also a set of challenges that bridges the gap between social and technical challenges. Human system designers, engineers, and certification authorities need to be able to understand and trust formal methods: how do we both train engineers to trust the outcomes of formal verification and grow formal verification tools to transparently explain the reasoning behind their proofs? Explainability of formal methods remains an active research area, as do the related areas of automated certificate generation for model checking, and automated validation of formal specifications of system requirements. There is also a large community researching educational techniques, such as how to train engineers to think formally, design with a verification mindset, and deeply understand what different V&V techniques prove, and don't prove about the system under test [224]. This is particularly challenging in the context of aerospace engineering [225].

Building off of these social and technical challenges as well as the discussion of humans in aerospace CPS (section V), we must not only ask where does the machine fit in safety-critical operations. We must also ask how to then assure the safety of such human-autonomy teams? In addition to formal verification, we need more advanced hazard analysis capabilities and system safety methods. One such approach, called Systems-theoretic Process Analysis, or STPA [226], has shown promise in aerospace CPS applications because it can help assess unsafe decision-making as it relates to system state, and feedback about that system state. Other work from the system safety community explicitly analyzes the coordination (or lack thereof) between multiple agents within a system [227–229]. Pointing to the earlier discussion of the V-model and the historically late introduction of verification, these approaches provide a way to identify system architectures from a safety-driven perspective, particularly when the ‘architectural elements’ have decision-making or autonomous capability.

**VII. Conclusion**

The Cyber-Physical Systems community and its outgrowths, though relatively new, have already had significant impact on the aerospace community. CPS has sought to create a science of systems [230] and have addressed many facets of modern complex vehicles, most prominently, real-time systems and control, humans in CPS, autonomy, and safety, validation, and verification. Here we have focused on synthesizing the impact of the CPS research community on those key areas and identified areas for future growth and exploration.

Aerospace vehicles are becoming more complex, incorporating machine-learned algorithms, and navigating an
increasingly crowded and uncertain national airspace. To meet the challenge of deploying safe, secure, reliable aerospace vehicles the community will need to continue to embrace new ideas from related areas and incorporate the best ideas into our community.

Acknowledgments
This work supported in part by NSF-2047971 and NASA 80NSSC19K1702.

References


