Addressing Autonomy in Conceptual Design

Robert Morris, Anjan Chakrabarty

■ *Aircraft design is an iterative process of creating a design concept from a set of requirements. Conceptual design is an early phase in the process, during which preliminary decisions and trade studies are made from a set of requirements related to mission objective and costs. Although much attention has been paid to applying autonomy technologies to robotic systems, including air vehicles, there has been little attention paid to incorporating autonomy as part of the conceptual design process. Consequently, designing for autonomy tends to be retrofitted to a vehicle that has already gone through a complete design process rather than as part of the initial process. This derivative approach to designing autonomous systems is suboptimal, and there is evidence that this has hindered the acceptance of autonomy technologies. This article proposes an approach to conceptual design for aircraft that incorporates autonomy into the conceptual design process. To illustrate the principles introduced, we consider the example of configuring an autonomous small unmanned aerial vehicle for searching and tracking a target of interest.*

Aircraft design is the iterative process of creating a detailed design concept from a set of requirements. Traditionally, conceptual design is an early phase in the process, during which preliminary decisions and trade studies are made about vehicle aerodynamics, structures, propulsion, stability, and control from a set of requirements related to mission and costs.

Recent advances in autonomy for complex robotic systems pose potentially significant changes in the design process. Machine autonomy is realized through hardware and AI software for sensing, communication, control, and navigation. Autonomy has been proposed for a wide range of robotic systems, from Mars rovers (Wettergreen et al. 2014; Jonsson, Morris, and Pedersen 2007) to self-driving cars (Bayouth, Nourbakhsh, and Thorpe 1997).

ISSN 0738-4602

With the advent of unmanned aerial vehicles (UAVs), nontraditional aspects of design such as size, weight, and power requirements for sensing systems have been considered (Jerath and Langelaan 2016). The movement toward autonomous UAVs adds further considerations of processing and memory requirements of onboard computers. To date, the ramifications of autonomy for design have not been fully integrated into practice. For example, in the most recent edition of the standard textbook on aircraft design (Raymer 2012), no mention of design for autonomy is to be found. As a consequence, autonomy tends to be something that is added to a fully designed system. Retrofitting something as potentially all-encompassing as autonomy into an existing design is clearly suboptimal. In addition, evidence suggests that early attempts at integration of autonomy into complex systems often failed to be accepted because autonomy was cobbled together as an afterthought in the design of the system (Murphy and Shields 2012). Instead, autonomy must be part of a rigorous conceptual design process as a condition for acceptance.

Adding considerations of autonomy to conceptual design will also address the so-called substitution myth (Christoffersen and Woods 2002), that is, the assumption that machine autonomy can be integrated into a system by a simple act of substitution, replacing manual operation with automation in a fully designed system. As noted by Johnson et al. (2013), substitution incurs a negative impact in the overall coordination of human and machine. Such coordination requirements, as we discuss below, are defined during conceptual design.

This article draws on previous work in the cognitive components of autonomy [for example, that of Ingrand and Ghallab (2014)], on architectures for autonomy [for example, that of Kortenkamp and Simmons (2008)], and on so-called hybrid human-machine autonomy [for example, that of Johnson et al. (2013)]. This article expands previous work by investigating traditional principles of conceptual design for aeronautics and shows in detail how machine autonomy can be integrated into the design process. Then we expand the scope of previous efforts by introducing a distinction between vehicle autonomy and operational autonomy, arguing for the primacy of the latter notion in any discussions of design for autonomy. Although the focus here is on aeronautical vehicles, the overall results can be applied to virtually any kind of robotic platform for which autonomy technology can be applied.

In the next section we review traditional concepts in design for aerial vehicles, with a focus on the conceptual design phase. Then we review the architectural components of autonomy.

In the section on conceptual design for autonomy, we propose a set of principles for integrating autonomy technology into conceptual design, then follow with a case study that illustrates the ideas presented with a conceptual design for a small UAV (sUAV) search and track mission.

A Review of Conceptual Design for Aeronautics

This section draws extensively on a standard and popular textbook on design for aeronautics (Raymer 2012). Aircraft design is divided into three phases: conceptual design, preliminary design, and detailed design. In conceptual design, requirements that drive the design are transformed into a wide range of aircraft configuration concepts, and trade studies are made to find those that are viable. In preliminary design, one configuration is chosen, and a more detailed design of major components is made, in addition to cost assessments. Finally, in the detailed design phase, the parts of the actual system and the fabrication and tooling processes are designed.

This article focuses exclusively on the conceptual design phase. The reason for this restriction is not because design for autonomy does not factor importantly into the other phases. Rather, the restriction limits the scope of the discussion to identifying general principles for integrating autonomous systems into the conceptual design phase.

Aircraft conceptual design asks questions like the following (Raymer 2012): (1) What requirements drive the design? (2) What should the vehicle look like and cost? (3) What trade-offs should be considered in determining optimal designs? (4) What technologies should be used (current or future state of the art)? (5) Do these requirements produce a viable and salable aircraft?

Types of requirements for conceptual design include a high-level definition of the purpose and operations of the aircraft, including the time line for development and the level of acceptable technological risk; mission- or customer-centric requirements such as size, weight, range, endurance, altitude, speed payload, and cost, and other specific requirements like low observability; equipment and technologies (including computer hardware or software) required to carry out mission; legal requirements for performance, design, and operations; and company-based best practices requirements.

The conceptual design effort typically begins with a conceptual sketch, a rough drawing used to estimate aerodynamics and weight requirements (a process called sizing). The sketch is analyzed against the mission requirements to determine feasibility of the design, and optimization techniques are used to find the cheapest or lightest aircraft that adhere to the performance and mission requirements. The results of the analysis and optimization provide inputs to a revised design that more clearly determine size and weight (and other) constraints on the aircraft. After possibly many iterations, the final design sketch passes to the preliminary design process for further development.

Architectures for Autonomy

By (machine) autonomy, we mean a collection of cognitive capabilities that combine to enable a system (sometimes referred to as an autonomous agent) that exhibits deliberative behaviors (Ingrand and Ghallab 2014). More specifically, this means that at a specified level of abstraction determined by system requirements, the vehicle is able to behave with goals in mind. Generally speaking, goal-based rationality requirements can be classified along a spectrum from the purely reactive (the system is able to respond in real time to avoid hazards) to the purely deliberative (the system is able to choose mission goals and perform actions that accomplish these goals). Between these extremes is a collection of tactical capabilities that allow the system to observe and act on the world in ways that contribute to accomplishing goals or remaining safe.

Cognitive capabilities for autonomy include mission and trajectory planning, observing events and features of the world relevant to accomplishing goals, learning and adapting to changes to the operational environment, and refining planned actions into triggers of motor forces (Ingrand and Ghallab 2014). Autonomy capabilities are implemented by hardware and software that interface with a machine platform. Autonomy software implements AI models and algorithms for predicting, understanding, and acting on the world.

Autonomy Architectures

Conceptual design for autonomy requires discussion of how the cognitive components of autonomy fit together into an integrated system. In this discussion we distinguish between a vehicle architecture and a broader notion of an operational architecture. The components of an operational architecture include the vehicle components and organization but also may include remote processors that reside on other vehicles or a ground system, as well as the components that enable human inputs to the system (figure 1). For aeronautical applications especially, vehicle architectures derive from operational architectures, and therefore the latter take precedence in design.

The primacy of operational architecture is evident in any reasonable definition of classes of autonomy, such as the one in the paper by Shladover (2016) for self-driving automobiles. Even at the highest (level 5) class of autonomy (the car behaves like a true chauffeur, "retain[ing] full vehicle control, need[ing] no human backup and drive[ing] in all conditions"), the human is still present and is controlling operations, such as by setting navigation goals.

In the following, we distinguish among three issues: defining the structure and style of an architecture, determining the distribution of capabilities, and designing for human-machine coordination.

Architecture Structure and Style

It has been often claimed that autonomous systems consist of layers (Alami et al. 1998; Coste-Maniere and Simmons 2000; Bayouth, Nourbakhsh, and Thorpe 1997). This means that high-level functionality can be recursively broken down into functionally simpler subsystems. Many architectures distinguish among three tiers: a deliberative layer, an executive layer, and a control layer (terms defining these layers may vary). Figure 2 shows the placement of these generic layers, as well as the hardware layer for the controlled vehicle. The deliberative layer is responsible for mission planning, including planning path trajectories and other actions. It is also responsible for building and maintaining high-level models of the world and the vehicle. The executive layer acts as a bridge between the deliberative layer and low-level control. Some of its capabilities include activating low-level behaviors, failure recognition, and triggering replanning activities. Finally, the control layer comprises the set of low-level behaviors and controllers that directly interface with hardware.

The integration of autonomy components requires design of the means by which components communicate with one another, sometimes referred to as the architectural style. Examples of infrastructure for communication are client-server and publish-subscribe. Recently, more developers have been taking advantage of the reliability, efficiency, and ease of use that externally available communication packages such as the Robotic Operating System (ROS) (Quigley et al. 2009) provide.

Distributing Cognitive Capabilities

Operational architectures are inherently distributed. We look at two types of distributed operational architectures: those that support the activities of a single agent and those that support coordination of multiple agents. In this section we are agnostic as to whether the capabilities are automated or performed by a human. In the next part we discuss human-machine distributed capabilities.

The first kind of distribution is a spatial distribution that can arise due to restrictions on the size, weight, or operational constraints of the vehicle. For example, due to size and weight restrictions of sUAVs, it is common to assign more computationally intensive or memory-intensive processing to a ground processor rather than the vehicle. An important factor in this kind of distribution of capabilities is the possible communication overhead incurred from ground to vehicle, as well as its effects on the responsiveness of the vehicle.

The second kind of distribution of capabilities arise when it makes sense to have coordinated activity among many agents to achieve mission goals. Some missions require solutions that involve autonomous agents operating as teams. In addition to the sensing, navigation, and control of capabilities of individual vehicles, additional sensing, communication, and planning capabilities are required to enable coordination. We may also distinguish between the middleware for coordination and algorithms for enabling coordination, such as team formation and distributed path planning. An example of the former is the use of a cloud-sourced database management system to enable communication of data and commands among networks of sUAVs (Tyagi and Nanda 2016).

Human-Centered Autonomy

Operational architectures for autonomy are inherently coordinated between human and machine. As

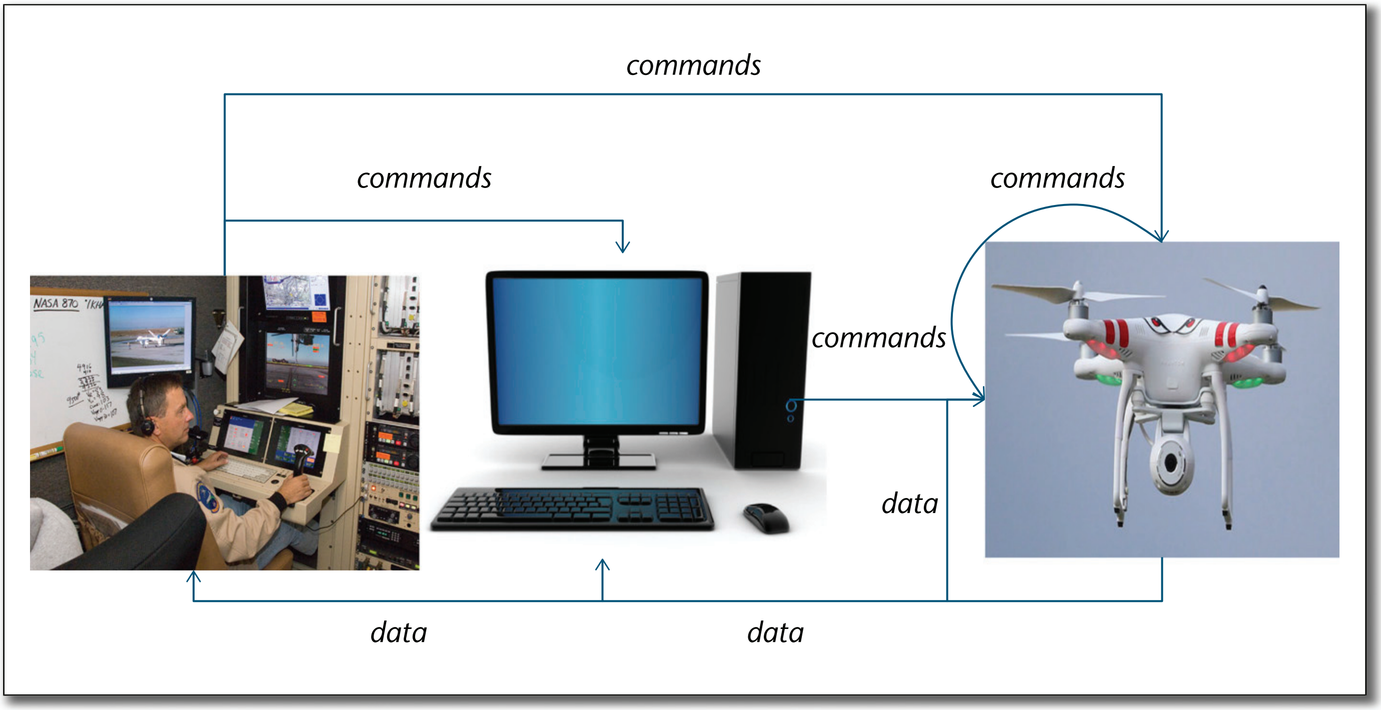


Figure 1. Schematic of an Operational Architecture.

Command/data loops within a system that integrates human decision making with vehicle autonomy.

noted earlier, coordination activity is joint activity, in the sense that what one agent does depends at least partially on what others are doing. Joint human-machine activity is relevant to the theme in this article because "in sophisticated human-agent systems, the underlying interdependence of joint activity is the critical design feature" (Johnson et al. 2013).

Joint activity poses new requirements for sensing, planning, communication, and other cognitive capabilities that are based on coordination to manage the interdependencies among human and machine.

Metrics for Autonomy

To evaluate and compare designs for autonomy within the conceptual design process, it is necessary to identify a set of quantifiable performance metrics. First, autonomy is a set of capabilities for exhibiting goal-directed behavior. Therefore, we define an *effectiveness metric,* namely, the degree to which the system exhibits goal-directed rationality at a required level of responsiveness. By responsiveness we simply mean that it does not take too long to accomplish its goals.

The second metric often associated with autonomous systems is *robustness*: the system is effective in a wide range of operating conditions. This does not necessarily require the system to exhibit goal-directed rationality on its own in all operating conditions; for example, under certain conditions, if it simply requests assistance from a human operator, that decision is sufficient to exhibit robustness.

The final metric is safety. Within the range of behaviors derived from mission requirements, the vehicle will not deliberately or accidentally harm itself, humans, or its environment.

More metrics for autonomous system performance can be proposed, but these three are sufficiently comprehensive for us to confidently use them in the following discussion.

Conceptual Design for Autonomy

In this section we combine the ideas of the previous two sections to provide a high-level description of the role of autonomy in conceptual design. More specifically, we focus on a design from a scratch problem, rather that what Raymer (2012) called a derivative design problem, one in which an existing detailed aircraft is modified to be autonomous. As the purpose of this article is to incorporate considerations of autonomy at the conceptual design phase, derivative design, although potentially important to study in the context of autonomy, is not an issue to be discussed here.

First, the high-level purpose of the mission gives rise to cognitive capability requirements for a vehicle. For example, consider a mission in which an sUAV is required to find small fires within an abandoned building. A core requirement for this application is that the vehicle can navigate in a small indoor space. Another requirement is that the vehicle can sense in a cluttered, smoke-filled area with no GPS. Constraints induced by these requirements can be classified into must-have capabilities (for example, the vehicle must contain a sensor that can detect fires) and can't-have capabilities (for example, a GPS and localization based on GPS, or vision-based navigation, which won't work in dark spaces). Furthermore, constraints stemming from requirements propagate into other constraints. For example, the requirement of navigating through cluttered spaces could rule out some classes of sUAV platforms because of size. The requirement for smaller platforms in turn limits the sensing and processing that can fit onboard.

Second, operational requirements induce constraints on the set of viable operational architectures. These may include constraints on roles humans play in the coordination of human and machine. In addition, operational requirements may induce constraints on the vehicle autonomy architecture.

Third, mission- or customer-centric requirements related to sizing could affect the degree of autonomy allowed. Especially for small aircraft, such as sUAVs, the equipment used to achieve high levels of automation may scale disproportionately and actually begin to drive size and performance metrics. One option for achieving autonomy on sUAVs is to distribute the cognitive capabilities between onboard and ground processes. This would increase the complexity of the design by requiring the addition of communication overhead between ground and vehicle; however, the need for a cognitive capability might favor a distributed design, as long as the required responsiveness of the system is maintained despite the communication overhead.

Fourth, considerations of equipment and technologies needed for a mission raise questions about the intended role of autonomy. For example, at NASA it has been common to classify machine autonomy as enabling or enhancing a mission. Autonomy is enabling if mission goals cannot be accomplished without it; autonomy is enhancing if it offers a better (safer, more effective, more robust) alternative to purely manual operations. Put another way, an enabling capability is usually one for which manual operations are impossible, too dangerous, or too difficult. Similarly, to be determined as enhancing often means that some machine capability improves the human operator's cognitive capabilities or offers a more robust, effective, and safe alternative to manual operations. As a special case, a mission might have a built-in requirement to test new technologies (for example, the remote agent experiment on NASA's Deep Space One mission).[1] In such special cases, the autonomy clearly becomes enabling for the vehicle.

Fifth, legal restrictions could affect all aspects of autonomy design. Anticipating the focus of the next section by way of illustration, current FAA rules on the operation of sUAVs restrict both human and autonomous operations and, relatedly, provide constraints on both vehicle and operational architectures.

Finally, as noted earlier, design requirements often arise from company best practices. Autonomy is a new suite of technologies, and therefore there might not be a history of best practices associated with autonomous design. Tying autonomy development to best practices is the best way to achieve acceptance (Bayouth, Nourbakhsh, and Thorpe 1997). One way to accomplish this is through a design for preplanned product improvement (Raymer 2012), a configuration that allows for the evolution of autonomy capabilities over time. We see examples of preplanned product improvement (P3I) extensively in defense industries and in automobile development.

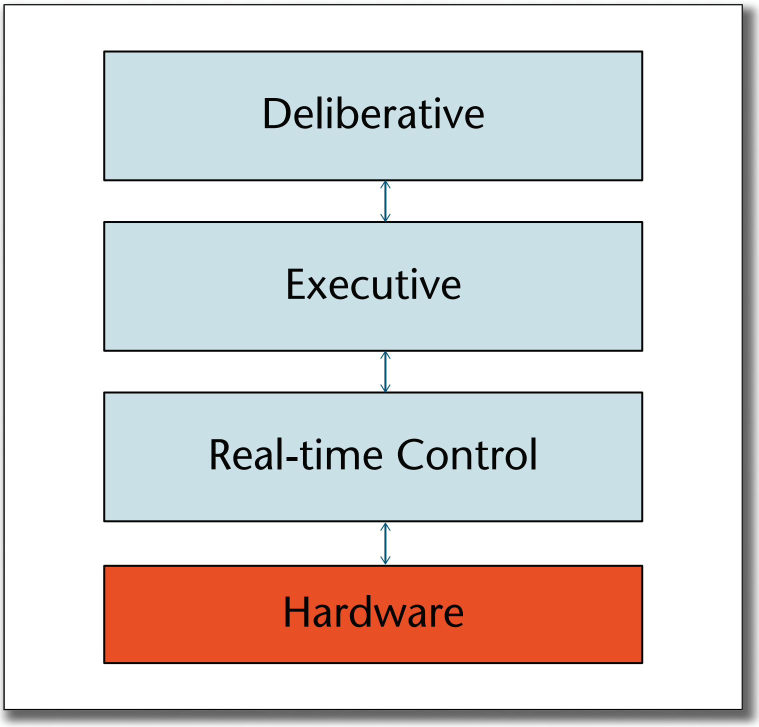


Figure 2. Generic Three-Tier Architecture for Autonomy.

Following the outline of the conceptual design process, once a viable operational architecture for autonomy has been selected from the set of requirements, the next step is to conduct trade studies to determine the best equipment (hardware and software) for implementing the component. A performance study would decide whether a candidate component has the desired responsiveness to inputs or whether a component exhibits the desired resolution. Another class of study is more related to sizing constraints — does the proposed component add too much weight or fit properly into the vehicle? Do the processing requirements make it impractical for the capability to be onboard? A third class of trade study involves development cost: for example, should the software required for the capability be developed in-house, or should an open source version of the capability be considered?

To summarize, conceptual design for autonomy consists of deriving component cognitive capabilities from mission and operational requirements. For each capability, a component analysis determines whether machine autonomy is enabling or enhancing. Component capabilities are combined into an architecture. Architectural issues include identifying a communication infrastructure to tie the components together;

Requirement	Relevance to Autonomous Design
Mission purpose	Cognitive components for accomplishing mission goals and for safe operations
Operations	Distributed architecture/human-machine architecture
Mission/customer-centric	Constraints on operational architecture
Equipment and technologies	Enabling versus enhancing technologies
Legalistic	Human-machine architecture
Best practices	P^3 *I* for integration of autonomy technologies

Table 1. Summary of Requirements for Conceptual Design and Relevance to Design for Autonomy.

determining the distribution of capabilities onboard and on remote processors, as well as determining the coordination of human and automation; and determining an autonomy deployment path that respects best practices and ensures acceptance.

The results of the discussion in this section are summarized in table 1. To complement and provide detail to this discussion, we next describe a real-world example.

Case Study: Autonomy for sUAVs

In this section, we illustrate the conceptual design process for autonomy with an example using sUAVs, which have been proposed for a variety of commercial (industrial and agricultural) and military applications, including border interdiction, search and rescue, wildfire suppression, communications relay, law enforcement, disaster and emergency management, and three-dimensional archaeological map reconstruction.

A recent body of work has emerged in which autonomy capabilities have been proposed for sUAVs, both fixed-wing and rotary wing. Designing autonomy for sUAVs is a different problem than designing for larger platforms. More specifically, there is an important trade-off between desired degrees of autonomy and the performance of the vehicle as captured by its size, weight, and power requirements. Additionally, sUAV platforms are typically cheap, modular, and easily reconfigurable. This makes it possible to apply them to highly specialized applications.

A number of technical challenges make developing hardware and software systems for sUAVs more difficult than for ground robots (Bachrach et al. 2011). Limited payload. This reduces the computational power available onboard and sometimes precludes the use of high-fidelity sensors.

Noisy position estimates. While sUAVs will generally have an inertial measurement unit (IMU), double-integrating acceleration measurements from lightweight microelectromechanical systems IMUs results in large position errors.

Fast dynamics. sUAVs have fast and unstable dynamics, which result in a host of sensing, estimation, control, and planning challenges.

Constant motion. Unlike ground vehicles, an sUAV cannot simply stop and perform more sensing or computation when its state estimates have large uncertainties.

Planning in a three-dimensional representation of an environment. A three-dimensional configuration space in general makes path planning more computationally intensive.

Some design parameters that drive requirements for sUAVs include whether the sUAV will be flown in an indoor or an outdoor environment; whether daytime or nighttime navigation is required, whether the operating environment is GPS available or GPS denied, whether the environment is well mapped or unknown, whether the environment is cluttered or uncluttered, and whether the environment is confined or open.

Another possible design consideration is cost. Lower-cost sUAVs — such as the AR.Drone,[2] in the price range below $1500 — use less-expensive hardware for onboard sensing and processing. Although they contain an IMU and often a GPS unit, measurement accuracy and stability are usually reduced. Consequently, cameras and computer vision are often used for autonomous control to compensate for performance limitations. On the other hand, higher-cost platforms, such as AscTec, offer improved flight stability and more sophisticated sensing units, such as laser rangefinders or thermal infrared cameras (Mathe and Buoniu 2015).

Example Mission: Autonomous Search and Track

To illustrate the process of integrating autonomy into the design of a complex robotic system, in this section we consider the application of autonomy to a search and track mission. In this application, an sUAV must search for a target of interest, and once the target is found it must be tracked until some terminal condition is attained. For example, the target of interest might be a human poaching rhino horns in South Africa (Save the Rhino International 2015). The purpose of this search and track application is to search for potential poachers; if one is detected, their location is communicated to ground control; if the target is moving, follow the target; terminate mission when target is captured.

Autonomous search and track offers a useful illustration of the layered nature of autonomous behaviors. Next, we examine the cognitive capabilities underlying search and track and how they combine into an operational architecture.

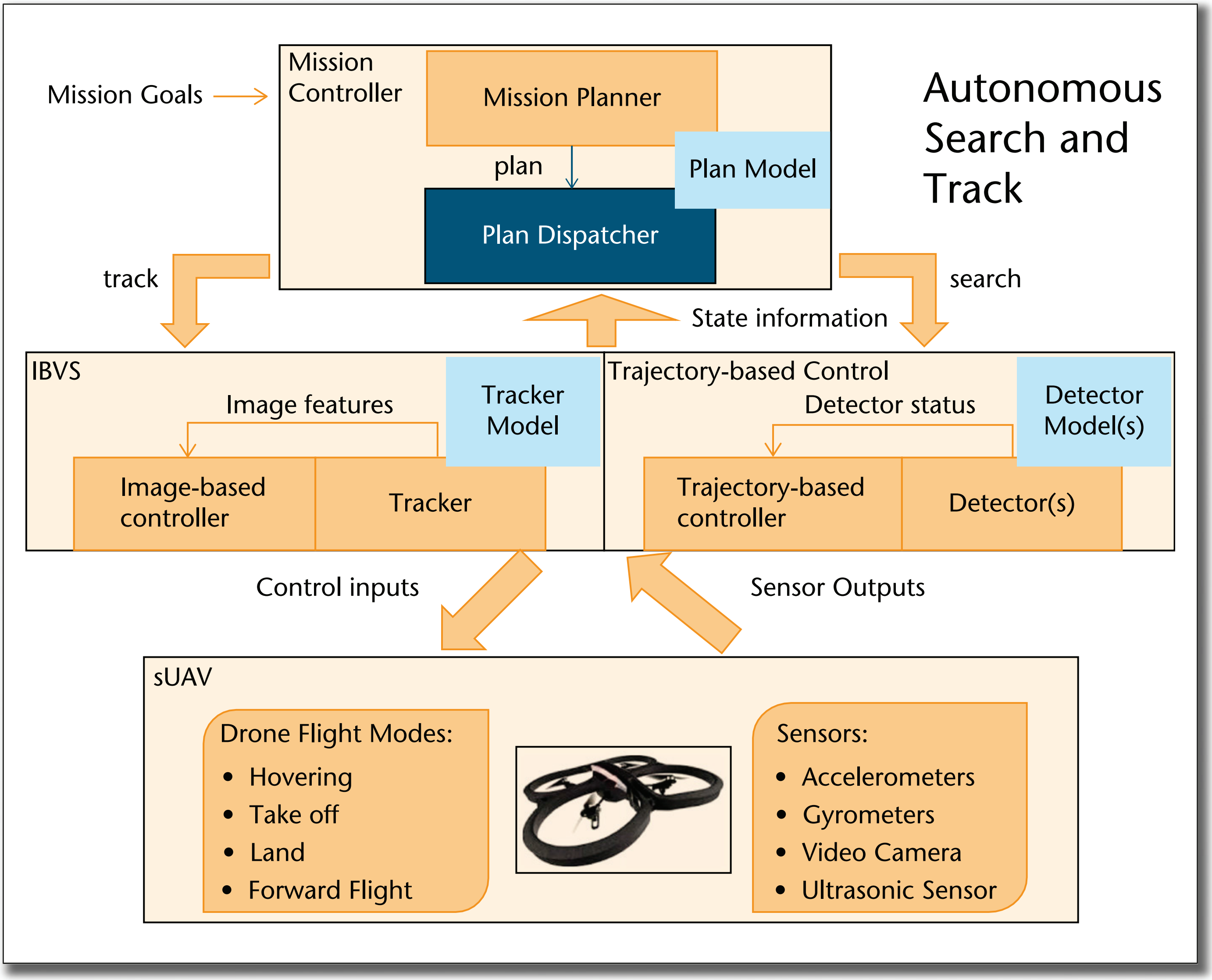


Figure 3. Operational Architecture for Autonomous Search and Track.

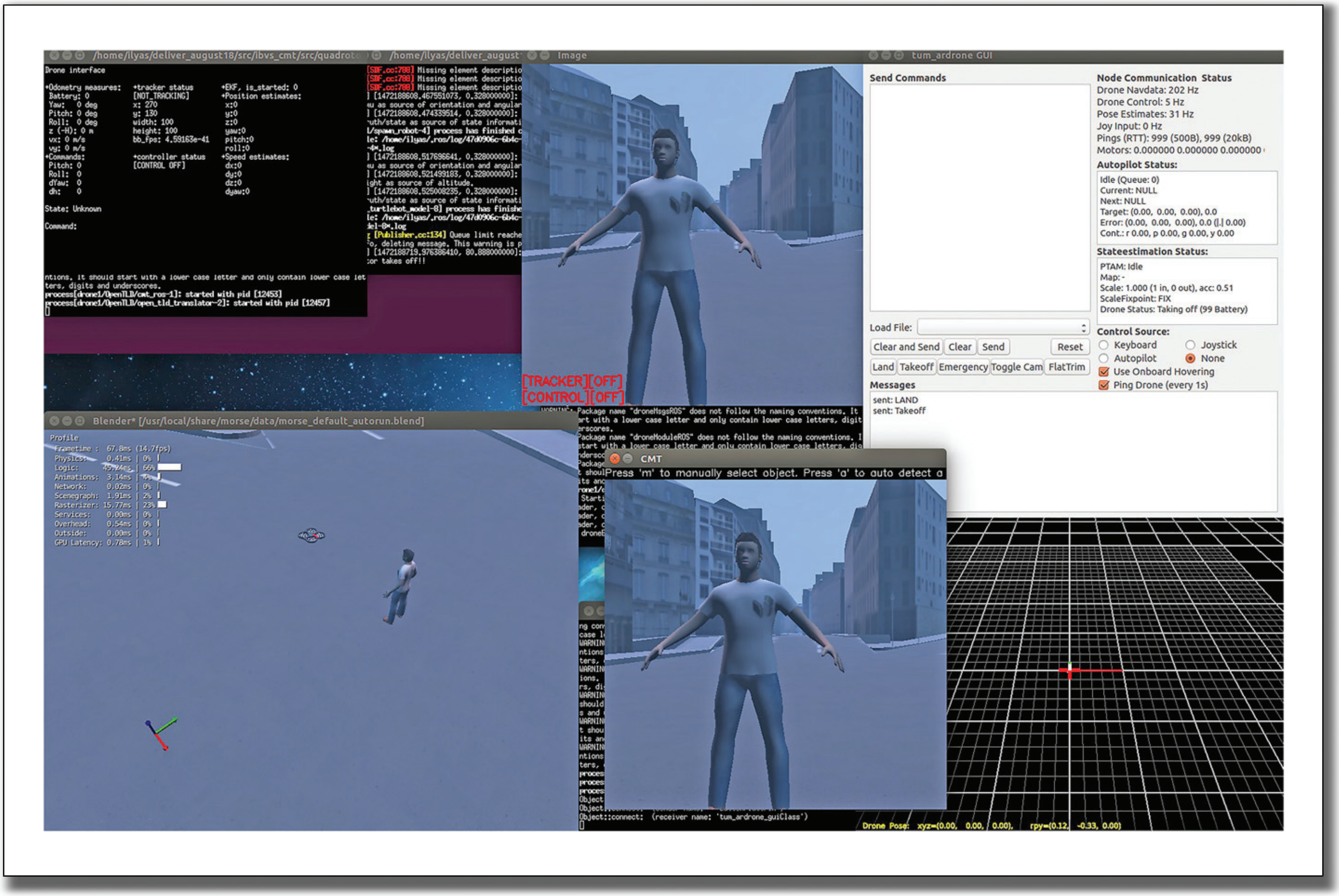
Component Cognitive Capabilities for Search and Track

The requirements for search and track give rise to four component cognitive capabilities.

First, sensing capabilities include the ability to detect a human being in motion. For machine detection, it is possible to use pretrained, open source software such as a HOG-Haar detector (Dalal and Triggs 2005) to detect human beings from video stream obtained from the UAV. Alternatively, a deep neural network can be trained, using images from relevant data sets (Krizhevsky, Sutskever, and Hinton 2012). For example, successful pedestrian detection using deep learning techniques has been proposed in the literature (Ouyang and Wang 2013).

Similarly, there are various approaches to tracking an object of interest, including tracking-learning-detection (Kalal, Mikolajczyk, and Matas 2012) and clustering of static-adaptive correspondences for deformable object tracking (CMT) (Nebehay and Pflugfelder 2015).

Second, an autonomous control alternating between following a trajectory (for search) and tracking an image must be developed. On the one hand, methods for achieving effective and robust trajectory-based control are available (Mathe and Buoniu 2015; Srivastava et al. 2014). On the other hand, visual servoing is the use of image features for controlling a robotic vehicle, and image-based visual servoing (IBVS) uses image features for control. The image features used in search and track are obtained by the tracker. The controller develops feedback commands to keep the bounding box in the center of the image plane. These control feedbacks are decoupled into altitude and lateral movements and are then sent to a calibrated controller, which outputs speed commands. See the paper by Pestana et al. (2014) for details of the controller used in the system developed for this article.


/home/ilyas/deliver_august18/src/ibvs_cmt/src/quadrot
/home/ilyas/deliver_august
Image
tum_ardrone GUI
Blender* [/usr/local/share/morse/data/morse_default_autorun.blend]
CMT
[TRACKER][OFF]
[CONTROL][OFF]

```
Drone interface

+Odometry measures:    +tracker status        +EKF, is_started: 0
 Battery: 0            [NOT_TRACKING]         +Position estimates:
 Yaw:   0 deg          x: 270                  x:0
 Pitch: 0 deg          y: 130                  y:0
 Roll:  0 deg          width: 100              z:0
 z (-H): 0 m           height: 100             yaw:0
 vx: 0 m/s             bb_fps: 4.59163e-41     pitch:0
 vy: 0 m/s                                     roll:0
+Commands:             +controller status     +Speed estimates:
 Pitch: 0              [CONTROL OFF]           dx:0
 Roll:  0                                      dy:0
 dYaw:  0                                      dz:0
 dh:    0                                      dyaw:0

State: Unknown

Command:

ntions, it should start with a lower case letter and only contain lower case let
ters, digits and underscores.
process[drone1/OpenTLD/cmt_ros-1]: started with pid [12453]
process[drone1/OpenTLD/open_tld_translator-2]: started with pid [12457]
```

```
[SDF.cc:788] Missing element descriptio
[SDF.cc:788] Missing element descriptio
[SDF.cc:788] Missing element descriptio
] [1472188608.467551073, 0.328000000]:
eu as source of orientation and angular
] [1472188608.474339514, 0.328000000]:
ruth/state as source of state informati
l/spawn_robot-4] process has finished c
le: /home/ilyas/.ros/log/47d0906c-6b4c-
-4*.log
] [1472188608.517696641, 0.328000000]:
eu as source of orientation and angular
] [1472188608.521499183, 0.328000000]:
ight as source of altitude.
] [1472188608.525008235, 0.328000000]:
ruth/state as source of state informati
_turtlebot_model-8] process has finishe
le: /home/ilyas/.ros/log/47d0906c-6b4c-
del-8*.log
g [Publisher.cc:134] Queue limit reache
fo, deleting message. This warning is p
] [1472188719.976386410, 80.888000000]:
tor takes off!!
```

Send Commands

Load File:

Clear and Send | Clear | Send | Reset

Land | Takeoff | Emergency | Toggle Cam | FlatTrim

Messages

sent: LAND
sent: Takeoff

Node Communication Status
Drone Navdata: 202 Hz
Drone Control: 5 Hz
Pose Estimates: 31 Hz
Joy Input: 0 Hz
Pings (RTT): 999 (500B), 999 (20kB)
Motors: 0.000000 0.000000 0.000000

Autopilot Status:
Idle (Queue: 0)
Current: NULL
Next: NULL
Target: (0.00, 0.00, 0.00), 0.0
Error: (0.00, 0.00, 0.00), 0.0 (|.| 0.00)
Cont.: r 0.00, p 0.00, g 0.00, y 0.00

Stateestimation Status:
PTAM: Idle
Map: -
Scale: 1.000 (1 in, 0 out), acc: 0.51
ScaleFixpoint: FIX
Drone Status: Taking off (99 Battery)

Control Source:
○ Keyboard ○ Joystick
○ Autopilot ● None
☑ Use Onboard Hovering
☑ Ping Drone (every 1s)

```
Profile
 Frametime :  67.8ms (14.7fps)
 Physics:      0.41ms | 0%
 Logic:       45.24ms | 66%
 Animations:   3.14ms | 4%
 Network:      0.02ms | 0%
 Scenegraph:   1.91ms | 2%
 Rasterizer:  15.77ms | 23%
 Services:     0.00ms | 0%
 Overhead:     0.54ms | 0%
 Outside:      0.00ms | 0%
 GPU Latency:  0.78ms | 1%
```

```
WARNING: Package name "droneMsgsROS" does not follow the naming conventions. It
art with a lower case letter and only contain lower case letters, digit
erscores.
Package name "droneModuleROS" does not follow the naming conventions. I
start with a lower case letter and only contain lower case letters, dig
Object::connect:  (receiver name: 'tum_ardrone_guiClass')
```

Press 'm' to manually select object. Press 'a' to auto detect a

Drone Pose: xyz=(0.00, 0.00, 0.00),   rpy=(0.12, -0.33, 0.00)

Figure 4. Testing Autonomous Search and Track Using the Morse Blender Simulation Environment.

Third, a deliberative layer must be able for deciding which behavioral mode (search or track) should be operating. This is a role provided by the executive (Williams and Nayak 1997), a system for dispatching actions, managing time and resources, monitoring the execution of the plan, and initiating plan changes. Execution models can be based on procedures or on finite-state machines (Bohren et al. 2011).

Finally, a deliberative planning layer must be used to devise a plan for search. Planning for search and track (and variants such as search and rescue) is one of the oldest problems in operations research. The foundations of the theory of search are found in the paper by Koopman (1957), which divides the problem into two subproblems: optimal allocation of effort (that is, what percentage of time to spend in a given subregion) and optimal rescue track. Planning for search is potentially challenging because it is assumed that the target is in an area that is too large to search exhaustively, the target's location is represented as a probability distribution over subregions of the search area, and the target may or may not be moving. A typical planning cycle involves the production of a probability distribution for the object's location at the time of the next search. A trajectory uses this distribution along with a list of assigned search assets to produce operationally feasible search plans that maximize the increase in probability of detecting the object. If the search is unsuccessful, a posterior probability map for object location that accounts for the unsuccessful search and the possible motion of the object is generated, providing the basis for planning the next increment of search (Kratzke, Stone, and Frost 2010).

These four cognitive components occupy a wide range of behaviors between the purely reactive and the purely deliberative. They interact in complex ways that require an effective and robust coordination mechanism, and they provide a good example of distributed, hierarchical organization, as we now examine.

Operational Architecture for Search and Track

We've noted that autonomy architectures are fundamentally hierarchical, distributed, and human centered. In this section we frame these features in the context of search and track.

Figure 3 proposes a hierarchical autonomous architecture for search and track. Mission goals are inputs to the system. A mission controller acts as planner and plan executive. High-level plans for searching are generated and dispatched at this level. The ability to switch between searching and tracking is represented

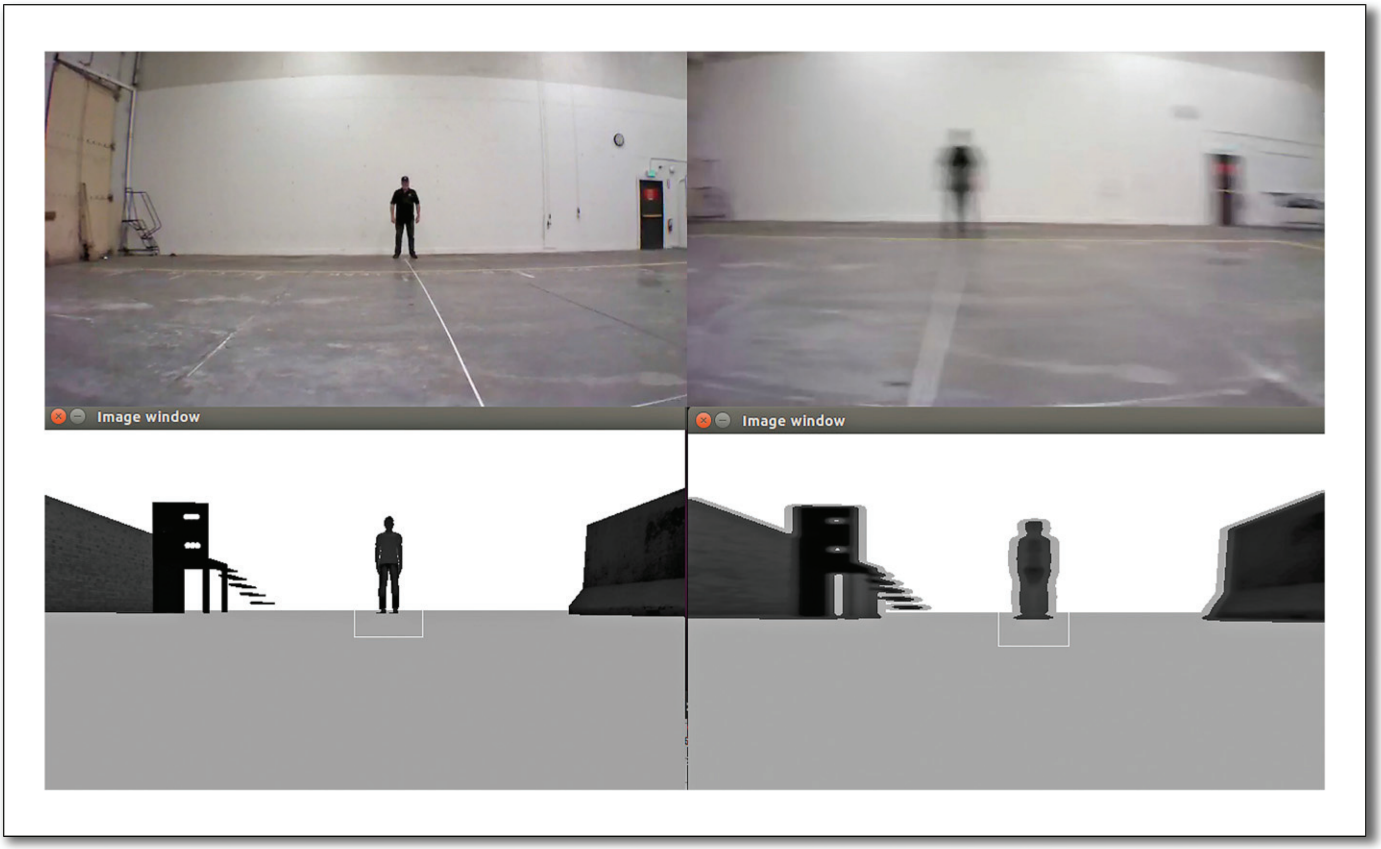


Figure 5. Simulating Motion Blur to Mimic Real Flight Data.

as a high-level loop from the plan dispatcher to the lower-level controllers. In this manner, low-level control alternates between an IBVS for tracking and a trajectory-based controller for searching. Both consist of a control loop that utilizes vision for estimating position (during search) and for keeping a target of interest in view (for tracking). Each controller sends control actions to manage the flight modes of the sUAV and uses sensory inputs inside their control loop. A hierarchical organization is transformed into an operational architecture by deciding, roughly speaking, what goes where. This decision has two parts. The first allows for a determination of the vehicle architecture, the capabilities that reside onboard. One design trade-off here is communication overhead and increased processing power (as well as cost). sUAVs are intrinsically limited to small payload capabilities as well as onboard processing power. One solution is to migrate some of these capabilities to a ground computer. For example, Chakrabarty et al. (2017) describe a tracking system using an AR.Drone with a visual search loop running on a ground computer.

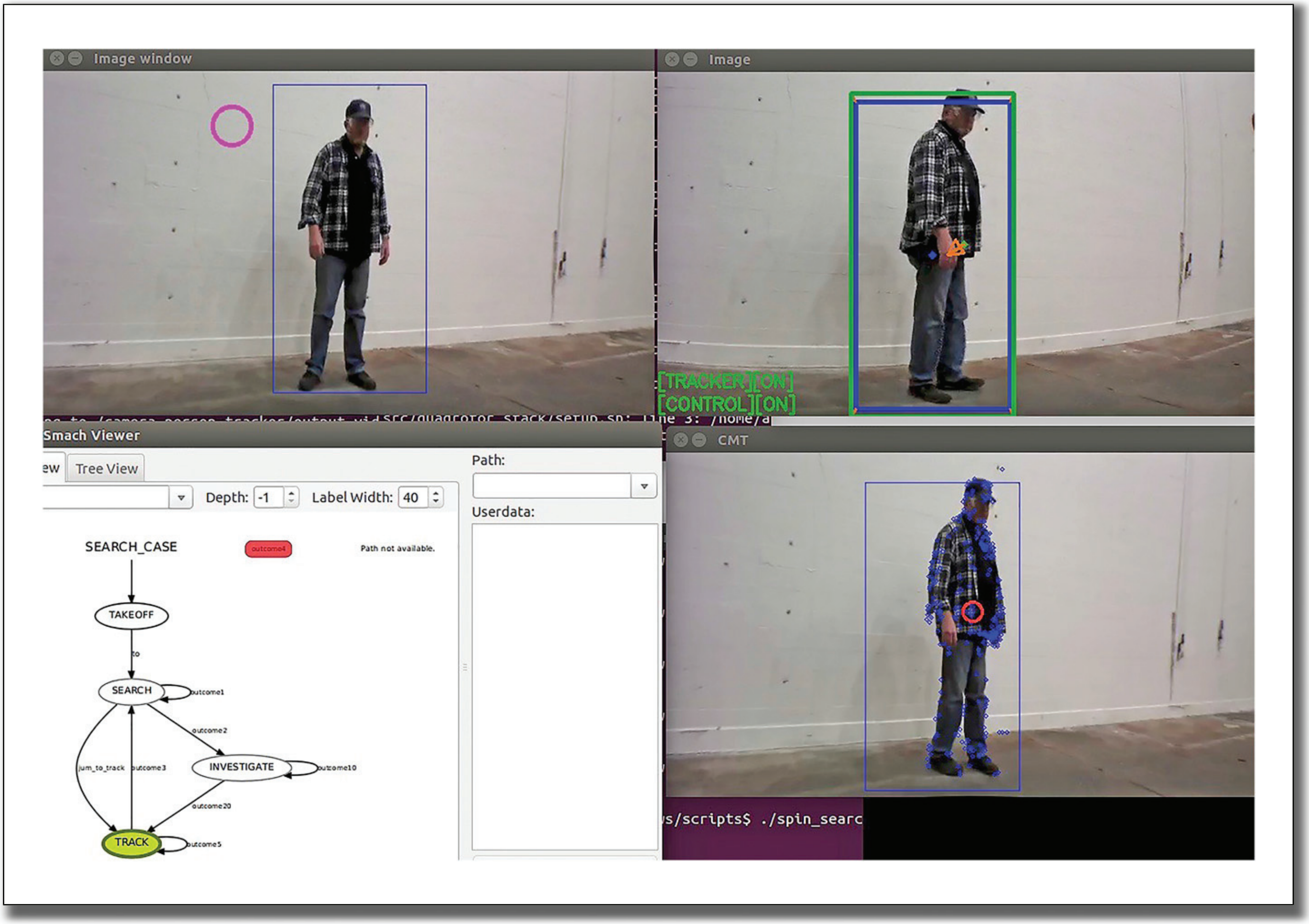
The second part of the decision requires specifying the boundaries between human and machine decision making. In the search and track example, it may be decided that humans at a console perform the search phase of the operation, whereas the sUAV conducts the tracking on its own. Clearly, other hybrid designs between the fully manual and the fully autonomous are possible.

Evaluating Autonomy During Conceptual Design

In an earlier section it was noted that part of the conceptual design phase involves analyzing and optimizing candidate designs. We close this study in this section by evaluating designs for autonomy, using the performance metrics defined earlier and using the search and track example. We focus on an example of evaluating a single autonomous behavior: the ability of the executive component of an autonomous sUAV to alternate between search and track to keep a target in view. This behavior was chosen because it requires an effective combination of inference and sensing to perform effectively. Morris et al. (2017) provide a more detailed discussion.

First, we review some general requirements. The overall purpose of the mission was a special case of an autonomy technology demonstration, that is, to test new autonomy capabilities related to search and track. For this reason, by definition, autonomy is enabling for this mission. Second, the interest was in a fast development and testing cycle, which means that preference was given to cognitive components that were readily available and could be integrated quickly.

For these reasons, the hardware and software were chosen to enable quick development and integration. First, an IBVS system (Pestana et al. 2014) and a tracker (Nebehay and Pflugfelder 2014) were integrated as modules in ROS (Quigley et al. 2009). A HOG-Haar



```
s/scripts$ ./spin_searc
```

Figure 6. Testing Autonomous Search and Track at the Ames Research Center Indoor Facility.

detector (Dalal and Triggs 2005) was integrated with the trajectory planner to search for a human target. Plan generation for search and track and the dispatching of plans were implemented using either ROSPlan (Cashmore et al. 2015) or SMACH (Bohren et al. 2011) for comparison. A Parrot AR.Drone quadrotor was commanded from a computer via WiFi link using the AR.Drone Autonomy ROS package[2] or in simulation using Gazebo (Koenig and Howard 2004) as well as MORSE-Blender. All simulations were run under Ubuntu 14.04 LTS 64-bit and an Intel Xeon E5-2630 @ 2.60 GHz × 17 CPU, a NVIDIA Quadro K5000 GPU, and 32 GB of RAM.

The simulation platform provided safe and fast opportunities for comparing the performance of different algorithms or models (figure 4) and allowed a quick way to fine-tune certain parameters such as controller gains and delays between target identification before flight experiments. Other trade studies for autonomy that benefitted from simulation include the choice of onboard sensors and the study of communication delay. For example, thermal cameras were studied for nighttime detection.

For target search from an sUAV, one important parameter is identification of the target from a moving platform. Images from a moving vehicle are often subject to motion blur. This makes object identification extremely difficult. Figure 5 shows how motion blur was reproduced in simulation. Simulated tests allowed for the discovery of the maximum velocity of the vehicle in which object detection still worked. Furthermore, it was found that different algorithms for object detection had different effects on the blur. While deep learning methods always outperformed simple HOG-Haar networks for pedestrian detection, they were also more susceptible to missed detection in case of motion blur. These factors helped us select the maximum speed at which the vehicle was allowed to operate while searching for a target.

In addition to simulation, flight tests during conceptual design were conducted in an indoor test facility at the NASA Ames Research Center. Some of the results summarized here are discussed in more detail elsewhere (Chakrabarty et al. 2017). The indoor facility at the Ames Research Center is a controlled environment that can be used to test systems before moving on to realistic outdoor environments. Specifically, for

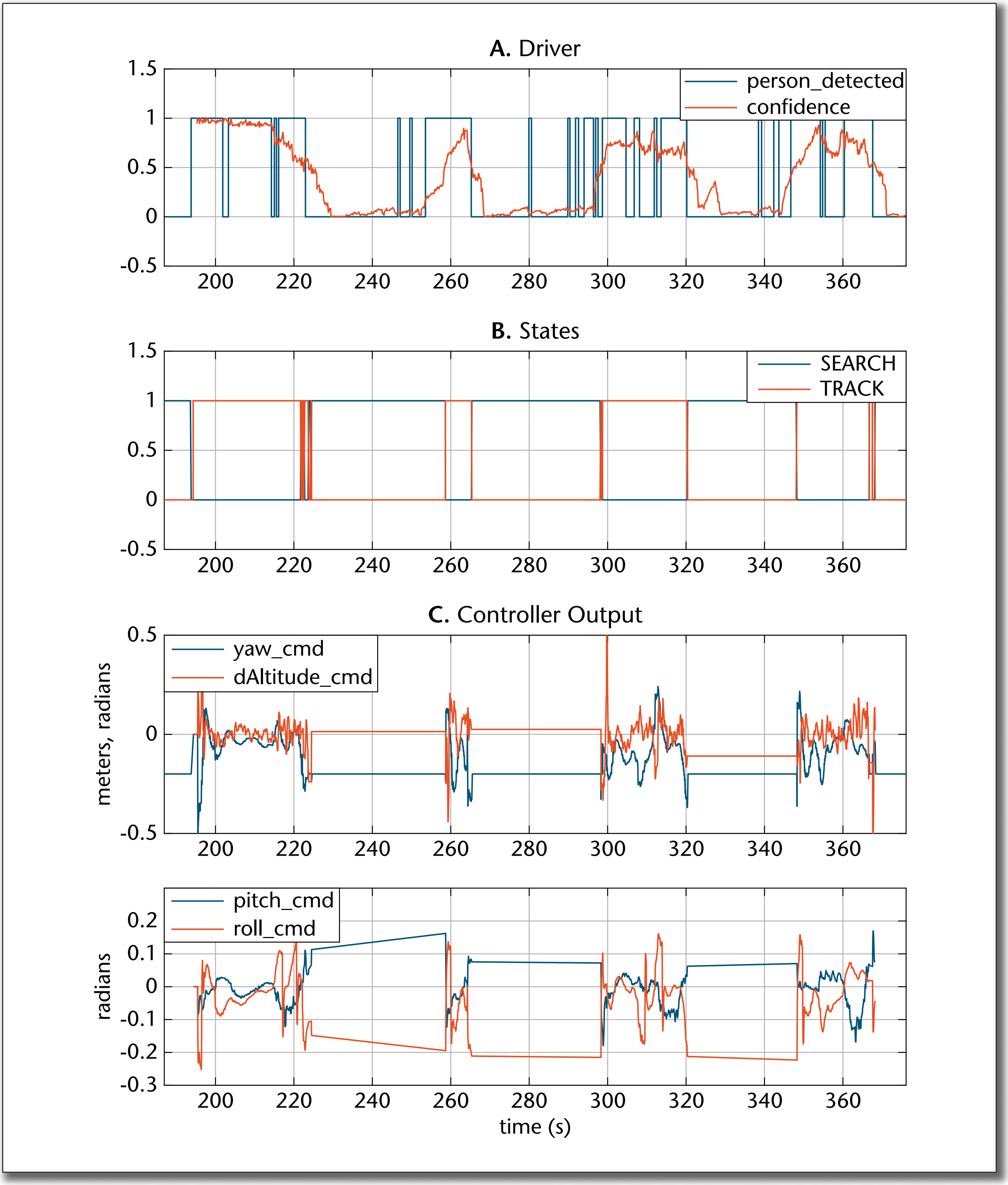


Figure 7. Testing the Ability of a Plan Executive to Alternate between Search and Track.

(a) The executive monitors the state variables of the detector and tracker to determine whether the target of interest is still in view. (b) The executive controls the high-level state variables search and track for replanning purposes and invokes the detector and tracker. (c), (d) The controller behavior changes based on whether it is given a trajectory to follow (search) or whether it is tracking an image.

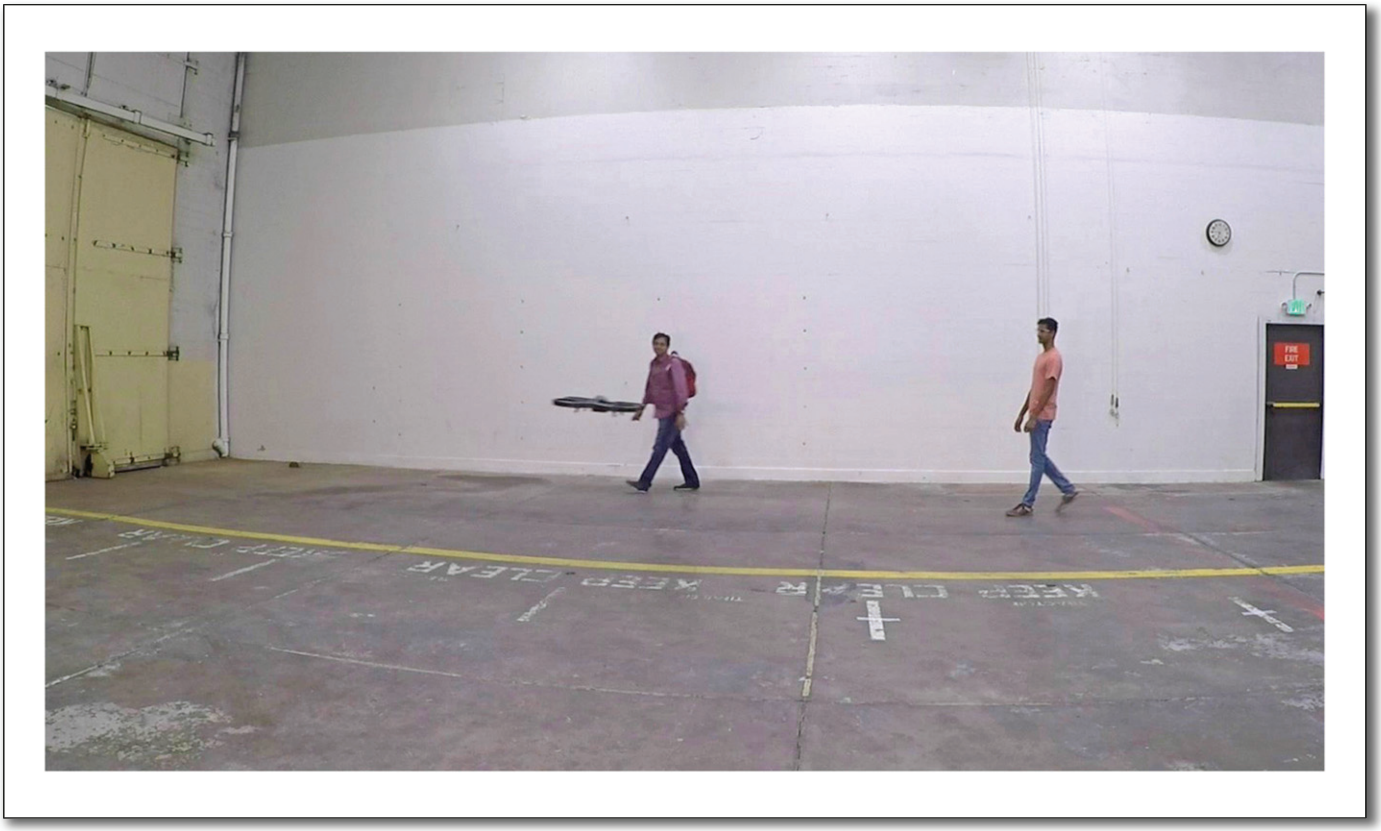


Figure 8. Tracking Person with Backpack Only.

search and track, the interactions between the continuous planning framework and the underlying sensing and control frameworks can be studied in detail. On the other hand, it is not a good environment for testing autonomy robustness; for this, a more realistic outdoor environment is required.

As noted, an executive dispatches plans and monitors their progress, possibly triggering new planning. In search and track, the executive must be able monitor the progress of the search for a target and to switch to tracking mode if the target is found.

To test the executive in our controlled environment, a human target would begin out of sight of the sUAV. The sUAV would take off and begin a pattern maneuver (such as a square or spiral pattern) in search mode. Once the human target was found, we studied the response time of the system to transition into track mode. Once in track mode, the target would first move in a slow walk to demonstrate simple tracking behavior. At some point, the target would take evasive maneuvers until the subject was out of sight to the sUAV. We then could observe the transition back into search mode, which consisted of following another pattern search (typically a simple rotation). This alternating behavior of search and track typically was repeated many times in a single run.

Figure 6 shows a screenshot of the run time behavior of the system. The top left window shows the output of the human detector window with the bounding box. The bounding box is inherited by the CMT tracker, as shown in the bottom right window. The top right window shows the output of the IBVS controller. The bottom left window shows part of the SMACH planner visualization.

Figure 7 illustrates the behavior of the executive. The variables *person_detected* and *confidence* in the CMT tracker are monitored by the executive to determine whether the system should be searching or tracking. The system is designed to switch from searching to tracking only when the detector finds the human figure and the confidence variable exceeds the desired threshold. This aspect is important as both variables are prone to noise because of their dependency on visual features.

Other indoor tests explored alternatives to target detection. First, a human detector model based on training a deep neural network was used to identify persons with backpacks (for example, to be used by an sUAV to monitor for drug smuggling across borders). Figure 8 shows a variation of the indoor experiment in which the system distinguished a human being with a backpack from one with no backpack. Other tests examined search and track with a thermal camera. The system was able to identify multiple human beings from only a low- resolution thermal image using deep networks.

Summary

Mission and operational requirements for effective and robust search and track lead to defining a number of cognitive capabilities in sensing, communication, and intelligent control and navigation during conceptual design. Autonomy can be enabling or enhancing for one or more of these capabilities. The design components of autonomy for search and track consist of algorithms and associated software models for planning, execution, search, detection, and tracking; sensing, processing, and memory hardware; aerial platforms; and infrastructure for development and testing (including ROS, simulation environments such as MORSE, and indoor or outdoor field testing). In addition, autonomy performance testing for search and track scans a wide range of goal-directed behaviors, from purely reactive (maintaining an object in view) to tactical (knowing when to switch from search to track mode when an object is found) to deliberative (generating effective plans for search). Finally, testing a system for autonomy involves the three metrics of effectiveness (does autonomy enhance or enable behaviors that accomplish mission and operational goals?), robustness (does autonomy adapt successfully to changes in its mission or operational environment?), and safety. These considerations of autonomy have never been a part of traditional conceptual design, as discussed earlier.

We presented a simple example of conceptual design for a search and track mission based on fast development and testing of autonomous cognitive capabilities in planning and execution. These requirements led to selection of platform and sensor hardware and software. We illustrated the role of simulation and testing in isolating a particular behavior of the executive to alternate between search and track, measuring the effectiveness of this component.

Conclusion

This article argued for autonomy as part of conceptual design to ensure optimal design and acceptance. We reviewed principles of conceptual design for aeronautical vehicles, as well as the capabilities and performance metrics of autonomous systems. We reviewed architectures for autonomy and argued that autonomy is inherently layered, distributed, and human centered. We briefly discussed how considerations of autonomy change the conceptual design process, starting from determining whether autonomy is enabling or enhancing for a mission. Finally, we walked through a simple but illustrative example of autonomous search and track for sUAVs.

As noted by Russell and Norvig (2016), "Intelligence is concerned mainly with rational action; ideally, an intelligent agent takes the best possible action in a situation." Here we have quantified rational action as correctness and responsiveness of an autonomous system. Considering autonomy at the conceptual design phase should be an integral part of designing future autonomous systems. Specifically, a more rigorous design based on autonomy will reduce design cycles and better ensure successful deployment.

Acknowledgments

The authors contributed equally to this article. This work was conducted through NASA's Convergent Aeronautics Solution project, through the Design Environment for Novel Vertical Lift Vehicles (DELIVER) subproject. The DELIVER team also consisted of Xavier Bouyssounouse and Rusty Hunt. The authors acknowledge contributions made by interns Suneel Belkhele, Sohan Vichare, and Adiyan Kaul and thank the anonymous reviewers for helpful suggestions in revising a previous draft.

Notes

1. ti.arc.nasa.gov/tech/asr/groups/planning-And-scheduling/remote-Agent/
2. github.com/AutonomyLab/ardrone_autonomy

References

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An Architecture for Autonomy. *International Journal of Robotics Research* 17(4): 315–37. doi.org/10.1177/027836499801700402

Bachrach, A.; Prentice, S.; He, R.; and Roy, N. 2011. Robust Autonomous Navigation in GPS-Denied Environments. *Journal of Field Robotics* 28(5): 644–66. doi.org/10.1002/rob.20400

Bayouth, M.; Nourbakhsh, I.; and Thorpe, C. 1997. A Hybrid Human-Computer Autonomous Vehicle Architecture. *In Third ECPD International Conference on Advanced Robotics, Intelligent Automation, and Control*, vol. 15, p. 17. London, UK.

Bohren, J.; Rusu, R.; Jones, E.; Marder-Eppstein, E.; Pantofaru, C.; Wise, M.; Msenlechner, L.; Meeussen, W.; and Holzer, S. 2011. Towards Autonomous Robotic Butlers: Lessons Learned with the PR2. *In 2011 IEEE International Conference on Robotics and Automation*, pp. 5568–75. New York: IEEE.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Huros, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*. Palo Alto, CA: AAAI Press.

Chakrabarty, A.; Morris, R. A.; Bouyssounouse, X.; and Hunt, R. 2017. An Integrated System for Autonomous Search and Track with a Small Unmanned Aerial Vehicle. In *AIAA Information Systems-AIAA Infotech@ Aerospace*. Palo Alto, CA: AAAI Press. doi.org/10.2514/6.2017-0671

Christoffersen, K., and Woods, D. D. 2002. How to Make Automated Systems Team Players. In *Advances in Human Performance and Cognitive Engineering Research*, vol. 2. Greenwich, CT: JAI Press/Elsevier. doi.org/10.1016/S1479-3601(02)02003-9

Coste-Maniere, E., and Simmons, R. 2000. Architecture, the Backbone of Robotic Systems. In *Proceedings of the International Conference on Robotics and Automation*, vol. 1, pp. 67–72. New York: IEEE.

Dalal, N., and Triggs, B. 2005. Histograms of Oriented Gradients for Human Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. New York: IEEE. doi.org/10.1109/CVPR.2005.177

Ingrand, F., and Ghallab, M. 2014. Deliberation for Autonomous Robots: A Survey. *Artificial Intelligence* 247: 10–44.

Jerath, K., and Langelaan, J. W. 2016. Simulation Framework for Incorporating Sensor Systems in UAs Conceptual Design. In *AIAA Modeling and Simulation Technologies Conference*. Reston, VA: American Institute of Aeronautics and Astronautics. doi.org/10.2514/6.2016-1186

Johnson, M.; Bradshaw, J. M.; Feltovich, P. J.; Jonker, C. M.; Van Riemsdijk, B.; and Sierhuis, M. 2014. Coactive Design: Designing Support for Interdependence in Joint Activity. *Journal of Human-Robot Interaction* 3(1): 43–69. doi.org/10.5898/JHRI.3.1.Johnson

Jonsson, A.; Morris, R.; and Pedersen, L. 2007. Autonomy in Space: Current Capabilities and Future Challenges. *AI Magazine* 28(4): 27–42.

Kalal, Z.; Mikolajczyk, K.; and Matas, J. 2012. Tracking Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(7): 1409–22. doi.org/10.1109/TPAMI.2011.239

Koenig, N., and Howard, A. 2004. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. New York: IEEE. doi.org/10.1109/IROS.2004.1389727

Koopman, B. 1957. The Theory of Search. III. The Optimum Distribution of Searching Effort. *Operations Research* 5(5): 613–26. doi.org/10.1287/opre.5.5.613

Kortenkamp, D., and Simmons, R. 2008. Robotic Systems Architectures and Programming. In *Springer Handbok of Robotics*, edited by B. Siciliano and O. Khatib, 187–206. New York: Springer. doi.org/10.1007/978-3-540-30301-5_9

Kratzke, T. M.; Stone, L.; and Frost, J. R. 2010. Search and Rescue Optimal Planning System. In *Proceedings of the 13th International Conference on Information Fusion*. New York: IEEE.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. Paper presented at the NIPS 2012 Neural Information Processing Systems. Lake Tahoe, NV, December 3–6.

Máthé, K., and Buoniu, L. 2015. Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection. *Sensors (Basel)* 15(7): 14887–916. doi.org/10.3390/s150714887

Morris, R.; Chakrabarty, A.; Bouyssounouse, X.; and Hunt, R. 2017. Autonomous Search-Detect-Track for Small UAVS. Paper presented at the ICAPS-17 Workshop on Integrated Planning and Execution. Pittsburgh, PA, June 19–20.

Murphy, R., and Shields, J. 2012. *Defense Science Board Task Force Report the Role of Autonomy in DoD Systems*. Technical Report CD 1172. Washington, DC: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics.

Nebehay, G., and Pflugfelder, R. 2014. Consensus-Based Matching and Tracking of Keypoints for Object Tracking. In *IEEE Winter Conference on Applications of Computer Vision*, 862–9. New York: IEEE. doi.org/10.1109/WACV.2014.6836013

Nebehay, G., and Pflugfelder, R. 2015. Clustering of Static-Adaptive Correspondences for Deformable Object Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2784–91. New York: IEEE. doi.org/10.1109/CVPR.2015.7298895

Ouyang, W., and Wang, X. 2013. Joint Deep Learning for Pedestrian Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2056–63. New York: IEEE.

Pestana, J.; Sanchez-Lopez, J. L.; Saripalli, S.; and Campoy, P. 2014. Computer Vision Based General Object Following for GPS-Denied Multirotor Unmanned Vehicles. In *American Control Conference*. New York: IEEE. doi.org/10.1109/ACC.2014.6858831

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; and Wheeler R.; and Ng, A. 2009. ROS: An Open-Source Robot Operating System. In *IEEE International Conference on Robotics and Automation*. New York: IEEE.

Raymer, D. 2012. *Aircraft Design: A Conceptual Approach*, Fifth edition. Reston, VA: American Institute of Aeronautics and Astronautics. doi.org/10.2514/4.869112

Russell, S. J., and Norvig, P. 2016. *Artificial Intelligence: A Modern Approach*. London: Pearson Education Limited.

Save the Rhino International. 2015. *The Use of Drones in Rhino Conservation*. www.savetherhino.org/thorny-issues/the-use-of-drones-in-rhino-conservation/.

Shladover, S. 2016. The Truth about Self-Driving Cars. *Scientific American* 314(6): 52–7. doi.org/10.1038/scientificamerican0616-52

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. In *Proceedings of the IEEE Conference on Robotics and Automation*. New York: IEEE. doi.org/10.1109/ICRA.2014.6906922

Tyagi, A., and Nanda, J. 2016. ATLAS: Big Data Storage and Analytics Tool for ATM Researchers. In *AIAA Aerospace and Infotech*. Reston, VA: American Institute of Aeronautics and Astronautics.

Wettergreen, D.; Foil, G.; Furlong, P. M.; and Thompson, D. R. 2014. Science Autonomy for Rover Subsurface Exploration of the Atacama Desert. *AI Magazine* 35(4): 47–60. doi.org/10.1609/aimag.v35i4.2554

Williams, B. C., and Nayak, P. P. 1997. A Reactive Planner for a Model-Based Executive. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, vol. 2, 1178–85. San Francisco, CA: Morgan Kaufmann.

Robert Morris is a senior researcher in the Exploration Technology Directorate, Intelligent Systems Division, at NASA Ames Research Center. His major research focus has been on designing automated planning and scheduling systems for NASA's exploration systems and mission operations. Morris has published numerous articles on constraint-based approaches to planning and scheduling and has coauthored a book on methods for temporal reasoning in AI planning and scheduling systems. More recently his focus has been on designing architectures for autonomous aeronautical systems.

Anjan Chakrabarty is a research engineer with the Advanced Control and Evolvable Systems Group in the Intelligent Systems Division at NASA Ames Research Center. He is employed by SGT Inc. Prior to joining SGT, he was a NASA postdoctoral program fellow. Chakrabarty is working on several UAV autonomy projects at Ames Research Center, including UTM TCL4 V2V communication and collision avoidance. He also has investigated the effect of autonomy on the design of UAVs. Chakrabarty earned his PhD in aerospace engineering from Pennsylvania State University. He was a member of the winning team Pipistrel-USA.com of the NASA Green Flight Challenge. Chakrabarty was responsible for flight path planning based on local wind field data using his PhD research.