A Systems Analysis of the Introduction of Unmanned Aircraft Into Aircraft Carrier Operations

Jason C. Ryan and Mary L. Cummings

***Abstract*—Recent advances in unmanned and autonomous vehicle technology are accelerating the push to integrate these vehicles into environments, such as the National Airspace System, the national highway system, and many manufacturing environments. These environments will require close collaboration between humans and vehicles, and their large scales mean that real-world field trials may be difficult to execute due to concerns of cost, availability, and technological maturity. This paper describes the use of an agent-based model to explore the system-level effects of unmanned vehicle implementation on the performance of these collaborative human–vehicle environments. In particular, this paper explores the introduction of three different unmanned vehicle control architectures into aircraft carrier flight deck operations. The different control architectures are tested under an example mission scenario using 22 aircraft. Results show that certain control architectures can improve the rate of launches, but these improvements are limited by the structure of flight deck operations and nature of the launch task (which is defined independently of vehicles). Until the launch task is improved, the effects of unmanned vehicle control architectures on flight deck performance will be limited.**

***Index Terms*—Agent-based modeling, decision support system, human–automation interaction, human supervisory control.**

I. Introduction

IMPROVING the capabilities of unmanned vehicles has been a key area of research over the past several years, but it is only relatively recently that they have improved to the point that integrating unmanned systems into civilian environments has become realistic. Several current examples exist: the Rio Tinto mining corporation is currently fielding autonomous hauling trucks in several locations [1], [2]; the Kiva systems order fulfillment system, which utilizes robotic pallet carriers, is used by online retailers Amazon and Zappos.com, among others [3]; automobile manufacturers continue to introduce new automated features, such as adaptive cruise control, lane departure warnings, and other technology while continuing to pursue automated road train and more advanced driverless technologies. Current initiatives by the defense advanced research projects agency (DARPA) seek to advance the capabilities of humanoid robotic systems in urban search and rescue scenarios (the DARPA Robotics Challenge), while other research seeks to address how robotic manufacturing systems cannot only work more safely near humans, but how they can also have a better understanding of the roles and tasks of their human collaborators [4].

Each of the domains described above can be characterized as a human centered, or heterogeneous manned-unmanned environment (HMUE): places where unmanned vehicles, human collaborators, and manned vehicles interact in the same physical space. A variety of heterogeneous manned environments, such as airport operations and aircraft carrier flight decks, already exist today. As unmanned vehicles transition into these domains, new paradigms of interaction will be required for human collaborators to work effectively with these unmanned vehicles. Changes to operating procedures may also be required to accommodate differences in capabilities between unmanned and manned vehicles. Very little research has addressed what these changes might be and how these changes, and the resulting introduction of unmanned vehicles, might affect the performance of the broader system.

Insight into the effects of unmanned vehicle integration on the performance of these larger scale systems could be gained by field testing prototype systems, but these environments are of such large scale (in physical size and number of active elements) that testing them in any meaningful manner would require significant investments of time, money, and physical resources. Even if these resources could be provided, if the unmanned vehicle systems being tested are not sufficiently similar to the future systems that would be deployed, the results may not be useful. Such tests might also place the autonomous systems, the researchers involved, and other by-standers in unforeseen danger, while also being limited in the number and types of conditions that could be explored.

Agent-based modeling [5] is one potential approach for addressing such limitations; these models attempt to replicate the behavior of individual actors within a given environment. Constructing such models of the real-world system provides an avenue for testing unmanned vehicle performance under a variety of test conditions that may not be easily achievable in live conditions. This paper examines the use of an agent-based simulation of a candidate HMUE, the aircraft carrier flight deck, for which the United States Navy is currently testing an unmanned carrier fighter aircraft that is expected to integrate fully into operations with manned aircraft and human crew [6], [7]. Flight deck operations are in many ways analogous to airport operations; in both environments, aircraft begin parked in a designated area, then coordinate with crew (or other controllers) to taxi to one of a few designated launch areas. With the Federal Aviation Administration's (FAA's) mandate to introduce unmanned aircraft

Manuscript received May 1, 2014; revised July 13, 2014 and October 1, 2014; accepted November 1, 2014. This work was supported by the Office of Naval Research Science of Autonomy program under Contract #N000140910625. The opinions expressed in this paper are those of the authors and do not necessarily reflect the opinions of the sponsoring agency. This paper was recommended by Associate Editor D. Gillan.

J. C. Ryan is with the Humans and Automation Lab, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: jachryan@gmail.com).

M. L. Cummings is with the Humans and Autonomy Lab, Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (e-mail: mary.cummings@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2014.2376355

into the national airspace system by 2015, inserting unmanned vehicles into ground operations will be a paramount concern. While the FAA has recognized the importance of ground operations in this integration, their major focus at this time appears to be on airborne operations [8]. While there are differences in the structure of airport and aircraft carrier operations (runways versus an open deck) and the role of human collaborators in those environments, lessons learned from the carrier flight deck should be applicable to airport operations.

This paper begins with a brief description of agent-based simulation and its applicability to modeling these environments. The three different unmanned vehicle control architectures of interest are then described, followed by a discussion of current aircraft carrier flight deck operations and how they are modeled within the Multi-Agent Safety and Control Simulation (MASCS). The modeling of the unmanned vehicle control architectures within MASCS is then described, followed by a description of the testing of these vehicles under mission conditions utilizing sets of 22 aircraft. Additional tests then explored how improvements to the launch task (with properties independent of vehicle control architecture) affect flight deck performance, as well as how changes to key control architecture design parameters can improve vehicle performance under these new conditions.

II. Related Work

A. *Agent-Based Modeling*

As described by Bonabeau [5], agent-based modeling is as much a perspective as it is a methodology, whereas the use of discrete-event or systems dynamics models may elicit specific ideas of structure and constituent components, agent-based models vary widely in content and in methods of application. Examples range from human decision-making (e.g., [9]) to epidemiology and disease transmission (e.g., [10]) to transportation systems involving both ground and aerial vehicles (e.g., [11]), among many others. The common theme is that, in each case, the agent-based modeling paradigm views the world "from the perspective of its constituent units" [5]—that the agents retain the same independence and exhibit similar decision-making outcomes as the real-world entities they are intended to replicate. The agents are then placed into the simulated environment, given a set of goals (which may be self-generated during the simulation), and allowed to make decisions independently from that point forward. An additional key to agent-based simulation development is the use of simple rules of behavior [5], [12]: agents are given limited options in any given decision they make. The simultaneous application of these simple decisions by the simulated agents is typically sufficient to replicate the behavior of the real-world system.

While agent-based modeling has been used to validate control and sensing algorithms for a variety of Unmanned Aerial Vehicle (UAV) systems (e.g., [13]), these studies have primarily addressed systems involving the collaboration of multiple vehicles in performing tasks, such as searching for and tracking targets. Another common area of interest is in the modeling and control of UAV "swarms" (large groups of robots coordinating on a specific task) [14]. Agent-based models have also been built to test unmanned vehicle survivability [15] and single UAV search [16]. However, none of these examples has considered the role of the human in interacting with the vehicle system, although other nonagent-based models of human-UAV interaction have been constructed [17], [18]. These latter models focused on the human as a processing server working within a queuing system, considering the behavior of the vehicle only in that it requires the operator to provide it a task. These latter models only, however, examined a single system of interest and did not construct models over a range of different vehicle control architectures. These models typically included only a few (4–8) aircraft in their models, as well.

No prior work could be located regarding the models of aircraft carrier flight deck operations, nor of agent-based models of similar collaborative human–vehicle domains (such as airport ground operations or mining environments) that included models of unmanned vehicle behaviors. The agent-based models cited in the earlier paragraph provided only general guidance on model development: the reliance on states, parameters, and simple rules of decision-making and motion. This formed the core approach to modeling the different unmanned vehicle architectures, with a focus on the features of UAV behavior that would have the most impact on the flight deck: physical motion, task execution, and interactions with operators and other human collaborators in the world. These techniques were used to develop the models of three different UAV control architectures—gestural control (GC), vehicle-based supervisory control (VBSC), and system-based supervisory control (SBSC), which are discussed in the following sections—to determine their effects on flight deck launch performance.

B. *Gesture Control Systems*

GC systems replace a human operator with a set of stereo-vision cameras and a sophisticated computer algorithm that replicates a subset of visual tasks performed by the human operator. In practice, this means that the system must be able to detect human collaborators in the world, track their movement, then track the motion of their hands, arms, and sometimes fingers. These motions are then translated into specific gesture definitions, which are correlated with specific tasks for the vehicles (e.g., "stop" or "turn left"). The vehicle's onboard autonomy then decomposes these actions into specific low-level control commands for sensors and effectors (throttle, steering, etc.).

While gesture recognition technology is available commercially in systems, such as the XBOX Kinect, the utilization of this technology for vehicle control remains in the research and development phase. One example of this is research currently being conducted by Song *et al.* that aims to construct a GC system specifically for aircraft carrier flight deck operations [19], [20]. This carrier environment is a unique case in that commands to human pilots are already provided solely through hand gestures, providing a natural mapping to gesture control systems.

C. *Supervisory Control Systems*

Sheridan's original definition of supervisory control systems defined them as systems in which one or more human operators are commanding and receiving feedback from an autonomous or robotic system that is performing tasks in the world [21]. This paper uses the term in a stricter sense: systems in which the human operator issues commands and receives feedback through a graphical user interface that translates symbolic inputs into actions for the robotic systems, which then decompose the actions into specific low-level control inputs. This typically requires substantial computational intelligence onboard the vehicle, as well as a variety of high-fidelity sensors to observe the world (GPS, LIDAR, and other range and sensing systems), cameras, high-accuracy inertial measurement devices, along with intelligent planning algorithms to handle vehicle routing and translate operator commands into low-level control inputs.

This paper classifies these systems into two forms. The first form is referred to as VBSCin which supervisors supply commands to a single vehicle at a time, working iteratively through the queue of vehicles waiting for instruction (e.g., [22]). The operator functions as a processing server within the system, maintaining a queue of vehicles awaiting task assignments and interacting with them individually. These systems have been a subject of significant prior work, a common theme of which has been the examination of the "fan-out" problem: how many vehicles the operator can manage without significant performance degradation. Past research has shown that operators can control up to eight ground vehicles or 12 aircraft at a time [23], an acceptable number for aircraft flight deck operations in which there are no more than 8–12 vehicles taxiing at one time.

Controlling more than 8 to 12 vehicles requires the introduction of additional autonomy in the form of intelligent planning and scheduling systems. These are integrated with additional decision support displays to form the class of SBSC systems, in which the operator works with the planning algorithms to replan all tasks for all vehicles simultaneously (e.g., [24]). Typically, the operator will provide goals for the planning algorithm (e.g., weights for the objective function) and may also insert additional constraints (e.g., time windows for the execution of certain tasks). The planning algorithm generates a candidate set of assignments that is then reviewed by the operator, who has a chance to reject the plan or make modifications. Once the plan is accepted, tasks are transmitted to vehicles, which then follow the prescribed motion paths and execute the assigned tasks. The human supervisor's responsibility is to monitor the execution of the plan and create new plans in the event of failures or performance degradation.

III. Multiagent Safety and Control Simulation Model

As the unmanned vehicle systems described in Section II do not yet exist for flight deck operations, simulations of their behavior cannot be validated directly. Instead, MASCS was created first as a model of manual control (MC) flight deck operations, with the unmanned vehicle architectures defined as modifications from this model. MASCS was then partially validated against empirical data in prior work [25] because of the limitations on accessible data on flight deck operations, pilot behavior, and other factors, full validation of this simulation is not possible at this time. Constructing the model required defining agents that replicate the behavior of human crew, pilots, aircraft, planning supervisors, flight deck equipment, as well as models of the tasks that they perform. Agent and task models consist of decision-making rules that govern agent behavior, parameters that describe how agents view and execute tasks in the world, and states that are viewable by other agents and are used in the decision-making routines.

The validation process compared the performance of individual aircraft and missions of multiple aircraft with empirical data on operations. This included the average time to complete launch missions of 18 to 34 aircraft, the interdeparture times of launches within those missions, and response of the system to changes in vehicle behaviors (e.g., speed). Additionally, a team of subject matter experts (SMEs) reviewed animations of simulation execution. SMEs had a range of experience on the flight deck, each over multiple years (two pilots each with more than 800 flight hours in fighter aircraft; another individual with over 3000 h as a Naval Flight Officer and 2 years as a launch and recovery officer). All were currently employed by Navy research organizations. SMEs were allowed to review repeated executions of the simulation in order to critique the activity of agents on the flight deck. SMEs were also shown quantitative results from validation testing to provide feedback on the accuracy of the simulation in terms of mission completion times. SMEs judged that the simulation was a reasonable replication of flight deck operations and did not find any major errors in modeling that required correction. Modeling proceeded to include the unmanned vehicles reported here, which were created by modifying six key parameters of the MC aircraft models: aircraft routing methods, visual detection accuracy, command comprehension rates, catapult alignment failure rates, camera field of view, and system latencies. This section first describes the basics of flight deck operations relevant to this paper and how these operations were modeled within MASCS before discussing the modeling of the unmanned vehicle control architectures.

A. Aircraft Carrier Flight Deck Operations

A digitized map of the flight deck appears in Fig. 1. Labels in the figure highlight the important features of the flight deck environment relevant to launch operations. Aircraft begin parked at the edges of the flight deck in the aft and starboard areas of deck; aircraft not able to fit in those areas are parked in the area known as "The Street." Aircraft will be assigned to launch from one of the four launch catapults (orange lines in the figure, numbered one to four). Each catapult is capable of launching a single aircraft at a time, but adjacent pairs of aircraft (1,2 or 3,4) are not able to operate in parallel, for two reasons. The first is that a single set of crew operates each pair of catapults and can only manage one catapult at a time. The second, and more important, is that the catapults are angled toward one another—simultaneous launches would result in a collision and loss of both aircraft. Catapults alternate launches within each pair, but the two pairs can process in parallel (e.g., 1 and 3 or 2 and 4). Operations typically allow one aircraft on the catapult and another waiting behind a protective barrier that rises from the deck during a launch. As such, each pair of catapults (if both are active) can queue up to four aircraft in the area at any time, but only one aircraft will be in the process of launching.

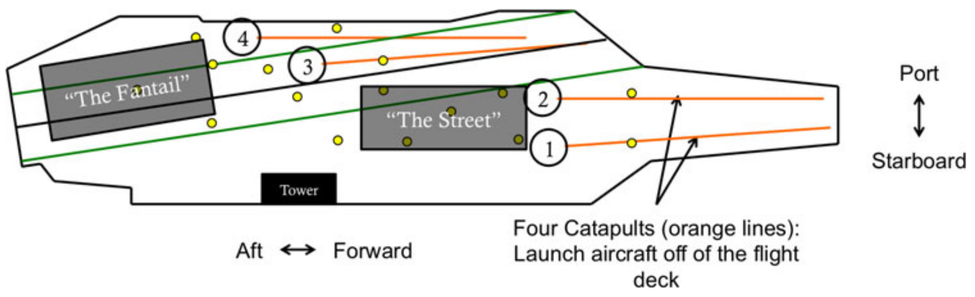
"The Fantail"
"The Street"
4
3
2
1
Tower
Port
Starboard
Aft ↔ Forward
Four Catapults (orange lines): Launch aircraft off of the flight deck

Fig. 1. Digital map of the aircraft carrier flight deck. Labels indicate important features for launch operations. Yellow dots represent Aircraft Director crew active on the flight deck.

The process of launching an aircraft requires that several tasks can be done in parallel. The aircraft must taxi forward on to the catapult, aligning its front wheel (nose gear) with the catapult centerline. The nose wheel is then physically attached to a part of the catapult known as "the pendant." While this is occurring, other crew are confirming the aircraft's current fuel level and total weight to calibrate the catapult for launch. The pilot also completes a set of control surface checks to ensure that the aircraft is ready for flight. These simultaneous tasks are modeled within MASCS as a single time distribution (N (109.65, 57.80^2)), based on prior research and discussions with SMEs.

Once these preparatory actions are complete, the catapult is triggered and the pendant begins pushing the nose gear and the aircraft forward up to flight speed. This acceleration task, also based on observations of flight deck operations, is modeled as a single time distribution (lognormal with location parameter $\mu = 1.005$ and shape parameter $\sigma = 0.0962$). Once the launch is complete, the aircraft at the adjacent catapult begins launch preparations, while the aircraft originally parked behind now taxis onto the catapult and waits for the adjacent catapult to complete its launch. Each of these tasks is independent of the type of vehicle being launched and is influenced primarily by the equipment and crew involved, rather than the control architecture.

Getting aircraft from parking places to catapults requires interactions with human crew on the flight deck. A senior enlisted person, termed the Deck Handler, and his staff are responsible for creating the schedule of operations and allocating aircraft to launch catapults. At the start of operations, assignments will be made such that queues at all catapults are filled. As launches occur and slots open at catapult queues, aircraft are dynamically allocated, assigning the next highest priority aircraft that has an open taxi route to a catapult. Aircraft assignments are then passed to a set of crew on the flight deck, termed Aircraft Directors (yellow circles in Fig. 1), who provide taxi instructions to aircraft. These crew work in a form of "zone coverage," each controlling a specific area of the deck (which may overlap with others). A Director will taxi the aircraft through their area, then hand the aircraft over to an adjacent Director before accepting another aircraft into their area. Directors communicate instructions to pilots using a set of hand gestures, informing the pilot when to drive forward, stop, turn, and to whom they are being passed off. Directors also must maintain an understanding of current traffic conditions on the flight deck, delaying some aircraft to allow other aircraft to taxi by. This may be because the other aircraft has a higher priority, or that failing to do so would "lock" the deck, with aircraft unable to taxi to their required destinations.

Models of these actors were constructed within the MASCS model, each with their own sets of decision rules to replicate the behaviors described in the previous paragraphs. Table I provides a summary list of agents modeled in MASCS and the relevant decision making rules and parameters of each. As noted previously, the goal in agent-based modeling is typically to make the decision rules as simple as possible [5]. In MASCS, this was done by building decision rules that had only a limited number of options in each decision. Most decision-making models in MASCS are binary (yes/no) and based on the current locations and states of agents on the deck. Options are often reviewed iteratively, selecting the first feasible option. If no feasible option exists, no action is taken and the system waits until conditions change.

The Deck Handler agent's scheduling rules are based on a set of assignment heuristics elicited from SMEs as part of prior work [26]. Given a list of aircraft waiting to launch, their locations, and their priorities, simple rules attempt to assign the highest priority aircraft to the nearest available catapult. Aircraft Director agents include a data structure describing to which other Directors they can pass aircraft to and rules that determine whether or not an aircraft can be passed off to the desired next Director, whether the Director can accept new aircraft themselves, as well as how to align to the aircraft they are currently instructing so that they are visible to the pilot. Aircraft and pilots are modeled within MASCS as a single unit at this time. Their models primarily concern physical motion on the flight deck and their ability to follow instructions from the Aircraft Director crew. Of greatest importance are rules that describe whether they can "see" a Director. Whether a pilot can "see" a Director is a simple check of whether the Director is in the visible area of the aircraft, as defined by a field-of-view parameter.

Aircraft Directors and Aircraft/Pilot models both require models of their motion on the flight deck. These models require both the definition of speed of travel and rules that describe the paths they should take on the flight deck. These rules were placed into separate task models that are assigned to agents for

TABLE I
LIST OF AGENTS REQUIRED BY THE MASCS SIMULATION OF FLIGHT DECK LAUNCH OPERATIONS, THE IMPORTANT FEATURES (PARAMETERS AND LOGIC STRUCTURES) INVOLVED IN THEIR MODELING, AND THE VALUES OF THOSE PARAMETERS

Agent	Feature	Description
Aircraft/Pilot	Speed	Rate at which vehicles taxi forward on the flight deck; *N* (3.5, 1.02)
	Turn Rate	Rate at which vehicles rotate during taxi operations; 18° /s
	Sight Angle	Area in front of vehicle to view Directors; 85°
Aircraft Director	Speed	Rate at which crew walk on the flight deck; *N* (3.5, 1.02)
	Alignment logic	Determines when and to what location Directors should align to aircraft
Deck Handler	Routing Logic	When and to what Director the vehicle should next be sent
	Traffic Management logic	How to deconflict taxi paths on the flight deck
	Routing Logic	Determines whether or not a taxi route exists between an aircraft and a catapult. Defined by heuristics that examine whether aircraft are parked in key places on flight deck.
	Planning Logic	Given rules of routing logic, assigns highest priority aircraft to nearest available catapult. Prioritize sending aircraft to catapults with no aircraft in queue.
Catapults	Launch Constraints	How launches alternate between catapults and when obstructions in the nearby area should stop a launch in progress
	Queue Structure	List of aircraft currently assigned to this catapult (maximum 2) and how aircraft are promoted in the queue

execution. These task models determine, for a given agent with an assigned destination, what combination of straight line and turn actions is required. During execution, the models continually check the conditions of the deck to make sure that the transit path is clear. For an aircraft taxiing on the flight deck, the "Taxi" task model continually checks 1) whether this action should be delayed due to traffic conditions and 2) if another aircraft is physically blocking the current vehicle, as well as 3) if the aircraft can "see" the Director. If any of these conditions is true, the motion is paused until conditions permit movement. All aircraft and crew execute such rules simultaneously during their operations.

Modeling the UAV control architectures described in Section II within MASCS requires changes to the models described above. The general planning methods do not differ substantially, nor do the models of physical motion. However, the rules governing task execution and taxi routing for MC operations may not apply to unmanned vehicles due to the differences in control architecture capabilities. Other parameters describing the vehicles' physical characteristics might also vary from the baseline MC model. The differences that define the unmanned vehicle control architectures used in this paper are discussed in the next section.

B. Models of Unmanned Vehicle Behavior

In the context of flight deck operations and the models of agent behavior, the important activities of a pilot that must be replicated by an unmanned vehicle control architecture involve three key aspects: visual detection of an Aircraft Director that is attempting to provide instructions, correctly interpreting task instructions, and correctly executing the commanded tasks. Each of these corresponds to a related parameter: the probability of observing the Director if within the vehicle's field of view, the probability of correctly interpreting the command, and the probability of correctly executing a task. For the baseline MC model of current flight deck operations described in the previous section, the likelihoods of these failures are so low that SMEs consider them nonexistent (0% likelihoods). A fourth difference in behavior comes from the fact that the Director is providing instructions in this "zone coverage" routing; this may not occur for some forms of UAV control, and thus how vehicles are routed forms a fourth behavioral parameter of "routing method."

Two other parameters relate to the physical characteristics of design of the vehicle: the field of view of the pilot/aircraft, which describes where the Director should be in order to provide commands, and the latency in processing commands and executing tasks. The field of view for manual operations is related to human sight range and the constraints of being in the cockpit; for unmanned vehicles, field of view is related to the cameras used on the vehicle. The latency in processing commands could come from a number of sources: delays in the communication of signals between operator and vehicle, the time required for the vehicle's computer system to process, or time required for the human operator to interact with the system. Each of these can also be modeled as variables within the MASCS simulation environment, and a full list of these six design parameters appears below. The following paragraphs describe the settings and variations of these six performance parameters in the context of the different vehicle control architectures.

1) Vehicle behavior parameters.
   a) How aircraft are *routed* on the flight deck (with or without the Aircraft Director crew).
   b) *Visual detection:* the probability of failing to detect an operator when they are within the field of view.
   c) *Task comprehension:* the probability of failing to comprehend an issued task.
   d) *Catapult alignment (task) execution:* the probability of failing to properly align to a catapult.
2) System Design parameters.
   a) The *field of view* of the pilot/autonomous vehicle, which affects how Aircraft Directors align to aircraft during operations.
   b) *Latency and lag* present in the system due to communications latencies, processing times, or limitations of the human operator.

1) Unmanned Aerial Vehicle Routing Methods: The first major difference between UAV Control Architectures involves how they are issued taxi commands and routed through the flight deck. GC systems interact with the crew on the flight deck in a similar fashion as MC vehicles, which requires modeling the three remaining vehicle behavior parameters in the list above. The two supervisory control systems leverage their centralized network to perform tasks in a different manner. VBSC systems would rely on commands issued from the Handler through a graphical user interface, which requires modeling the behavior of the Handler in operating such a system. In this form of system, the operator functions as a queuing server, which requires the definition of both a service time and a server policy for managing tasks. For SBSC systems, the operator interacts periodically with a planning algorithm that replans all tasks for all vehicles at once. Once the plan is updated, the aircraft are allowed to execute tasks until such a time as a new plan is required.

The planning heuristics used currently by experienced planners aim to keep traffic moving on the flight deck in an orderly fashion, minimizing the occurrence of routing conflicts or other blockages that limit aircraft taxi motions [26]. To ensure this continuity in VBSC systems, experienced Handlers would likely operate in a “highest priority first” fashion, prioritizing aircraft nearest to their assigned catapults in order to keep traffic flowing on the deck. The time required to process tasks is taken from prior research by Nehme [17], whose research involved operators controlling a team of unmanned vehicles through a VBSC-style interface, requiring them to define waypoints that defined vehicle transit paths, among other task assignments. Nehme characterized the “idle” task, in which operators assigned a new transit task to a waiting vehicle, as normally distributed with a mean of 3.19 s and standard deviation of 7.19 s (bounded [0, inf]).

The interaction of the Handler’s interaction time with each vehicle and the current number of vehicles in the queue also generates latencies for VBSC systems: not only must a VBSC vehicle wait N (3.19, 7.19^2) seconds at the start of each of its own tasks, it must wait for all other vehicles ahead of it in the queue to be processed. If four vehicles are ahead in the queue, the fifth vehicle must wait (on average) $3.19 * 5 = 15.95$ s before starting its next task. Thus, there is a significant penalty for including additional vehicles in the VBSC queueing system, which has been characterized in many previous studies (see [22], [23] for examples).

Additionally, since this style of routing does not rely on the network of crew on the flight deck, the Handler can also assign longer taxi tasks on the flight deck—taxiing from aft to forward, barring other constraints, could be done in a single motion rather than requiring multiple handoffs between Directors. The taxi paths, however, would largely stay the same. These same routing changes also would occur for the SBSC architecture, which enable a single operator, with the help of a scheduling algorithm, to simultaneously define task lists for all vehicles. In the SBSC architecture, once assignments are made, just as with VBSC vehicles, tasks are carried out without the assistance of the Aircraft Directors in taxi operations. Prior work in this area [26] demonstrated that even complex planning algorithms have trouble decomposing the geometric constraints on the flight deck and provide little benefit over human planning. The appropriate development and testing of planning algorithms for the flight deck lies outside the scope of this research; for SBSC systems, the same Handler planning heuristics applied to the other control architectures (CAs), which performed as well or better than other planning algorithms in prior work [26], are applied here.

2) Visual Detection, Task Comprehension, and Task Execution: For the Gesture control systems that rely on visual information, the processes of visual detection of Aircraft Directors, comprehension of the commanded tasks, and the execution of those tasks must also be modeled. There is a chance of failure in each individual recognition task, most importantly in recognizing and the Director and identifying the commands they are providing. As these systems are still largely in the research and development phase, there is no true failure rate known for gesture control activity in the flight deck environment. A review of several papers [20], [27]–[33], most of which incorporated data for multiple tests and trials and references multiple prior studies, provided a picture of the current general state of gesture recognition research. Out of the 124 different experimental results reported, eight report accuracies of 99% or better (all from a dataset using only ten actions and a simple, homogeneous background), 32 reported results better than 95%, and 77 reported better than 90% accuracy. Overall, the studies report an average failure rate of 11.16%. It is not clear how this failure rate translates into realistic future operations; while the technology will certainly improve, a variety of environmental conditions (sun glare, obstructions, large numbers of moving vehicles) will limit performance in the real world. While future real-world failure rates are still not known, the median failure rate of 11.16% observed in the research is used as the failure rate of the baseline GC system.

A logical implementation of GC could imply that a failure to recognize a Director might only happen once, after which the Director would have the authority to force control of the vehicle. If a vehicle fails to recognize a *task*, however, there may not be a simple method to force the system to recognize that specific task. It may take several attempts for the vehicle to register the correct command, with each failure accruing a time penalty. For the baseline MASCS model of GC, the penalty for failing to acquire a new Director is set at 5 s, representing the time required to realize the vehicle has not recognized the Director and for the Director to issue an override. The time penalty for failing to acquire task is modeled as a uniform distribution between 5 and 10 s, representing the time for the operator to stop, issue a cancel command, and wait for the system to become ready to accept new instructions. However, the original latency in processing is not applied to these failures. The number of failures in command comprehension is determined at the start of each action commanded by a Director by a Bernoulli trial generator, after which the total time penalty is solved by randomly sampling the uniform interval [5], [10] once for each failure incurred.

3) Latencies: One form of latency in the control architectures occurs for the VBSC systems, due to the way in which the operator manages the queue of waiting vehicles (described previously in Section III-B1). GC systems will also experience a form of latency due to software processing, as the system attempts to translate the hand gestures of the Director into understandable

TABLE II
COMPARISON OF UAV CONTROL ARCHITECTURE DESIGN PARAMETERS DEFINED FOR MASCS

	Manual Control	Gesture Control	Vehicle-Based Supervisory Control	System-Based Supervisory Control
Routing Method	Zone coverage	Zone coverage	Centralized control	Centralized control
Visual detection rate	P(fail) = 0	P(fail) = 0.1118	N/A (central control)	N/A (central control)
Task Comprehension Rate	P(fail) = 0	P(fail) = 0.1118	N/A (central control)	N/A (central control)
Task Execution Rate	P(fail) = 0	P(fail) = 0.05	P(fail) = 0.025	P(fail) = 0.025
Field of View	85°	65°	N/A (central control)	N/A (central control)
Lag/Latency	None	0–3 s (computation)	N (3.19, 7.192) per task	None

action commands. The prior research reviewed for error rates [20], [27]–[33] also shows this requires anywhere from <<1 to 3 s; this is modeled as a uniform distribution over the range [0, 3], randomly sampled and applied to the start of every action commanded by a Director.

C. Performance Comparisons

Table II provides a comparison of the four UAV control architectures to the baseline MC model in terms of the six primary design parameters described earlier. For the MC model, SMEs report that failures on these tasks are so rare that they do not really recognize them as occurring in the system: pilots rarely fail to see a Director if they are within their field of view, rarely fail to interpret a command, and rarely fail to properly align to a catapult. The field of view parameter was calibrated in prior work to a value of 85°. These settings were utilized in the model and partially validated in prior work [25] and are considered to be representative of the highly skilled pilots that work within the aircraft carrier flight deck.

Each of the unmanned vehicle control architectures defined in previous sections can be modeled in MASCS by changes to the six design parameters listed in Table II. As compared with a baseline MC vehicle, GC systems will still operate under the crew's "zone coverage" operating paradigm, but will introduce both a small to moderate processing latency at the start of each action and a chance to fail to recognize either the person issuing the command or the command itself (these are akin to another small delay at the start of operations). As noted in Sections III-B2 and III-B3, this latency ranges from 0–3 s and the average failure rates are slightly over 10%. The processing latency is randomly sampled over the defined interval at the start of the attempted task. Task failures can occur repeatedly, with each failure resulting in a time penalty between 5 and 10 s to resubmit commands to a GC vehicle. The time penalties are also sampled randomly over the defined interval. The use of cameras by the gesture recognition system might also constrain the applicable field of view for these vehicles, and the lack of a practiced pilot in aligning the aircraft to the catapult at the end of taxi operations will likely increase the rate of error on this task. However, there is no current evidence to suggest what this failure rate may be in real life. A catapult alignment failure rate of 5%, translating to roughly one failure per mission (once launch of a 22 aircraft sortie), was selected for the GC vehicles based on SME opinions.

As VBSC and SBSC systems are centrally controlled, they receive no instructions from the crew. Thus, the visual detection and task comprehension rates are not applicable for these two systems, nor is the modeling of a field of view. Both are expected to at least have a nonzero failure rate in aligning to aircraft catapults, set at once per two missions. SBSC systems have no latency, as vehicles are sufficiently autonomous to execute the series of tasks provided to them quickly and without human interaction and any replanning occurs in parallel to vehicle operations. VBSC systems, however, rely on a human operator to supply tasks individually to aircraft, which must wait in a queue for further instructions. This introduces latency in the form of a wait time for operator interaction, which for large numbers of aircraft may lead to significant delays in task execution.

The differences in performance of individual vehicles under these control architectures should thus be a function of the cumulative delays and latencies experienced in task recognition and execution. The GC and VBSC systems both experience a greater level of latencies and delays than MC and SBSC systems and should thus perform more poorly. For GC systems, these arise from the chances of repeated errors and processing delays on the part of the vehicles, while for VBSC they are due to the need for repeated interactions with the human supervisors. However, the impact of these delays and latencies when multiple aircraft are included in flight deck operations will be influenced by the structure of modern aircraft carrier flight deck operations. The model of operations constructed for this paper was partially validated in [25] and is discussed in the next section.

IV. EFFECTS ON OPERATIONS ON UNMANNED AERIAL VEHICLE PERFORMANCE

The models of unmanned vehicle control architectures described in the previous section were implemented into the MASCS environment and verified through a set of tests that examined the behavior of an individual aircraft on the flight deck. The same mission scenario was executed for each of the four control architectures (MC, GC, VBSC, and SBSC) with 300 replications performed in each case. A diagram of this task appears in Fig. 2: the aircraft begins parked in the aftarea of the flight deck and is assigned to launch from catapult 3, requiring a series of straight-line and turning motions before culminating with the launch preparation and launch acceleration tasks. This same scenario was used previously in the development and testing of the baseline MC model [25]. The results of these simulations appear in Fig. 3, which shows the average time for a single aircraft to complete the launch mission (from taxi through launch acceleration) for each the four control architectures, with whiskers indicating ± 1 standard error in the data.

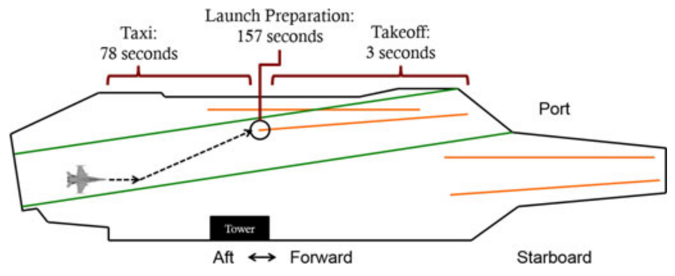

Fig. 2. Diagram of mission executed in the single aircraft performance comparisons.

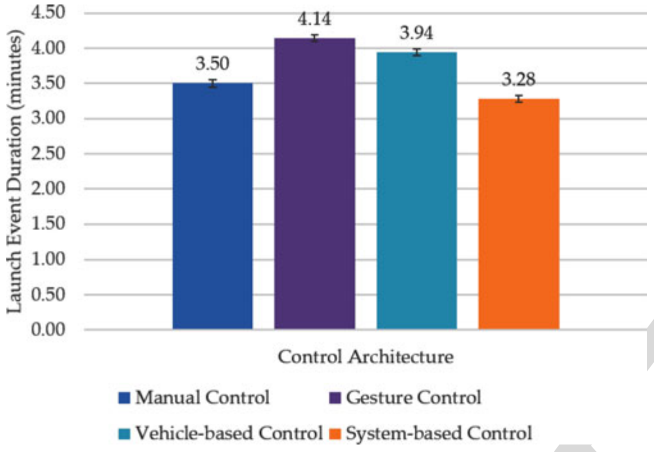

Fig. 3. Values for launch event duration (LD) for single aircraft testing for manual control (MC), gestural control (GC), vehicle-based supervisory control (VBSC), and system-based supervisory control (SBSC). Bars indicate mean LD for sets of 300 replications, with whiskers showing ±1 standard error.

In general, the responses of four control architectures match the expectations described earlier: the GC and vehicle-based supervisory control (VBSC) cases being the slowest and MC and SBSC being the fastest. An analysis of variance (ANOVA) test demonstrates that the differences between the MC, GC, SBSC, and VBSC cases are significant at the $\alpha = 0.05$ level ($F(3, 1196) = 66.07, p < 0.0001$), and a Tukey test reveals that all pairwise comparisons are significant. In terms of practical differences, GC vehicles take 0.64 min longer to complete the task, VBSC 0.44 min, and SBSC are 0.22 min faster. While these are indeed small differences, percentagewise these translate into percentage differences of 18.4%, 12.5%, and −6.3%, respectively, from the MC baseline.

However, the structure of flight deck operations may affect how these variations in individual vehicle performance are felt in missions using many aircraft. Flight deck operations, as well as airport operations, are such that aircraft depart an initial parking area and are sent to queue at a specific launch site. In terms of system performance, as long as another aircraft arrives at the site prior to the launch of the previous aircraft, the flight deck (or airport) is working effectively. In order for the differences in the individual control architectures to be observed at the mission level, the delays and latencies that affect GC and VBSC performance must disrupt taxi operations upstream of the catapult sufficiently to break the chain of queuing that occurs.

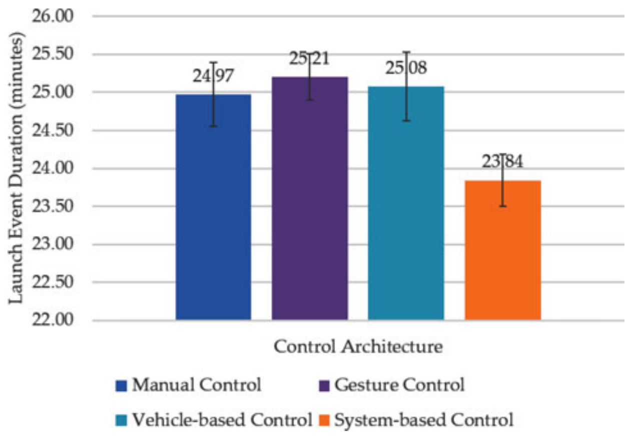

Fig. 4. Results for Launch event Duration (LD) for each control architecture for missions using 22 aircraft under the original launch preparation time model. Whiskers indicate ±1 standard error.

To test whether or not this occurs, the four control architectures were tested in missions using 22 aircraft. In these missions, all 22 aircraft begin parked at the edges of the flight deck. The Deck Handler agent then dynamically assigns aircraft in real time based on current aircraft priority and whether nearby catapults are open and accessible to the aircraft. Each mission utilized the same initial conditions (vehicle parking locations and priorities) and used a completely homogeneous vehicle set (all MC, or all Gesture Control, etc.). Thirty replications of each mission were performed and the total time to complete the mission (from initialization to completion of the last launch) was logged for each. Fig. 4 provides a column chart of the average time to complete missions for each control architecture, with whiskers indicating ±1 standard error in each dataset.

The data in the figure suggests no real variations occur in system performance, with an ANOVA test returning marginally significant results ($F(3, 116) = 2.57, p = 0.058$). This suggests that the effects of GC and VBSC system delays are not sufficiently disruptive to cause differences at the mission level, nor are the benefits of the SBSC system, which does not rely on the crew "zone coverage" routing, sufficient to improve performance at a highly significant level.

One interpretation of these results is that an error exists either within the aircraft models, the routing of aircraft to catapults, or the processing of tasks at the catapults. However, the individual models of aircraft (see Fig. 3) showed that differences in performance do exist on an individual vehicle level; some mechanism at the mission level is preventing these differences in individual performance from having an aggregate effect. One possible explanation regards how aircraft queue at launch catapults, as described in Section III-A: each catapult can allow one aircraft onto the catapult and another to queue behind. At a catapult pair, only one aircraft is actively preparing for launch at any given time. Once this aircraft launches, the aircraft parked behind it moves forward onto the catapult and the aircraft sitting on the paired catapult begins preparing for launch. A new aircraft is then taxied to the empty place in the catapult queue.

The actions of the Deck Handler create this queueing pattern immediately and keep it in existence throughout the mission.

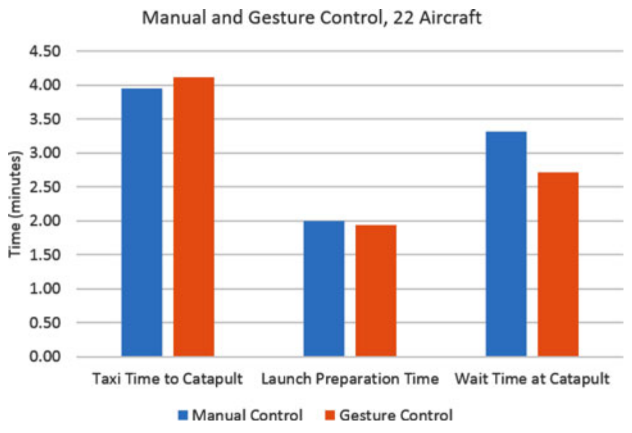

Fig. 5. Column chart of breakdown of tasks in a 22 aircraft mission using manual control (MC) or gesture control (GC).

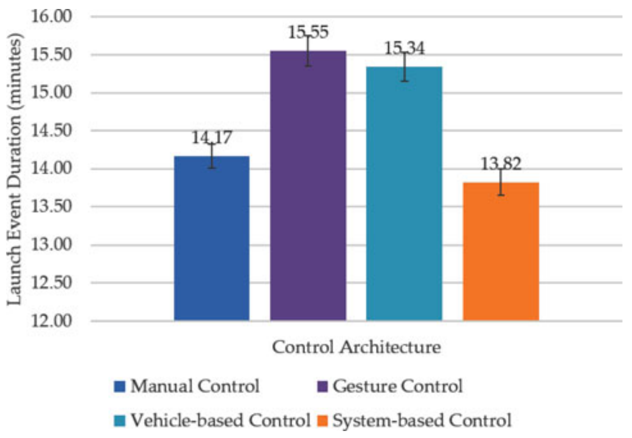

Fig. 6. Results for launch event duration (LD) for each control architecture for missions using 22 aircraft under the automated launch preparation time model. Whiskers indicate ± 1 standard error.

With this queueing pattern in place, new aircraft arrive at catapults only to wait a period of time in the queue. If all control architectures are efficient enough in taxi operations to supply aircraft to catapults before the queues empty, then differences in launch event durations across control architectures at the mission level will not be observed. Emptying a queue means that a catapult pair would have to process all three prior aircraft in the time it takes the new (fourth) aircraft to reach the queue. Additional output data from the MASCS simulation suggests that this does not occur and the queues remain in place throughout the mission (see Fig. 5). In a 22 aircraft mission using only MC aircraft, the average time for an aircraft to taxi to a catapult is just under 4 min. At an average launch time of 2 min, the queue would be cleared in 6 min, greater than the average taxi time. This implies that aircraft should be encountering queues at catapults, and data demonstrates that they are, at an average wait time of 3.32 min. The figure also includes the same data for Gesture Control, the worst performer in the single aircraft testing described earlier, which provides similar results for taxi times, launch times, and wait times.

These data suggest that the manner in which aircraft are queued at catapults, combined with the current launch preparation process, serves to bottleneck operations. Reducing the time required to complete the launch preparation task should provide conditions in which queues are eliminated, allowing differences to appear between control architectures at the mission level. The next section examines how decreasing the mean and variance of the launch preparation task affects the performance of the different UAV control architectures.

A. *Effects of Launch Preparation Time*

The launch preparation task is a highly manual process executed by several members of the deck crew, a key component of which is the physical attachment of the aircraft nosegear to the catapult. The pilot must first correctly align the aircraft to the catapult before a crew member connects a piece of the catapult mechanism to the aircraft nosegear; after this occurs, the catapult pulls the aircraft forward to test the connection. Other tasks performed by other crew involve information processing, setting the strength of the catapult based on the current weight of the aircraft. Since all vehicles connect to the same catapult mechanism and perform the same checks, the time to complete this task is defined independently of the types of aircraft being connected. Additionally, because these tasks occur largely in parallel, the entire process was modeled as a single normal distribution with MASCS. Decreasing the mean value of this process could potentially be accomplished through better training of the crew, while decreasing the variance might be done by automating certain aspects of the task and removing the more variable human components of the task.

A new series of tests were conducted using a "reduced" launch preparation time with significantly reduced mean and variance. In this reduced model, the mean was reduced by half and the variance by three-quarters (to $N\,(54.82,\ 28.9^2)$). All four control architectures were tested under this launch preparation time model using 22 aircraft. Thirty replications were again performed for each case, and the results of these tests appear in Fig. 6 with whiskers denoting ± 1 standard error.

At these settings ($\mu = 54.82$ s, $\sigma = 28.9$ s), an ANOVA reveals that significant differences do exist within the data ($F(3, 116) = 18.63, p < 0.0001$). A Tukey test reveals significant differences between two pairs of control architectures: GC ($\mu = 15.55$) and VBSC ($\mu = 15.34$) are both significantly different from MC ($\mu = 14.17$) and SBSC ($\mu = 13.82$). However, no significant differences exist within each pair (e.g., GC and VBSC). In terms of practical effects, the difference between the averages of the fastest (SBSC) and slowest (GC) control architectures is 1.73 min, equivalent to 12.2% of the MC baseline.

That variations in control architecture performance do arise with this automated launch preparation time model suggests that the characteristics of the launch preparation time do have significant effects on operations. The original launch preparation time, with its high mean and variance, serves to bottleneck flight deck operations; regardless of what is occurring, aircraft must always queue behind other aircraft at the catapults. While this makes the system robust to variations in unmanned vehicle behavior (the delays that occur in GC and VBSC systems), it also means that beneficial effects of unmanned vehicle introduction are also obscured. Improving operations on the flight deck would first require improving the performance of this vehicle-independent processing task, rather than improving the performance of individual vehicles.

These results are also important for other similar domains, such as future airport operations. Given the first set of results under the original launch preparation model, the introduction of unmanned vehicle systems provided no benefit whatsoever to system operations due to the settings of the vehicle-independent launch preparation task. Given this data, a stakeholder would likely decide that UAV integration provides no tangible benefits to operations. They might also decide that because there are no differences, all types of unmanned vehicles provide the same level of performance in the system; the choice might then be made based on other factors. If this occurs, and the system moves to a faster tempo of operation, this decision might have severe consequences. In aircraft carrier operations, tempo increases during wartime combat operations; in airports, increasing the tempo could help relieve the congestion common at many U.S. airports. At this increased tempo setting, if stakeholders have already locked in to using a GC or VBSC system based on the previous data, operations will not perform to the level that was originally expected.

The use of simulations like MASCS, as presented here, can help to clarify what differences can be expected from these systems in the real world and how their performance might be influenced by other process-level tasks independent of the UAV system control architecture itself. They can also help to explore under what conditions performance of these systems could be improved. While the results shown above demonstrated that the GC architecture performed poorly under the "reduced" launch preparation time model, the GC system also had the most changes from the MC baseline, varying on five of the six design parameters listed in Section III-C. Improvements along some combination of these parameters should be sufficient to provide GC performance equivalent to that of the baseline MC architecture. The next section demonstrates how simulations like MASCS can be used to examine the effects of changing vehicle performance parameters for the control architecture models in an example design space exploration.

B. Design Space Exploration

In future systems, stakeholders may seek to understand what changes to unmanned system control architectures are required to improve the performance under certain system conditions. This section examines how this might occur for the case of the GC architecture, which previous results showed as having the worst performance of the modeled architectures. The most important changes in the Gesture Control architecture involve the increased failure rates of these vehicles in recognizing Aircraft Directors and their commands and the processing lag in recognizing those commands. The failures also carry with them time penalties that delay vehicle activity on the flight deck and that, as demonstrated, can sometimes be detrimental to flight deck performance. Improving the accuracy of these systems and reducing their processing times should improve the performance

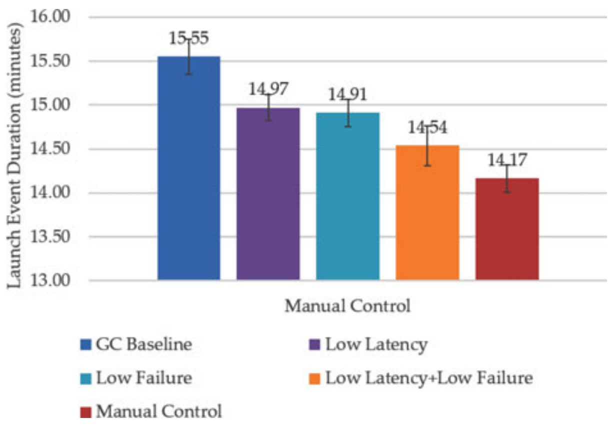

Fig. 7. Results for launch event duration (LD) for variations of 22 aircraft Gestural Control (GC) model using the "reduced" (mean-50%, standard deviation-50%) launch preparation time distribution. Whiskers indicate ± 1 standard error.

of the GC system, but it is not known which of these carries more weight in terms of improving mission performance.

Three new GC models are tested in this section over a variety of mission conditions. First, a "Low Failure" case is created in which the failure rate is reduced from 11.16% to 1%, modeling a "best-case scenario" of system accuracy. Second, a "Low Latency" is created in which commands are processed at a maximum of 0.5 s. A third model combines the two cases, generating a Low Latency + Low Failure condition that should show improvement in performance. A first series of tests addressed the effects of these models in missions involving 22 aircraft under the original launch preparation model ($N\,(109.65, 57.80^2)$), but no significant differences were observed (ANOVA, $F(4, 145) = 0.51, p = 0.73$). This is to be expected, given the effectiveness of the original launch preparation time distribution in obscuring variations between control architectures.

A second round of testing then proceeded to the "reduced" launch preparation model ($N\,(54.82, 28.92)$), reducing both the mean and standard deviation by 50%. Thirty replications of the same 22 aircraft mission used previously were conducted for each of the three new GC models (Low Failure, Low Latency, Low Latency + Low Failure). Given the results of the previous section, the variations in performance were expected to appear at this faster launch preparation time model. The results of these tests appear in Fig. 7 and include results for the baseline GC and MC models for comparison.

The figure shows that the data exhibits a noticeable trend from the baseline GC model to Low Latency, then Low Failure, then Low Latency + Failure, then the baseline MC model. An ANOVA shows that significant differences do exist between the five test cases ($F(4, 145) = 8.12, p = 0.0001$). A Tukey post hoc test shows that, most importantly, the GC Low Latency + Low Failure and MC cases are not significantly different from one another ($p = 0.60$). The MC data is significantly different from the GC Baseline ($p < 0.0001$), Low Failure ($p = 0.017$), and Low Latency ($p = 0.034$) cases. The Low Latency + Low Failure case is also significantly different from the GC baseline model ($p = 0.0012$), but is not different from the Low Failure ($p = 0.17$) and Low Latency ($p = 0.10$) cases. This is fairly interesting and highlights the significance of the earlier issues with the baseline GC models: the GC architecture requires substantial improvements to *both* the gesture recognition accuracy *and* processing speeds to achieve performance equivalent to MC operations. Improvements to either of these elements individually were not strong enough to generate significant improvements in GC performance.

V. Conclusion

This paper has described the use of an agent-based simulation model of aircraft carrier flight deck operations that allowed for a systems-level analysis of the key features of unmanned vehicle behavior that influence aircraft carrier flight deck performance. Testing demonstrated that the launch preparation process, whose execution times are independent of any specific unmanned vehicle control architecture, heavily influenced system behavior and effectively obscured differences between control architectures. Differences in control architecture performance were only observed under a task model with significantly reduced mean and variance. However, this launch preparation process is not slated for improvement in the near future. Ultimately, these results suggest that failure to reexamine the structure of flight deck operations and the system as whole will ultimately limit the effectiveness of introducing autonomy into aircraft on the flight deck.

Additional testing examined how, under such a reduced launch preparation time, the simulation could be used to investigate the effects of modifying control architecture parameters. These results showed that Gesture Control systems required improvements to *both* accuracy rates and processing times in order to achieve performance equivalent to manual control operations in a 22 aircraft mission. Overall, these results suggest that the structure and definition of parameters within the broader system (the flight deck and tasks defined independently of the control architectures) has a significant impact on the behavior of the control architectures modeled in this paper. For stakeholders making decisions regarding which types of control architectures to implement in future operations, for environments with similar structure to the carrier flight deck, the definition of tasks like the launch preparation time (the final task performed by vehicles before exiting the system) greatly affects control architecture performance.

The focus of MASCS at this time has been a general examination of how different unmanned vehicle control architectures affect operations on the aircraft carrier flight deck, using models of future UAV control architectures that have reached some level of maturity in operations. A significant limitation of this modeling and this research has been the exclusion of rare and safety-critical events from the system. These are an important aspect of operations in any complex system, and models of rare events, failures, and other issues could be included in the modeling of the physical environment (weather conditions, damage to the ship and flight deck), the vehicle models (mechanical or electrical failures, etc.) or in the modeling of the human actors (poor decision making, random accidents, etc.). These are all important areas of future work that can be addressed in future research, but until formally validated, MASCS cannot be considered a true predictor of real-world system performance.

Ultimately, as unmanned systems move from operating in isolated environments to more complex human-centered ones, understanding the drivers of performance for both unmanned vehicles and the systems in which they operate is key. This research has demonstrated how this can be achieved through the use of agent-based modeling of unmanned vehicle systems and provided results that suggest that, for future aircraft carrier flight deck and airport operations, improving aspects of the system itself are as important, if not more so, than UAV improving the performance of the UAV systems being introduced. Future work will address how these control architectures interact with changes to the operational structure, examining how changes to the nature of planning and routing of aircraft tasks in the flight deck environment interact with other elements to drive variations not only in mission durations, but in terms of safety and efficiency as well.

References

[1] R. Tinto, “Mine of the future,” 2011. Q1
[2] A. Topf, “Rio Tinto boosts driverless truck fleet for use in Pilbara,” 2011.
[3] J. Scanlon, “How KIVA robots help Zappos and Walgreens,” 2009.
[4] S. Nikolaidis and J. Shah, “Human-Robot interactive planning using cross-training-a human team training approach,” presented at the AIAA Infotech@Aerospace, Garden Grove, CA, 2012.
[5] E. Bonabeau, “Agent-based modeling: Methods and techniques for simulating human systems,” *Proc. Nat. Academy Sci. United States Amer.*, vol. 99, no. Suppl. 3, pp. 7280–7287, 2002.
[6] B. McKinney, “Unmanned combat air system carrier demonstration (UCAS-D),” 2011.
[7] W. J. Hennigan, “Navy drone X-47B lands on carrier deck in historic first,” Jul. 2013.
[8] “Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap,” Tech. Rep., Federal Aviation Admin., Washington, DC, USA, 2013. Q2
[9] K. Sycara and M. Lewis, “Agent-based approaches to dynamic team simulation,” Tech. Rep. NPRST-TN-08-9, Navy Personnel Research, Studies, and Technology Division, Bureau of Naval Personnel, Millington, TN, USA, 2008.
[10] K. Carley, D. Fridsma, E. Casman, N. Altman, J. Chang, B. Kaminsky, D. Nave, and A. Yahja, “BioWar: Scalable multi-agent social and epidemiological simulation of bioterrorism events,” *IEEE Trans. Syst., Man Cybern. Part A, Syst. Humans*, vol. 36, no. 2, pp. 252–265, 2006.
[11] S. M. Lee, A. R. Pritchett, and K. M. Corker, “Evaluating transformations of the air transportation system through agent-based modeling and simulation,” in *Proc. 7th FAA/Eurcontrol Seminar Air Traffic Manage. Res. Develop.*, 2007. Q3
[12] J. M. Epstein and R. M. Axtell, *Growing Artificial Societies: Social Science From the Bottom Up*. Washington, DC, USA: Brookings Institution, 1996.
[13] S. J. Rasmussen and P. R. Chandler, “MultiUAV: A Multiple UAV Simulation for Investigation of Cooperative Control,” presented at the Winter Simulation Conf., San Diego, CA, USA, 2002.
[14] J. J. Corner and G. B. Lamont, “Parallel simulation of UAV swarm scenarios,” presented at the Winter Simulation Conf., Washington, DC, USA, 2004.
[15] R. E. Weibel and R. J. Hansman Jr., “Safety considerations for operation of different classes of UAVs in the NAS,” in presented at the AIAA’s 4th Aviation Technol., Integration, Operations Forum, Chicago, IL, USA, 2004.
[16] B. Schumann, J. Scanlan, and K. Takeda, “Evaluating design decisions in real-time using operations modelling,” presented at the Air Transport Operations Symp., Delft, The Netherlands, 2011.
[17] C. Nehme, “Modeling human supervisory control in heterogeneous unmanned vehicle systems,” Ph.D. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.

[18] A. A. Mkrtchyan, "Modeling operator performance in low task load supervisory domains," M.S. thesis, Massachusetts Inst of Technol., Cambridge, MA, USA, 2011.
[19] Y. Song, D. Demirdjian, and R. Davis, "Multi-Signal gesture recognition using temporal smoothing hidden conditional random fields," presented at the 9th IEEE Conf. Automat. Face Gesture Recog., Santa Barbara, CA, USA, 2011.
[20] Y. Song, D. Demirdjian, and R. Davis, "Continuous body and hand gesture recognition for natural human-computer interaction," *ACM Trans. Interactive Intell. Syst.*, vol. 2, no. 1, article 5, 2012.
[21] T. B. Sheridan, Telerobotics, *Automation and Human Supervisory Control*. Cambridge, MA, USA: The MIT Press, 1992.
[22] M. L. Cummings and S. Guerlain, "Developing operator capacity estimates for supervisory control of autonomous vehicles," *Human Factors*, vol. 49, no. 1, pp. 1–15, 2007.
[23] M. L. Cummings and P. J. Mitchell, "Predicting controller capacity in remote supervision of multiple unmanned vehicles," *IEEE Trans. Syst., Man, Cybern. Part A, Syst. Humans*, vol. 38, no. 2, pp. 451–460, 2008. Q4
[24] M. L. Cummings, J. How, A. Whitten, and O. Toupet, "The impact of human-automation collaboration in decentralized multiple unmanned vehicle control," *Proc. IEEE*, vol. 100, no. 3, pp. 660–671, Mar. 2012.
[25] J. C. Ryan and M. L. Cummings, "Development of an agent-based model for aircraft carrier flight deck operations," *Model. Simulation J.*, 2014. Q5
[26] J. C. Ryan, A. G. Banerjee, M. L. Cummings, and N. Roy, "Comparing the performance of expert user heuristics and an integer linear program in aircraft carrier deck operations," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 761–773, Jun. 2014.
[27] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Action as Space-Time Shapes," presented at the 10th IEEE Int. Conf. Comput. Vision, Beijing, China, 2005.
[28] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape- motion prototypes," presented at the 12th Int. Conf. Comput. Vision, Kyoto, Japan, 2009.
[29] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," presented at the IEEE 11th Int. Conf. Comput. Vision, Rio de Janeiro, Brazil, 2007.
[30] P. Scovanner, S. Ali, and M. Shah, "A 3-Dimensional SIFT Descriptor and its application to action recognition," presented at the ACM Multimedia, Augsburg, Bavaria, Germany, 2007.
[31] I. Laptev, M. Marszalek, C. Schimd, and B. Rozenfeld, "Learning realistic human action from movies," presented at the IEEE Conf. Comput. Vision Pattern Recog., Anchorage, AK, USA, 2008.
[32] K. Schindler and L. van Gool, "Action Snippets: How many frames does human action recognition require?," presented at the IEEE Conf. Comput. Vision Pattern Recog., Anchorage, AK, USA, 2008.
[33] J. Yuan, Z. Liu, and Y. Wu, "Discriminative subvolume search for efficient action detection," presented at the IEEE Conf. Comput. Vision Pattern Recog., Miami, FL, USA, 2009.

Jason C. Ryan (M'12) received the Bachelor's of Science degree in aerospace engineering from the University of Alabama, Tuscaloosa, AL, USA, in 2007, the Master's of Science degree in aeronautics and astronautics and the Ph.D. degree in human systems engineering both from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2011, and 2014, respectively.

He is currently with Aurora Flight Sciences.

Mary L. Cummings (SM'03) received the B.S. degree in mathematics from the United States Naval Academy, Annapolis, MD, USA, in 1988, the M.S. degree in space systems engineering from the Naval Postgraduate School, Monterey, CA, USA, in 1994, and the Ph.D. degree in systems engineering from the University of Virginia, Charlottesville, VA, USA, in 2004.

She is currently a Visiting Professor with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology and an Associate Professor in the Department of Mechanical Engineering and Materials Science at Duke University, Durham, NC, USA.

QUERIES

Q1. Author: Please provide full bibliographic details in Refs. [1], [2], [3], [6] and [7].
Q2. Author: Please provide report number in Ref. [8].
Q3. Author: Please provide page range in Ref. [11].
Q4. Author: Please provide month in Ref. [23].
Q5. Author: Please provide volume and page range in Ref. [25].

A Systems Analysis of the Introduction of Unmanned Aircraft Into Aircraft Carrier Operations

Jason C. Ryan and Mary L. Cummings

***Abstract*—Recent advances in unmanned and autonomous vehicle technology are accelerating the push to integrate these vehicles into environments, such as the National Airspace System, the national highway system, and many manufacturing environments. These environments will require close collaboration between humans and vehicles, and their large scales mean that real-world field trials may be difficult to execute due to concerns of cost, availability, and technological maturity. This paper describes the use of an agent-based model to explore the system-level effects of unmanned vehicle implementation on the performance of these collaborative human–vehicle environments. In particular, this paper explores the introduction of three different unmanned vehicle control architectures into aircraft carrier flight deck operations. The different control architectures are tested under an example mission scenario using 22 aircraft. Results show that certain control architectures can improve the rate of launches, but these improvements are limited by the structure of flight deck operations and nature of the launch task (which is defined independently of vehicles). Until the launch task is improved, the effects of unmanned vehicle control architectures on flight deck performance will be limited.**

***Index Terms*—Agent-based modeling, decision support system, human–automation interaction, human supervisory control.**

I. Introduction

IMPROVING the capabilities of unmanned vehicles has been a key area of research over the past several years, but it is only relatively recently that they have improved to the point that integrating unmanned systems into civilian environments has become realistic. Several current examples exist: the Rio Tinto mining corporation is currently fielding autonomous hauling trucks in several locations [1], [2]; the Kiva systems order fulfillment system, which utilizes robotic pallet carriers, is used by online retailers Amazon and Zappos.com, among others [3]; automobile manufacturers continue to introduce new automated features, such as adaptive cruise control, lane departure warnings, and other technology while continuing to pursue automated road train and more advanced driverless technologies. Current initiatives by the defense advanced research projects agency (DARPA) seek to advance the capabilities of humanoid robotic systems in urban search and rescue scenarios (the DARPA Robotics Challenge), while other research seeks to address how robotic manufacturing systems cannot only work more safely near humans, but how they can also have a better understanding of the roles and tasks of their human collaborators [4].

Each of the domains described above can be characterized as a human centered, or heterogeneous manned-unmanned environment (HMUE): places where unmanned vehicles, human collaborators, and manned vehicles interact in the same physical space. A variety of heterogeneous manned environments, such as airport operations and aircraft carrier flight decks, already exist today. As unmanned vehicles transition into these domains, new paradigms of interaction will be required for human collaborators to work effectively with these unmanned vehicles. Changes to operating procedures may also be required to accommodate differences in capabilities between unmanned and manned vehicles. Very little research has addressed what these changes might be and how these changes, and the resulting introduction of unmanned vehicles, might affect the performance of the broader system.

Insight into the effects of unmanned vehicle integration on the performance of these larger scale systems could be gained by field testing prototype systems, but these environments are of such large scale (in physical size and number of active elements) that testing them in any meaningful manner would require significant investments of time, money, and physical resources. Even if these resources could be provided, if the unmanned vehicle systems being tested are not sufficiently similar to the future systems that would be deployed, the results may not be useful. Such tests might also place the autonomous systems, the researchers involved, and other by-standers in unforeseen danger, while also being limited in the number and types of conditions that could be explored.

Agent-based modeling [5] is one potential approach for addressing such limitations; these models attempt to replicate the behavior of individual actors within a given environment. Constructing such models of the real-world system provides an avenue for testing unmanned vehicle performance under a variety of test conditions that may not be easily achievable in live conditions. This paper examines the use of an agent-based simulation of a candidate HMUE, the aircraft carrier flight deck, for which the United States Navy is currently testing an unmanned carrier fighter aircraft that is expected to integrate fully into operations with manned aircraft and human crew [6], [7]. Flight deck operations are in many ways analogous to airport operations; in both environments, aircraft begin parked in a designated area, then coordinate with crew (or other controllers) to taxi to one of a few designated launch areas. With the Federal Aviation Administration's (FAA's) mandate to introduce unmanned aircraft

Manuscript received May 1, 2014; revised July 13, 2014 and October 1, 2014; accepted November 1, 2014. This work was supported by the Office of Naval Research Science of Autonomy program under Contract #N000140910625. The opinions expressed in this paper are those of the authors and do not necessarily reflect the opinions of the sponsoring agency. This paper was recommended by Associate Editor D. Gillan.

J. C. Ryan is with the Humans and Automation Lab, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: jachryan@gmail.com).

M. L. Cummings is with the Humans and Autonomy Lab, Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (e-mail: mary.cummings@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2014.2376355

into the national airspace system by 2015, inserting unmanned vehicles into ground operations will be a paramount concern. While the FAA has recognized the importance of ground operations in this integration, their major focus at this time appears to be on airborne operations [8]. While there are differences in the structure of airport and aircraft carrier operations (runways versus an open deck) and the role of human collaborators in those environments, lessons learned from the carrier flight deck should be applicable to airport operations.

This paper begins with a brief description of agent-based simulation and its applicability to modeling these environments. The three different unmanned vehicle control architectures of interest are then described, followed by a discussion of current aircraft carrier flight deck operations and how they are modeled within the Multi-Agent Safety and Control Simulation (MASCS). The modeling of the unmanned vehicle control architectures within MASCS is then described, followed by a description of the testing of these vehicles under mission conditions utilizing sets of 22 aircraft. Additional tests then explored how improvements to the launch task (with properties independent of vehicle control architecture) affect flight deck performance, as well as how changes to key control architecture design parameters can improve vehicle performance under these new conditions.

II. Related Work

A. Agent-Based Modeling

As described by Bonabeau [5], agent-based modeling is as much a perspective as it is a methodology, whereas the use of discrete-event or systems dynamics models may elicit specific ideas of structure and constituent components, agent-based models vary widely in content and in methods of application. Examples range from human decision-making (e.g., [9]) to epidemiology and disease transmission (e.g., [10]) to transportation systems involving both ground and aerial vehicles (e.g., [11]), among many others. The common theme is that, in each case, the agent-based modeling paradigm views the world "from the perspective of its constituent units" [5]—that the agents retain the same independence and exhibit similar decision-making outcomes as the real-world entities they are intended to replicate. The agents are then placed into the simulated environment, given a set of goals (which may be self-generated during the simulation), and allowed to make decisions independently from that point forward. An additional key to agent-based simulation development is the use of simple rules of behavior [5], [12]: agents are given limited options in any given decision they make. The simultaneous application of these simple decisions by the simulated agents is typically sufficient to replicate the behavior of the real-world system.

While agent-based modeling has been used to validate control and sensing algorithms for a variety of Unmanned Aerial Vehicle (UAV) systems (e.g., [13]), these studies have primarily addressed systems involving the collaboration of multiple vehicles in performing tasks, such as searching for and tracking targets. Another common area of interest is in the modeling and control of UAV "swarms" (large groups of robots coordinating on a specific task) [14]. Agent-based models have also been built to test unmanned vehicle survivability [15] and single UAV search [16]. However, none of these examples has considered the role of the human in interacting with the vehicle system, although other nonagent-based models of human-UAV interaction have been constructed [17], [18]. These latter models focused on the human as a processing server working within a queuing system, considering the behavior of the vehicle only in that it requires the operator to provide it a task. These latter models only, however, examined a single system of interest and did not construct models over a range of different vehicle control architectures. These models typically included only a few (4–8) aircraft in their models, as well.

No prior work could be located regarding the models of aircraft carrier flight deck operations, nor of agent-based models of similar collaborative human–vehicle domains (such as airport ground operations or mining environments) that included models of unmanned vehicle behaviors. The agent-based models cited in the earlier paragraph provided only general guidance on model development: the reliance on states, parameters, and simple rules of decision-making and motion. This formed the core approach to modeling the different unmanned vehicle architectures, with a focus on the features of UAV behavior that would have the most impact on the flight deck: physical motion, task execution, and interactions with operators and other human collaborators in the world. These techniques were used to develop the models of three different UAV control architectures—gestural control (GC), vehicle-based supervisory control (VBSC), and system-based supervisory control (SBSC), which are discussed in the following sections—to determine their effects on flight deck launch performance.

B. Gesture Control Systems

GC systems replace a human operator with a set of stereo-vision cameras and a sophisticated computer algorithm that replicates a subset of visual tasks performed by the human operator. In practice, this means that the system must be able to detect human collaborators in the world, track their movement, then track the motion of their hands, arms, and sometimes fingers. These motions are then translated into specific gesture definitions, which are correlated with specific tasks for the vehicles (e.g., "stop" or "turn left"). The vehicle's onboard autonomy then decomposes these actions into specific low-level control commands for sensors and effectors (throttle, steering, etc.).

While gesture recognition technology is available commercially in systems, such as the XBOX Kinect, the utilization of this technology for vehicle control remains in the research and development phase. One example of this is research currently being conducted by Song *et al.* that aims to construct a GC system specifically for aircraft carrier flight deck operations [19], [20]. This carrier environment is a unique case in that commands to human pilots are already provided solely through hand gestures, providing a natural mapping to gesture control systems.

C. Supervisory Control Systems

Sheridan's original definition of supervisory control systems defined them as systems in which one or more human operators are commanding and receiving feedback from an autonomous or robotic system that is performing tasks in the world [21]. This paper uses the term in a stricter sense: systems in which the human operator issues commands and receives feedback through a graphical user interface that translates symbolic inputs into actions for the robotic systems, which then decompose the actions into specific low-level control inputs. This typically requires substantial computational intelligence onboard the vehicle, as well as a variety of high-fidelity sensors to observe the world (GPS, LIDAR, and other range and sensing systems), cameras, high-accuracy inertial measurement devices, along with intelligent planning algorithms to handle vehicle routing and translate operator commands into low-level control inputs.

This paper classifies these systems into two forms. The first form is referred to as VBSCin which supervisors supply commands to a single vehicle at a time, working iteratively through the queue of vehicles waiting for instruction (e.g., [22]). The operator functions as a processing server within the system, maintaining a queue of vehicles awaiting task assignments and interacting with them individually. These systems have been a subject of significant prior work, a common theme of which has been the examination of the "fan-out" problem: how many vehicles the operator can manage without significant performance degradation. Past research has shown that operators can control up to eight ground vehicles or 12 aircraft at a time [23], an acceptable number for aircraft flight deck operations in which there are no more than 8–12 vehicles taxiing at one time.

Controlling more than 8 to 12 vehicles requires the introduction of additional autonomy in the form of intelligent planning and scheduling systems. These are integrated with additional decision support displays to form the class of SBSC systems, in which the operator works with the planning algorithms to replan all tasks for all vehicles simultaneously (e.g., [24]). Typically, the operator will provide goals for the planning algorithm (e.g., weights for the objective function) and may also insert additional constraints (e.g., time windows for the execution of certain tasks). The planning algorithm generates a candidate set of assignments that is then reviewed by the operator, who has a chance to reject the plan or make modifications. Once the plan is accepted, tasks are transmitted to vehicles, which then follow the prescribed motion paths and execute the assigned tasks. The human supervisor's responsibility is to monitor the execution of the plan and create new plans in the event of failures or performance degradation.

III. Multiagent Safety and Control Simulation Model

As the unmanned vehicle systems described in Section II do not yet exist for flight deck operations, simulations of their behavior cannot be validated directly. Instead, MASCS was created first as a model of manual control (MC) flight deck operations, with the unmanned vehicle architectures defined as modifications from this model. MASCS was then partially validated against empirical data in prior work [25] because of the limitations on accessible data on flight deck operations, pilot behavior, and other factors, full validation of this simulation is not possible at this time. Constructing the model required defining agents that replicate the behavior of human crew, pilots, aircraft, planning supervisors, flight deck equipment, as well as models of the tasks that they perform. Agent and task models consist of decision-making rules that govern agent behavior, parameters that describe how agents view and execute tasks in the world, and states that are viewable by other agents and are used in the decision-making routines.

The validation process compared the performance of individual aircraft and missions of multiple aircraft with empirical data on operations. This included the average time to complete launch missions of 18 to 34 aircraft, the interdeparture times of launches within those missions, and response of the system to changes in vehicle behaviors (e.g., speed). Additionally, a team of subject matter experts (SMEs) reviewed animations of simulation execution. SMEs had a range of experience on the flight deck, each over multiple years (two pilots each with more than 800 flight hours in fighter aircraft; another individual with over 3000 h as a Naval Flight Officer and 2 years as a launch and recovery officer). All were currently employed by Navy research organizations. SMEs were allowed to review repeated executions of the simulation in order to critique the activity of agents on the flight deck. SMEs were also shown quantitative results from validation testing to provide feedback on the accuracy of the simulation in terms of mission completion times. SMEs judged that the simulation was a reasonable replication of flight deck operations and did not find any major errors in modeling that required correction. Modeling proceeded to include the unmanned vehicles reported here, which were created by modifying six key parameters of the MC aircraft models: aircraft routing methods, visual detection accuracy, command comprehension rates, catapult alignment failure rates, camera field of view, and system latencies. This section first describes the basics of flight deck operations relevant to this paper and how these operations were modeled within MASCS before discussing the modeling of the unmanned vehicle control architectures.

A. Aircraft Carrier Flight Deck Operations

A digitized map of the flight deck appears in Fig. 1. Labels in the figure highlight the important features of the flight deck environment relevant to launch operations. Aircraft begin parked at the edges of the flight deck in the aft and starboard areas of deck; aircraft not able to fit in those areas are parked in the area known as "The Street." Aircraft will be assigned to launch from one of the four launch catapults (orange lines in the figure, numbered one to four). Each catapult is capable of launching a single aircraft at a time, but adjacent pairs of aircraft (1,2 or 3,4) are not able to operate in parallel, for two reasons. The first is that a single set of crew operates each pair of catapults and can only manage one catapult at a time. The second, and more important, is that the catapults are angled toward one another—simultaneous launches would result in a collision and loss of both aircraft. Catapults alternate launches within each pair, but the two pairs can process in parallel (e.g., 1 and 3 or 2 and 4). Operations typically allow one aircraft on the catapult and another waiting behind a protective barrier that rises from the deck during a launch. As such, each pair of catapults (if both are active) can queue up to four aircraft in the area at any time, but only one aircraft will be in the process of launching.

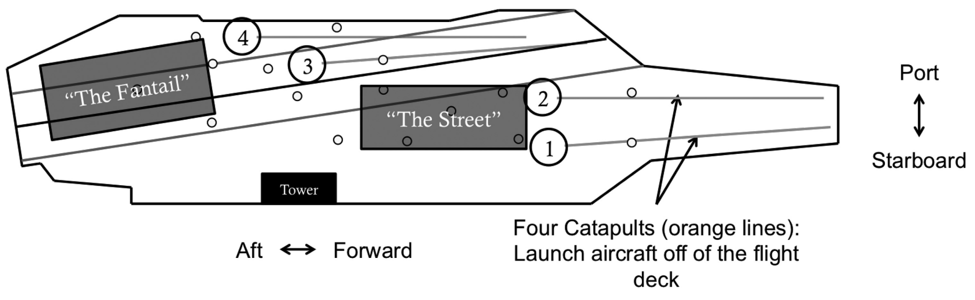
"The Fantail"
"The Street"
Tower
4
3
2
1
Port
Starboard
Aft ↔ Forward
Four Catapults (orange lines): Launch aircraft off of the flight deck

Fig. 1. Digital map of the aircraft carrier flight deck. Labels indicate important features for launch operations. Yellow dots represent Aircraft Director crew active on the flight deck.

The process of launching an aircraft requires that several tasks can be done in parallel. The aircraft must taxi forward on to the catapult, aligning its front wheel (nose gear) with the catapult centerline. The nose wheel is then physically attached to a part of the catapult known as "the pendant." While this is occurring, other crew are confirming the aircraft's current fuel level and total weight to calibrate the catapult for launch. The pilot also completes a set of control surface checks to ensure that the aircraft is ready for flight. These simultaneous tasks are modeled within MASCS as a single time distribution (*N* (109.65, 57.80^2)), based on prior research and discussions with SMEs.

Once these preparatory actions are complete, the catapult is triggered and the pendant begins pushing the nose gear and the aircraft forward up to flight speed. This acceleration task, also based on observations of flight deck operations, is modeled as a single time distribution (lognormal with location parameter $\mu = 1.005$ and shape parameter $\sigma = 0.0962$). Once the launch is complete, the aircraft at the adjacent catapult begins launch preparations, while the aircraft originally parked behind now taxis onto the catapult and waits for the adjacent catapult to complete its launch. Each of these tasks is independent of the type of vehicle being launched and is influenced primarily by the equipment and crew involved, rather than the control architecture.

Getting aircraft from parking places to catapults requires interactions with human crew on the flight deck. A senior enlisted person, termed the Deck Handler, and his staff are responsible for creating the schedule of operations and allocating aircraft to launch catapults. At the start of operations, assignments will be made such that queues at all catapults are filled. As launches occur and slots open at catapult queues, aircraft are dynamically allocated, assigning the next highest priority aircraft that has an open taxi route to a catapult. Aircraft assignments are then passed to a set of crew on the flight deck, termed Aircraft Directors (yellow circles in Fig. 1), who provide taxi instructions to aircraft. These crew work in a form of "zone coverage," each controlling a specific area of the deck (which may overlap with others). A Director will taxi the aircraft through their area, then hand the aircraft over to an adjacent Director before accepting another aircraft into their area. Directors communicate instructions to pilots using a set of hand gestures, informing the pilot when to drive forward, stop, turn, and to whom they are being passed off. Directors also must maintain an understanding of current traffic conditions on the flight deck, delaying some aircraft to allow other aircraft to taxi by. This may be because the other aircraft has a higher priority, or that failing to do so would "lock" the deck, with aircraft unable to taxi to their required destinations.

Models of these actors were constructed within the MASCS model, each with their own sets of decision rules to replicate the behaviors described in the previous paragraphs. Table I provides a summary list of agents modeled in MASCS and the relevant decision making rules and parameters of each. As noted previously, the goal in agent-based modeling is typically to make the decision rules as simple as possible [5]. In MASCS, this was done by building decision rules that had only a limited number of options in each decision. Most decision-making models in MASCS are binary (yes/no) and based on the current locations and states of agents on the deck. Options are often reviewed iteratively, selecting the first feasible option. If no feasible option exists, no action is taken and the system waits until conditions change.

The Deck Handler agent's scheduling rules are based on a set of assignment heuristics elicited from SMEs as part of prior work [26]. Given a list of aircraft waiting to launch, their locations, and their priorities, simple rules attempt to assign the highest priority aircraft to the nearest available catapult. Aircraft Director agents include a data structure describing to which other Directors they can pass aircraft to and rules that determine whether or not an aircraft can be passed off to the desired next Director, whether the Director can accept new aircraft themselves, as well as how to align to the aircraft they are currently instructing so that they are visible to the pilot. Aircraft and pilots are modeled within MASCS as a single unit at this time. Their models primarily concern physical motion on the flight deck and their ability to follow instructions from the Aircraft Director crew. Of greatest importance are rules that describe whether they can "see" a Director. Whether a pilot can "see" a Director is a simple check of whether the Director is in the visible area of the aircraft, as defined by a field-of-view parameter.

Aircraft Directors and Aircraft/Pilot models both require models of their motion on the flight deck. These models require both the definition of speed of travel and rules that describe the paths they should take on the flight deck. These rules were placed into separate task models that are assigned to agents for

TABLE I
LIST OF AGENTS REQUIRED BY THE MASCS SIMULATION OF FLIGHT DECK LAUNCH OPERATIONS, THE IMPORTANT FEATURES (PARAMETERS AND LOGIC STRUCTURES) INVOLVED IN THEIR MODELING, AND THE VALUES OF THOSE PARAMETERS

Agent	Feature	Description
Aircraft/Pilot	Speed	Rate at which vehicles taxi forward on the flight deck; *N* (3.5, 1.02)
	Turn Rate	Rate at which vehicles rotate during taxi operations; 18° /s
	Sight Angle	Area in front of vehicle to view Directors; 85°
Aircraft Director	Speed	Rate at which crew walk on the flight deck; *N* (3.5, 1.02)
	Alignment logic	Determines when and to what location Directors should align to aircraft
Deck Handler	Routing Logic	When and to what Director the vehicle should next be sent
	Traffic Management logic	How to deconflict taxi paths on the flight deck
	Routing Logic	Determines whether or not a taxi route exists between an aircraft and a catapult. Defined by heuristics that examine whether aircraft are parked in key places on flight deck.
	Planning Logic	Given rules of routing logic, assigns highest priority aircraft to nearest available catapult. Prioritize sending aircraft to catapults with no aircraft in queue.
Catapults	Launch Constraints	How launches alternate between catapults and when obstructions in the nearby area should stop a launch in progress
	Queue Structure	List of aircraft currently assigned to this catapult (maximum 2) and how aircraft are promoted in the queue

execution. These task models determine, for a given agent with an assigned destination, what combination of straight line and turn actions is required. During execution, the models continually check the conditions of the deck to make sure that the transit path is clear. For an aircraft taxiing on the flight deck, the "Taxi" task model continually checks 1) whether this action should be delayed due to traffic conditions and 2) if another aircraft is physically blocking the current vehicle, as well as 3) if the aircraft can "see" the Director. If any of these conditions is true, the motion is paused until conditions permit movement. All aircraft and crew execute such rules simultaneously during their operations.

Modeling the UAV control architectures described in Section II within MASCS requires changes to the models described above. The general planning methods do not differ substantially, nor do the models of physical motion. However, the rules governing task execution and taxi routing for MC operations may not apply to unmanned vehicles due to the differences in control architecture capabilities. Other parameters describing the vehicles' physical characteristics might also vary from the baseline MC model. The differences that define the unmanned vehicle control architectures used in this paper are discussed in the next section.

B. Models of Unmanned Vehicle Behavior

In the context of flight deck operations and the models of agent behavior, the important activities of a pilot that must be replicated by an unmanned vehicle control architecture involve three key aspects: visual detection of an Aircraft Director that is attempting to provide instructions, correctly interpreting task instructions, and correctly executing the commanded tasks. Each of these corresponds to a related parameter: the probability of observing the Director if within the vehicle's field of view, the probability of correctly interpreting the command, and the probability of correctly executing a task. For the baseline MC model of current flight deck operations described in the previous section, the likelihoods of these failures are so low that SMEs consider them nonexistent (0% likelihoods). A fourth difference in behavior comes from the fact that the Director is providing instructions in this "zone coverage" routing; this may not occur for some forms of UAV control, and thus how vehicles are routed forms a fourth behavioral parameter of "routing method."

Two other parameters relate to the physical characteristics of design of the vehicle: the field of view of the pilot/aircraft, which describes where the Director should be in order to provide commands, and the latency in processing commands and executing tasks. The field of view for manual operations is related to human sight range and the constraints of being in the cockpit; for unmanned vehicles, field of view is related to the cameras used on the vehicle. The latency in processing commands could come from a number of sources: delays in the communication of signals between operator and vehicle, the time required for the vehicle's computer system to process, or time required for the human operator to interact with the system. Each of these can also be modeled as variables within the MASCS simulation environment, and a full list of these six design parameters appears below. The following paragraphs describe the settings and variations of these six performance parameters in the context of the different vehicle control architectures.

1) Vehicle behavior parameters.
    a) How aircraft are *routed* on the flight deck (with or without the Aircraft Director crew).
    b) *Visual detection:* the probability of failing to detect an operator when they are within the field of view.
    c) *Task comprehension:* the probability of failing to comprehend an issued task.
    d) *Catapult alignment (task) execution:* the probability of failing to properly align to a catapult.
2) System Design parameters.
    a) The *field of view* of the pilot/autonomous vehicle, which affects how Aircraft Directors align to aircraft during operations.
    b) *Latency and lag* present in the system due to communications latencies, processing times, or limitations of the human operator.

1) Unmanned Aerial Vehicle Routing Methods: The first major difference between UAV Control Architectures involves how they are issued taxi commands and routed through the flight deck. GC systems interact with the crew on the flight deck in a similar fashion as MC vehicles, which requires modeling the three remaining vehicle behavior parameters in the list above. The two supervisory control systems leverage their centralized network to perform tasks in a different manner. VBSC systems would rely on commands issued from the Handler through a graphical user interface, which requires modeling the behavior of the Handler in operating such a system. In this form of system, the operator functions as a queuing server, which requires the definition of both a service time and a server policy for managing tasks. For SBSC systems, the operator interacts periodically with a planning algorithm that replans all tasks for all vehicles at once. Once the plan is updated, the aircraft are allowed to execute tasks until such a time as a new plan is required.

The planning heuristics used currently by experienced planners aim to keep traffic moving on the flight deck in an orderly fashion, minimizing the occurrence of routing conflicts or other blockages that limit aircraft taxi motions [26]. To ensure this continuity in VBSC systems, experienced Handlers would likely operate in a "highest priority first" fashion, prioritizing aircraft nearest to their assigned catapults in order to keep traffic flowing on the deck. The time required to process tasks is taken from prior research by Nehme [17], whose research involved operators controlling a team of unmanned vehicles through a VBSC-style interface, requiring them to define waypoints that defined vehicle transit paths, among other task assignments. Nehme characterized the "idle" task, in which operators assigned a new transit task to a waiting vehicle, as normally distributed with a mean of 3.19 s and standard deviation of 7.19 s (bounded $[0, \inf]$).

The interaction of the Handler's interaction time with each vehicle and the current number of vehicles in the queue also generates latencies for VBSC systems: not only must a VBSC vehicle wait $N\ (3.19,\ 7.19^2)$ seconds at the start of each of its own tasks, it must wait for all other vehicles ahead of it in the queue to be processed. If four vehicles are ahead in the queue, the fifth vehicle must wait (on average) $3.19 * 5 = 15.95$ s before starting its next task. Thus, there is a significant penalty for including additional vehicles in the VBSC queueing system, which has been characterized in many previous studies (see [22], [23] for examples).

Additionally, since this style of routing does not rely on the network of crew on the flight deck, the Handler can also assign longer taxi tasks on the flight deck—taxiing from aft to forward, barring other constraints, could be done in a single motion rather than requiring multiple handoffs between Directors. The taxi paths, however, would largely stay the same. These same routing changes also would occur for the SBSC architecture, which enable a single operator, with the help of a scheduling algorithm, to simultaneously define task lists for all vehicles. In the SBSC architecture, once assignments are made, just as with VBSC vehicles, tasks are carried out without the assistance of the Aircraft Directors in taxi operations. Prior work in this area [26] demonstrated that even complex planning algorithms have trouble decomposing the geometric constraints on the flight deck and provide little benefit over human planning. The appropriate development and testing of planning algorithms for the flight deck lies outside the scope of this research; for SBSC systems, the same Handler planning heuristics applied to the other control architectures (CAs), which performed as well or better than other planning algorithms in prior work [26], are applied here.

2) Visual Detection, Task Comprehension, and Task Execution: For the Gesture control systems that rely on visual information, the processes of visual detection of Aircraft Directors, comprehension of the commanded tasks, and the execution of those tasks must also be modeled. There is a chance of failure in each individual recognition task, most importantly in recognizing and the Director and identifying the commands they are providing. As these systems are still largely in the research and development phase, there is no true failure rate known for gesture control activity in the flight deck environment. A review of several papers [20], [27]–[33], most of which incorporated data for multiple tests and trials and references multiple prior studies, provided a picture of the current general state of gesture recognition research. Out of the 124 different experimental results reported, eight report accuracies of 99% or better (all from a dataset using only ten actions and a simple, homogeneous background), 32 reported results better than 95%, and 77 reported better than 90% accuracy. Overall, the studies report an average failure rate of 11.16%. It is not clear how this failure rate translates into realistic future operations; while the technology will certainly improve, a variety of environmental conditions (sun glare, obstructions, large numbers of moving vehicles) will limit performance in the real world. While future real-world failure rates are still not known, the median failure rate of 11.16% observed in the research is used as the failure rate of the baseline GC system.

A logical implementation of GC could imply that a failure to recognize a Director might only happen once, after which the Director would have the authority to force control of the vehicle. If a vehicle fails to recognize a *task*, however, there may not be a simple method to force the system to recognize that specific task. It may take several attempts for the vehicle to register the correct command, with each failure accruing a time penalty. For the baseline MASCS model of GC, the penalty for failing to acquire a new Director is set at 5 s, representing the time required to realize the vehicle has not recognized the Director and for the Director to issue an override. The time penalty for failing to acquire task is modeled as a uniform distribution between 5 and 10 s, representing the time for the operator to stop, issue a cancel command, and wait for the system to become ready to accept new instructions. However, the original latency in processing is not applied to these failures. The number of failures in command comprehension is determined at the start of each action commanded by a Director by a Bernoulli trial generator, after which the total time penalty is solved by randomly sampling the uniform interval [5], [10] once for each failure incurred.

3) Latencies: One form of latency in the control architectures occurs for the VBSC systems, due to the way in which the operator manages the queue of waiting vehicles (described previously in Section III-B1). GC systems will also experience a form of latency due to software processing, as the system attempts to translate the hand gestures of the Director into understandable

TABLE II
COMPARISON OF UAV CONTROL ARCHITECTURE DESIGN PARAMETERS DEFINED FOR MASCS

	Manual Control	Gesture Control	Vehicle-Based Supervisory Control	System-Based Supervisory Control
Routing Method	Zone coverage	Zone coverage	Centralized control	Centralized control
Visual detection rate	P(fail) = 0	P(fail) = 0.1118	N/A (central control)	N/A (central control)
Task Comprehension Rate	P(fail) = 0	P(fail) = 0.1118	N/A (central control)	N/A (central control)
Task Execution Rate	P(fail) = 0	P(fail) = 0.05	P(fail) = 0.025	P(fail) = 0.025
Field of View	85°	65°	N/A (central control)	N/A (central control)
Lag/Latency	None	0–3 s (computation)	*N* (3.19, 7.192) per task	None

action commands. The prior research reviewed for error rates [20], [27]–[33] also shows this requires anywhere from <<1 to 3 s; this is modeled as a uniform distribution over the range [0, 3], randomly sampled and applied to the start of every action commanded by a Director.

C. Performance Comparisons

Table II provides a comparison of the four UAV control architectures to the baseline MC model in terms of the six primary design parameters described earlier. For the MC model, SMEs report that failures on these tasks are so rare that they do not really recognize them as occurring in the system: pilots rarely fail to see a Director if they are within their field of view, rarely fail to interpret a command, and rarely fail to properly align to a catapult. The field of view parameter was calibrated in prior work to a value of 85°. These settings were utilized in the model and partially validated in prior work [25] and are considered to be representative of the highly skilled pilots that work within the aircraft carrier flight deck.

Each of the unmanned vehicle control architectures defined in previous sections can be modeled in MASCS by changes to the six design parameters listed in Table II. As compared with a baseline MC vehicle, GC systems will still operate under the crew's "zone coverage" operating paradigm, but will introduce both a small to moderate processing latency at the start of each action and a chance to fail to recognize either the person issuing the command or the command itself (these are akin to another small delay at the start of operations). As noted in Sections III-B2 and III-B3, this latency ranges from 0–3 s and the average failure rates are slightly over 10%. The processing latency is randomly sampled over the defined interval at the start of the attempted task. Task failures can occur repeatedly, with each failure resulting in a time penalty between 5 and 10 s to resubmit commands to a GC vehicle. The time penalties are also sampled randomly over the defined interval. The use of cameras by the gesture recognition system might also constrain the applicable field of view for these vehicles, and the lack of a practiced pilot in aligning the aircraft to the catapult at the end of taxi operations will likely increase the rate of error on this task. However, there is no current evidence to suggest what this failure rate may be in real life. A catapult alignment failure rate of 5%, translating to roughly one failure per mission (once launch of a 22 aircraft sortie), was selected for the GC vehicles based on SME opinions.

As VBSC and SBSC systems are centrally controlled, they receive no instructions from the crew. Thus, the visual detection and task comprehension rates are not applicable for these two systems, nor is the modeling of a field of view. Both are expected to at least have a nonzero failure rate in aligning to aircraft catapults, set at once per two missions. SBSC systems have no latency, as vehicles are sufficiently autonomous to execute the series of tasks provided to them quickly and without human interaction and any replanning occurs in parallel to vehicle operations. VBSC systems, however, rely on a human operator to supply tasks individually to aircraft, which must wait in a queue for further instructions. This introduces latency in the form of a wait time for operator interaction, which for large numbers of aircraft may lead to significant delays in task execution.

The differences in performance of individual vehicles under these control architectures should thus be a function of the cumulative delays and latencies experienced in task recognition and execution. The GC and VBSC systems both experience a greater level of latencies and delays than MC and SBSC systems and should thus perform more poorly. For GC systems, these arise from the chances of repeated errors and processing delays on the part of the vehicles, while for VBSC they are due to the need for repeated interactions with the human supervisors. However, the impact of these delays and latencies when multiple aircraft are included in flight deck operations will be influenced by the structure of modern aircraft carrier flight deck operations. The model of operations constructed for this paper was partially validated in [25] and is discussed in the next section.

IV. EFFECTS ON OPERATIONS ON UNMANNED AERIAL VEHICLE PERFORMANCE

The models of unmanned vehicle control architectures described in the previous section were implemented into the MASCS environment and verified through a set of tests that examined the behavior of an individual aircraft on the flight deck. The same mission scenario was executed for each of the four control architectures (MC, GC, VBSC, and SBSC) with 300 replications performed in each case. A diagram of this task appears in Fig. 2: the aircraft begins parked in the aftarea of the flight deck and is assigned to launch from catapult 3, requiring a series of straight-line and turning motions before culminating with the launch preparation and launch acceleration tasks. This same scenario was used previously in the development and testing of the baseline MC model [25]. The results of these simulations appear in Fig. 3, which shows the average time for a single aircraft to complete the launch mission (from taxi through launch acceleration) for each the four control architectures, with whiskers indicating ±1 standard error in the data.

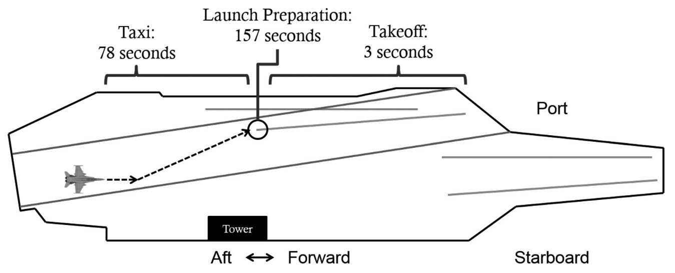

Fig. 2. Diagram of mission executed in the single aircraft performance comparisons.

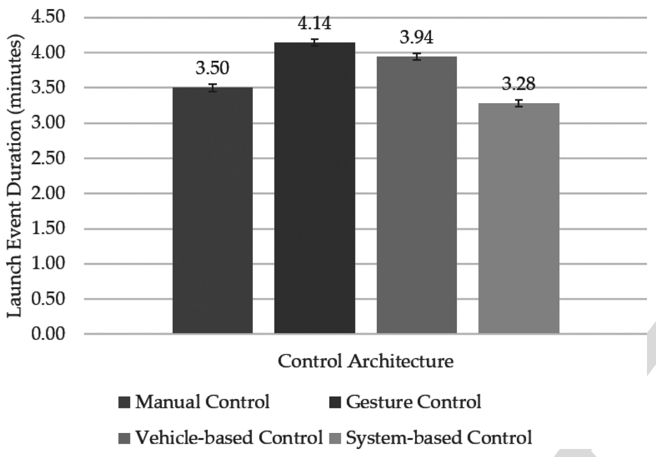

Fig. 3. Values for launch event duration (LD) for single aircraft testing for manual control (MC), gestural control (GC), vehicle-based supervisory control (VBSC), and system-based supervisory control (SBSC). Bars indicate mean LD for sets of 300 replications, with whiskers showing ±1 standard error.

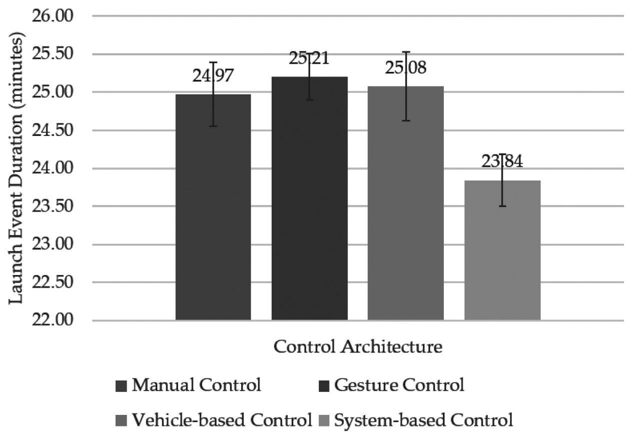

Fig. 4. Results for Launch event Duration (LD) for each control architecture for missions using 22 aircraft under the original launch preparation time model. Whiskers indicate ±1 standard error.

In general, the responses of four control architectures match the expectations described earlier: the GC and vehicle-based supervisory control (VBSC) cases being the slowest and MC and SBSC being the fastest. An analysis of variance (ANOVA) test demonstrates that the differences between the MC, GC, SBSC, and VBSC cases are significant at the $\alpha = 0.05$ level ($F(3, 1196) = 66.07, p < 0.0001$), and a Tukey test reveals that all pairwise comparisons are significant. In terms of practical differences, GC vehicles take 0.64 min longer to complete the task, VBSC 0.44 min, and SBSC are 0.22 min faster. While these are indeed small differences, percentagewise these translate into percentage differences of 18.4%, 12.5%, and −6.3%, respectively, from the MC baseline.

However, the structure of flight deck operations may affect how these variations in individual vehicle performance are felt in missions using many aircraft. Flight deck operations, as well as airport operations, are such that aircraft depart an initial parking area and are sent to queue at a specific launch site. In terms of system performance, as long as another aircraft arrives at the site prior to the launch of the previous aircraft, the flight deck (or airport) is working effectively. In order for the differences in the individual control architectures to be observed at the mission level, the delays and latencies that affect GC and VBSC performance must disrupt taxi operations upstream of the catapult sufficiently to break the chain of queuing that occurs.

To test whether or not this occurs, the four control architectures were tested in missions using 22 aircraft. In these missions, all 22 aircraft begin parked at the edges of the flight deck. The Deck Handler agent then dynamically assigns aircraft in real time based on current aircraft priority and whether nearby catapults are open and accessible to the aircraft. Each mission utilized the same initial conditions (vehicle parking locations and priorities) and used a completely homogeneous vehicle set (all MC, or all Gesture Control, etc.). Thirty replications of each mission were performed and the total time to complete the mission (from initialization to completion of the last launch) was logged for each. Fig. 4 provides a column chart of the average time to complete missions for each control architecture, with whiskers indicating ±1 standard error in each dataset.

The data in the figure suggests no real variations occur in system performance, with an ANOVA test returning marginally significant results ($F(3, 116) = 2.57, p = 0.058$). This suggests that the effects of GC and VBSC system delays are not sufficiently disruptive to cause differences at the mission level, nor are the benefits of the SBSC system, which does not rely on the crew "zone coverage" routing, sufficient to improve performance at a highly significant level.

One interpretation of these results is that an error exists either within the aircraft models, the routing of aircraft to catapults, or the processing of tasks at the catapults. However, the individual models of aircraft (see Fig. 3) showed that differences in performance do exist on an individual vehicle level; some mechanism at the mission level is preventing these differences in individual performance from having an aggregate effect. One possible explanation regards how aircraft queue at launch catapults, as described in Section III-A: each catapult can allow one aircraft onto the catapult and another to queue behind. At a catapult pair, only one aircraft is actively preparing for launch at any given time. Once this aircraft launches, the aircraft parked behind it moves forward onto the catapult and the aircraft sitting on the paired catapult begins preparing for launch. A new aircraft is then taxied to the empty place in the catapult queue.

The actions of the Deck Handler create this queueing pattern immediately and keep it in existence throughout the mission.

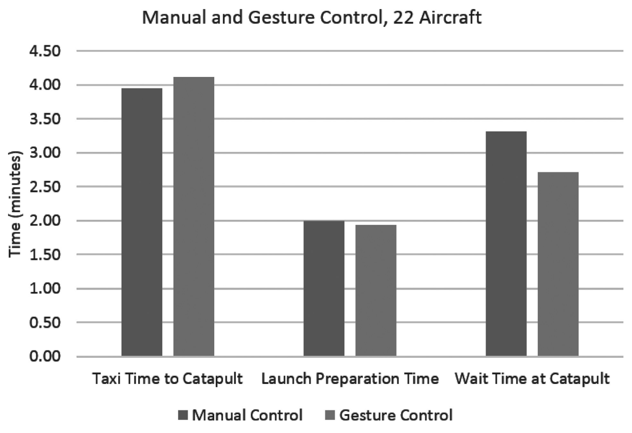

Fig. 5. Column chart of breakdown of tasks in a 22 aircraft mission using manual control (MC) or gesture control (GC).

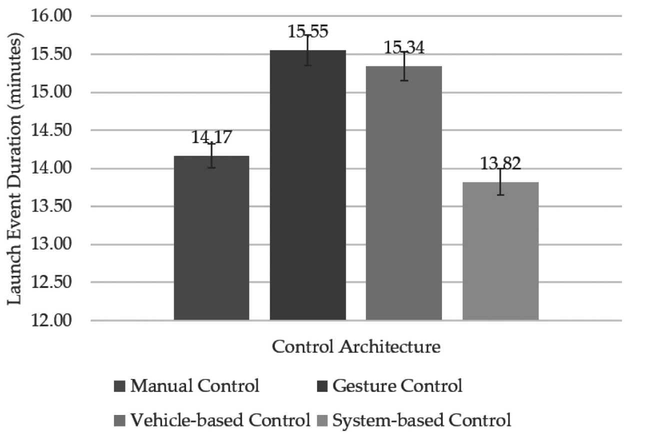

Fig. 6. Results for launch event duration (LD) for each control architecture for missions using 22 aircraft under the automated launch preparation time model. Whiskers indicate ± 1 standard error.

With this queueing pattern in place, new aircraft arrive at catapults only to wait a period of time in the queue. If all control architectures are efficient enough in taxi operations to supply aircraft to catapults before the queues empty, then differences in launch event durations across control architectures at the mission level will not be observed. Emptying a queue means that a catapult pair would have to process all three prior aircraft in the time it takes the new (fourth) aircraft to reach the queue. Additional output data from the MASCS simulation suggests that this does not occur and the queues remain in place throughout the mission (see Fig. 5). In a 22 aircraft mission using only MC aircraft, the average time for an aircraft to taxi to a catapult is just under 4 min. At an average launch time of 2 min, the queue would be cleared in 6 min, greater than the average taxi time. This implies that aircraft should be encountering queues at catapults, and data demonstrates that they are, at an average wait time of 3.32 min. The figure also includes the same data for Gesture Control, the worst performer in the single aircraft testing described earlier, which provides similar results for taxi times, launch times, and wait times.

These data suggest that the manner in which aircraft are queued at catapults, combined with the current launch preparation process, serves to bottleneck operations. Reducing the time required to complete the launch preparation task should provide conditions in which queues are eliminated, allowing differences to appear between control architectures at the mission level. The next section examines how decreasing the mean and variance of the launch preparation task affects the performance of the different UAV control architectures.

A. *Effects of Launch Preparation Time*

The launch preparation task is a highly manual process executed by several members of the deck crew, a key component of which is the physical attachment of the aircraft nosegear to the catapult. The pilot must first correctly align the aircraft to the catapult before a crew member connects a piece of the catapult mechanism to the aircraft nosegear; after this occurs, the catapult pulls the aircraft forward to test the connection. Other tasks performed by other crew involve information processing, setting the strength of the catapult based on the current weight of the aircraft. Since all vehicles connect to the same catapult mechanism and perform the same checks, the time to complete this task is defined independently of the types of aircraft being connected. Additionally, because these tasks occur largely in parallel, the entire process was modeled as a single normal distribution with MASCS. Decreasing the mean value of this process could potentially be accomplished through better training of the crew, while decreasing the variance might be done by automating certain aspects of the task and removing the more variable human components of the task.

A new series of tests were conducted using a "reduced" launch preparation time with significantly reduced mean and variance. In this reduced model, the mean was reduced by half and the variance by three-quarters (to N (54.82, 28.9^2)). All four control architectures were tested under this launch preparation time model using 22 aircraft. Thirty replications were again performed for each case, and the results of these tests appear in Fig. 6 with whiskers denoting ± 1 standard error.

At these settings ($\mu = 54.82$ s, $\sigma = 28.9$ s), an ANOVA reveals that significant differences do exist within the data ($F(3, 116) = 18.63, p < 0.0001$). A Tukey test reveals significant differences between two pairs of control architectures: GC ($\mu = 15.55$) and VBSC ($\mu = 15.34$) are both significantly different from MC ($\mu = 14.17$) and SBSC ($\mu = 13.82$). However, no significant differences exist within each pair (e.g., GC and VBSC). In terms of practical effects, the difference between the averages of the fastest (SBSC) and slowest (GC) control architectures is 1.73 min, equivalent to 12.2% of the MC baseline.

That variations in control architecture performance do arise with this automated launch preparation time model suggests that the characteristics of the launch preparation time do have significant effects on operations. The original launch preparation time, with its high mean and variance, serves to bottleneck flight deck operations; regardless of what is occurring, aircraft must always queue behind other aircraft at the catapults. While this makes the system robust to variations in unmanned vehicle behavior (the delays that occur in GC and VBSC systems), it also means that beneficial effects of unmanned vehicle introduction are also obscured. Improving operations on the flight deck would first require improving the performance of this vehicle-independent processing task, rather than improving the performance of individual vehicles.

These results are also important for other similar domains, such as future airport operations. Given the first set of results under the original launch preparation model, the introduction of unmanned vehicle systems provided no benefit whatsoever to system operations due to the settings of the vehicle-independent launch preparation task. Given this data, a stakeholder would likely decide that UAV integration provides no tangible benefits to operations. They might also decide that because there are no differences, all types of unmanned vehicles provide the same level of performance in the system; the choice might then be made based on other factors. If this occurs, and the system moves to a faster tempo of operation, this decision might have severe consequences. In aircraft carrier operations, tempo increases during wartime combat operations; in airports, increasing the tempo could help relieve the congestion common at many U.S. airports. At this increased tempo setting, if stakeholders have already locked in to using a GC or VBSC system based on the previous data, operations will not perform to the level that was originally expected.

The use of simulations like MASCS, as presented here, can help to clarify what differences can be expected from these systems in the real world and how their performance might be influenced by other process-level tasks independent of the UAV system control architecture itself. They can also help to explore under what conditions performance of these systems could be improved. While the results shown above demonstrated that the GC architecture performed poorly under the "reduced" launch preparation time model, the GC system also had the most changes from the MC baseline, varying on five of the six design parameters listed in Section III-C. Improvements along some combination of these parameters should be sufficient to provide GC performance equivalent to that of the baseline MC architecture. The next section demonstrates how simulations like MASCS can be used to examine the effects of changing vehicle performance parameters for the control architecture models in an example design space exploration.

B. Design Space Exploration

In future systems, stakeholders may seek to understand what changes to unmanned system control architectures are required to improve the performance under certain system conditions. This section examines how this might occur for the case of the GC architecture, which previous results showed as having the worst performance of the modeled architectures. The most important changes in the Gesture Control architecture involve the increased failure rates of these vehicles in recognizing Aircraft Directors and their commands and the processing lag in recognizing those commands. The failures also carry with them time penalties that delay vehicle activity on the flight deck and that, as demonstrated, can sometimes be detrimental to flight deck performance. Improving the accuracy of these systems and reducing their processing times should improve the performance

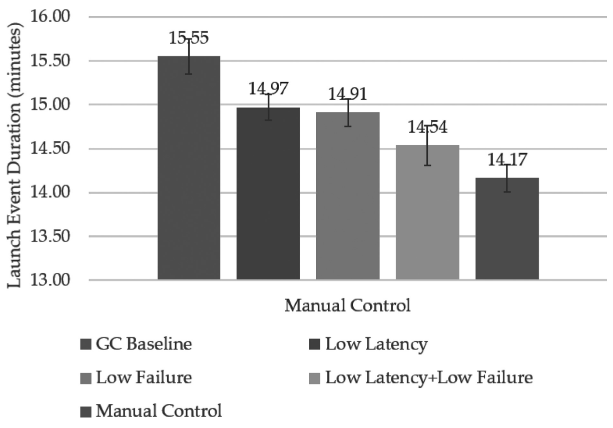

Fig. 7. Results for launch event duration (LD) for variations of 22 aircraft Gestural Control (GC) model using the "reduced" (mean-50%, standard deviation-50%) launch preparation time distribution. Whiskers indicate ±1 standard error.

of the GC system, but it is not known which of these carries more weight in terms of improving mission performance.

Three new GC models are tested in this section over a variety of mission conditions. First, a "Low Failure" case is created in which the failure rate is reduced from 11.16% to 1%, modeling a "best-case scenario" of system accuracy. Second, a "Low Latency" is created in which commands are processed at a maximum of 0.5 s. A third model combines the two cases, generating a Low Latency + Low Failure condition that should show improvement in performance. A first series of tests addressed the effects of these models in missions involving 22 aircraft under the original launch preparation model ($N\,(109.65,\ 57.80^2)$), but no significant differences were observed (ANOVA, $F(4, 145) = 0.51, p = 0.73$). This is to be expected, given the effectiveness of the original launch preparation time distribution in obscuring variations between control architectures.

A second round of testing then proceeded to the "reduced" launch preparation model ($N\,(54.82,\ 28.92)$), reducing both the mean and standard deviation by 50%. Thirty replications of the same 22 aircraft mission used previously were conducted for each of the three new GC models (Low Failure, Low Latency, Low Latency + Low Failure). Given the results of the previous section, the variations in performance were expected to appear at this faster launch preparation time model. The results of these tests appear in Fig. 7 and include results for the baseline GC and MC models for comparison.

The figure shows that the data exhibits a noticeable trend from the baseline GC model to Low Latency, then Low Failure, then Low Latency + Failure, then the baseline MC model. An ANOVA shows that significant differences do exist between the five test cases ($F(4, 145) = 8.12, p = 0.0001$). A Tukey post hoc test shows that, most importantly, the GC Low Latency + Low Failure and MC cases are not significantly different from one another ($p = 0.60$). The MC data is significantly different from the GC Baseline ($p < 0.0001$), Low Failure ($p = 0.017$), and Low Latency ($p = 0.034$) cases. The Low Latency + Low Failure case is also significantly different from the GC baseline model ($p = 0.0012$), but is not different from the Low Failure ($p = 0.17$) and Low Latency ($p = 0.10$) cases. This is fairly interesting and highlights the significance of the earlier issues with the baseline GC models: the GC architecture requires substantial improvements to *both* the gesture recognition accuracy *and* processing speeds to achieve performance equivalent to MC operations. Improvements to either of these elements individually were not strong enough to generate significant improvements in GC performance.

V. Conclusion

This paper has described the use of an agent-based simulation model of aircraft carrier flight deck operations that allowed for a systems-level analysis of the key features of unmanned vehicle behavior that influence aircraft carrier flight deck performance. Testing demonstrated that the launch preparation process, whose execution times are independent of any specific unmanned vehicle control architecture, heavily influenced system behavior and effectively obscured differences between control architectures. Differences in control architecture performance were only observed under a task model with significantly reduced mean and variance. However, this launch preparation process is not slated for improvement in the near future. Ultimately, these results suggest that failure to reexamine the structure of flight deck operations and the system as whole will ultimately limit the effectiveness of introducing autonomy into aircraft on the flight deck.

Additional testing examined how, under such a reduced launch preparation time, the simulation could be used to investigate the effects of modifying control architecture parameters. These results showed that Gesture Control systems required improvements to *both* accuracy rates and processing times in order to achieve performance equivalent to manual control operations in a 22 aircraft mission. Overall, these results suggest that the structure and definition of parameters within the broader system (the flight deck and tasks defined independently of the control architectures) has a significant impact on the behavior of the control architectures modeled in this paper. For stakeholders making decisions regarding which types of control architectures to implement in future operations, for environments with similar structure to the carrier flight deck, the definition of tasks like the launch preparation time (the final task performed by vehicles before exiting the system) greatly affects control architecture performance.

The focus of MASCS at this time has been a general examination of how different unmanned vehicle control architectures affect operations on the aircraft carrier flight deck, using models of future UAV control architectures that have reached some level of maturity in operations. A significant limitation of this modeling and this research has been the exclusion of rare and safety-critical events from the system. These are an important aspect of operations in any complex system, and models of rare events, failures, and other issues could be included in the modeling of the physical environment (weather conditions, damage to the ship and flight deck), the vehicle models (mechanical or electrical failures, etc.) or in the modeling of the human actors (poor decision making, random accidents, etc.). These are all important areas of future work that can be addressed in future research, but until formally validated, MASCS cannot be considered a true predictor of real-world system performance.

Ultimately, as unmanned systems move from operating in isolated environments to more complex human-centered ones, understanding the drivers of performance for both unmanned vehicles and the systems in which they operate is key. This research has demonstrated how this can be achieved through the use of agent-based modeling of unmanned vehicle systems and provided results that suggest that, for future aircraft carrier flight deck and airport operations, improving aspects of the system itself are as important, if not more so, than UAV improving the performance of the UAV systems being introduced. Future work will address how these control architectures interact with changes to the operational structure, examining how changes to the nature of planning and routing of aircraft tasks in the flight deck environment interact with other elements to drive variations not only in mission durations, but in terms of safety and efficiency as well.

References

[1] R. Tinto, “Mine of the future,” 2011. Q1

[2] A. Topf, “Rio Tinto boosts driverless truck fleet for use in Pilbara,” 2011.

[3] J. Scanlon, “How KIVA robots help Zappos and Walgreens,” 2009.

[4] S. Nikolaidis and J. Shah, “Human-Robot interactive planning using cross-training-a human team training approach,” presented at the AIAA Infotech@Aerospace, Garden Grove, CA, 2012.

[5] E. Bonabeau, “Agent-based modeling: Methods and techniques for simulating human systems,” *Proc. Nat. Academy Sci. United States Amer.*, vol. 99, no. Suppl. 3, pp. 7280–7287, 2002.

[6] B. McKinney, “Unmanned combat air system carrier demonstration (UCAS-D),” 2011.

[7] W. J. Hennigan, “Navy drone X-47B lands on carrier deck in historic first,” Jul. 2013.

[8] “Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap,” Tech. Rep., Federal Aviation Admin., Washington, DC, USA, 2013. Q2

[9] K. Sycara and M. Lewis, “Agent-based approaches to dynamic team simulation,” Tech. Rep. NPRST-TN-08-9, Navy Personnel Research, Studies, and Technology Division, Bureau of Naval Personnel, Millington, TN, USA, 2008.

[10] K. Carley, D. Fridsma, E. Casman, N. Altman, J. Chang, B. Kaminsky, D. Nave, and A. Yahja, “BioWar: Scalable multi-agent social and epidemiological simulation of bioterrorism events,” *IEEE Trans. Syst., Man Cybern. Part A, Syst. Humans*, vol. 36, no. 2, pp. 252–265, 2006.

[11] S. M. Lee, A. R. Pritchett, and K. M. Corker, “Evaluating transformations of the air transportation system through agent-based modeling and simulation,” in *Proc. 7th FAA/Eurcontrol Seminar Air Traffic Manage. Res. Develop.*, 2007. Q3

[12] J. M. Epstein and R. M. Axtell, *Growing Artificial Societies: Social Science From the Bottom Up*. Washington, DC, USA: Brookings Institution, 1996.

[13] S. J. Rasmussen and P. R. Chandler, “MultiUAV: A Multiple UAV Simulation for Investigation of Cooperative Control,” presented at the Winter Simulation Conf., San Diego, CA, USA, 2002.

[14] J. J. Corner and G. B. Lamont, “Parallel simulation of UAV swarm scenarios,” presented at the Winter Simulation Conf., Washington, DC, USA, 2004.

[15] R. E. Weibel and R. J. Hansman Jr., “Safety considerations for operation of different classes of UAVs in the NAS,” in presented at the AIAA’s 4th Aviation Technol., Integration, Operations Forum, Chicago, IL, USA, 2004.

[16] B. Schumann, J. Scanlan, and K. Takeda, “Evaluating design decisions in real-time using operations modelling,” presented at the Air Transport Operations Symp., Delft, The Netherlands, 2011.

[17] C. Nehme, “Modeling human supervisory control in heterogeneous unmanned vehicle systems,” Ph.D. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.

[18] A. A. Mkrtchyan, "Modeling operator performance in low task load supervisory domains," M.S. thesis, Massachusetts Inst of Technol., Cambridge, MA, USA, 2011.

[19] Y. Song, D. Demirdjian, and R. Davis, "Multi-Signal gesture recognition using temporal smoothing hidden conditional random fields," presented at the 9th IEEE Conf. Automat. Face Gesture Recog., Santa Barbara, CA, USA, 2011.

[20] Y. Song, D. Demirdjian, and R. Davis, "Continuous body and hand gesture recognition for natural human-computer interaction," *ACM Trans. Interactive Intell. Syst.*, vol. 2, no. 1, article 5, 2012.

[21] T. B. Sheridan, Telerobotics, *Automation and Human Supervisory Control*. Cambridge, MA, USA: The MIT Press, 1992.

[22] M. L. Cummings and S. Guerlain, "Developing operator capacity estimates for supervisory control of autonomous vehicles," *Human Factors*, vol. 49, no. 1, pp. 1–15, 2007.

[23] M. L. Cummings and P. J. Mitchell, "Predicting controller capacity in remote supervision of multiple unmanned vehicles," *IEEE Trans. Syst., Man, Cybern. Part A, Syst. Humans*, vol. 38, no. 2, pp. 451–460, 2008. Q4

[24] M. L. Cummings, J. How, A. Whitten, and O. Toupet, "The impact of human-automation collaboration in decentralized multiple unmanned vehicle control," *Proc. IEEE*, vol. 100, no. 3, pp. 660–671, Mar. 2012.

[25] J. C. Ryan and M. L. Cummings, "Development of an agent-based model for aircraft carrier flight deck operations," *Model. Simulation J.*, 2014. Q5

[26] J. C. Ryan, A. G. Banerjee, M. L. Cummings, and N. Roy, "Comparing the performance of expert user heuristics and an integer linear program in aircraft carrier deck operations," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 761–773, Jun. 2014.

[27] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Action as Space-Time Shapes," presented at the 10th IEEE Int. Conf. Comput. Vision, Beijing, China, 2005.

[28] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape- motion prototypes," presented at the 12th Int. Conf. Comput. Vision, Kyoto, Japan, 2009.

[29] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," presented at the IEEE 11th Int. Conf. Comput. Vision, Rio de Janeiro, Brazil, 2007.

[30] P. Scovanner, S. Ali, and M. Shah, "A 3-Dimensional SIFT Descriptor and its application to action recognition," presented at the ACM Multimedia, Augsburg, Bavaria, Germany, 2007.

[31] I. Laptev, M. Marszalek, C. Schimd, and B. Rozenfeld, "Learning realistic human action from movies," presented at the IEEE Conf. Comput. Vision Pattern Recog., Anchorage, AK, USA, 2008.

[32] K. Schindler and L. van Gool, "Action Snippets: How many frames does human action recognition require?," presented at the IEEE Conf. Comput. Vision Pattern Recog., Anchorage, AK, USA, 2008.

[33] J. Yuan, Z. Liu, and Y. Wu, "Discriminative subvolume search for efficient action detection," presented at the IEEE Conf. Comput. Vision Pattern Recog., Miami, FL, USA, 2009.

Jason C. Ryan (M'12) received the Bachelor's of Science degree in aerospace engineering from the University of Alabama, Tuscaloosa, AL, USA, in 2007, the Master's of Science degree in aeronautics and astronautics and the Ph.D. degree in human systems engineering both from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2011, and 2014, respectively.

He is currently with Aurora Flight Sciences.

Mary L. Cummings (SM'03) received the B.S. degree in mathematics from the United States Naval Academy, Annapolis, MD, USA, in 1988, the M.S. degree in space systems engineering from the Naval Postgraduate School, Monterey, CA, USA, in 1994, and the Ph.D. degree in systems engineering from the University of Virginia, Charlottesville, VA, USA, in 2004.

She is currently a Visiting Professor with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology and an Associate Professor in the Department of Mechanical Engineering and Materials Science at Duke University, Durham, NC, USA.

QUERIES

Q1. Author: Please provide full bibliographic details in Refs. [1], [2], [3], [6] and [7].
Q2. Author: Please provide report number in Ref. [8].
Q3. Author: Please provide page range in Ref. [11].
Q4. Author: Please provide month in Ref. [23].
Q5. Author: Please provide volume and page range in Ref. [25].