Adaptive Multiscale Optimization: Concept and Case Study on Simulated UAV Surveillance Operations

Irwin O. Reyes, Peter A. Beling, *Member, IEEE*, and Barry M. Horowitz

***Abstract*—This paper presents a resource-usage management scheme called adaptive multiscale optimization (AMO). AMO mitigates dynamically emerging bottlenecks in shared resources by enabling system components to individually select the appropriate operational mode to collectively improve the whole system performance. Components periodically collect and broadcast relevant local measurements to other components, in aggregate, producing an overall view of the performance of the system. These measurements, combined with the preset objective function that encodes the operator's priorities and desired tradeoffs, drive the logic that selects component modes as system state, and user demand changes over time. As a representative use case, we have analyzed its behavior as applied in a three-dimensional object-recognition application involving multiple unmanned aerial vehicles utilizing shared communications and data processing resources. We demonstrate AMO's benefits and tradeoffs through a series of simulator runs, covering such use cases as increasing contention, sudden reduction in system capacity, and varying AMO coordination overhead.**

***Index Terms*—Cost function, optimization, pattern recognition, system performance, unmanned aerial vehicles.**

I. Introduction

INCREASE in data flow in wireless-networked devices results in communication and computation challenges to accommodate that demand. Given sufficient quantities of data to be exchanged and among networked components, the processing and transmission capabilities of a wireless network are exposed as performance bottlenecks. These basic operations not only consume the finite material resources available to the network—processing cycles and bandwidth for computation and communication, respectively, but also introduce latency as a function of the volume of the data. Time-sensitive wireless applications are especially sensitive to these network-resource bottlenecks.

Systems in a variety of application domains currently face, or soon will face, bottlenecks in computation and communication resources as growth in demand outpaces infrastructure. In enterprise and consumer telecommunications, the Cisco Systems Visual Network Index study forecasts that global fixed and mobile-Internet traffic will triple between 2012 and 2017, reaching 1.4 zettabytes annually by 2017 [1]. Streaming video on mobile devices presents unique difficulties in the form of processing under significant power constraints [2] and needing to reliably transmit large quantities of sequential data over a variable noisy channel [3]. Wireless sensor networks, used in such diverse fields as real-time environmental monitoring and homeland security, have similar challenges in that nodes are limited to their sensing, processing, and transmission capabilities by the power and bandwidth made available to them at deployment time [4], [5].

This paper introduces a concept that we term as adaptive multiscale optimization (AMO). Broadly, AMO is a scheme that attempts to address limitations on shared resources through individual components collectively optimizing a system-wide user-defined objective function. These limitations may arise from such effects as varying availability of the shared resources, increasing service demands from the components, and stricter requirements as set by the user. As the state of the system changes and these constraints manifest themselves, it is necessary for components to respond appropriately to ensure that the user's demands are satisfied as best as possible. Failing to do so may result in the degradation of overall system performance due to the suboptimal use of increasingly constrained shared resources. However, implementing AMO introduces additional complexity and costs to the system: components need to share an appropriate level of coordinating information to be aware of resource usage; the end-user must have a means by which to specify system demands; and the components must be able to make the adjustments that affect resource consumption and benefit.

The AMO concept can be viewed as falling within the broad paradigm of dynamic data-driven application systems (DDDASs) introduced by Darema [6]–[8]. The DDDAS paradigm emphasizes a two-way interaction between control or simulation models and data acquisition methods, with system measurements influencing model execution and, conversely, the models directing system measurement. Our principal contributions are 1) the definition of control and measurement concept within DDDAS that is particularly suited to decentralized environments, where system components share a common resource, such as communication link and 2) a detailed investigation of the benefits of the concept in a surveillance scenario involving collections of unmanned aerial vehicles (UAVs or "drones") that, in addition to being an example of a resource-constrained system, is important in its own right, as discussed below.

UAVs have found widespread use with today's military forces, with aircrafts such as the MQ-1 Predator and RQ-9 Global Hawk commonly deployed on long-range reconnaissance missions over Afghanistan and other combat zones

Manuscript received March 03, 2015; revised June 04, 2015; accepted October 19, 2015. Date of publication December 17, 2015; date of current version November 22, 2017.

I. O. Reyes is with the Dell Research, Santa Clara, CA 95054 USA (e-mail: irwin_reyes@dell.com).

P. A. Beling and B. M. Horowitz are with the University of Virginia, Charlottesville, VA 22904 USA.

Digital Object Identifier 10.1109/JSYST.2015.2503395

[9]. These platforms carry an array of high-resolution sensor packages. These sensors produce massive data sets that must be transmitted to information centers and analyzed for mission-critical information, all with very strict requirements on accuracy and timeliness. This has generated much interest in developing computerized automation for the analysis of dense sets of field-collected data. Such automatic systems often depend on the nature of the sensor data collected and the task at hand. Laser-radar (LADAR or equivalently LIDAR) is one such sensor technology deployed on UAVs for the purpose of detecting and identifying objects of interest occluded by environmental artifacts such as clouds, foliage, and camouflage meshes (see, e.g., [10]–[14]). LADAR is especially suited for object recognition as it produces three-dimensional (3-D) scenes that allow analysts to rotate the view and reveal objects otherwise difficult to discern in a standard two-dimensional visual imagery.

To reliably provide value to the decision-makers, these UAV-borne sensing platforms and their supporting infrastructure must be properly allocated, coordinated, and prioritized. Various use cases have inspired an extensive literature on UAV communication and control [15]–[17], adaptive search [18], [19], and dynamic information aggregation [20]. In the context of multi-UAV surveillance applications, the DDDAS framework has been used to define a process by which live system models present on each of the UAVs and a common command center accept measurements about the system, subsequently driving decisions to optimize such metrics as computational efficiency [20], network quality-of-service (QoS) [21], and task completion times [22]. Related work has seen the application of DDDAS or similar concepts to control of UAV swarms [23], [24] and to UAV-based crowd control [25], [26].

Our work parallels this UAV planning literature in that we also focus on improving performance in a multi-UAV surveillance scenario through appropriate policy selections driven by system models and live measurements. However, the AMO formulation is distinguished by the degree to which it emphasizes decentralized and multiobjective decision-making. AMO lacks a strong notion of a central aggregator and policy maker, instead placing this responsibility on each of the component UAVs in the system, which independently set policies for themselves based on the state-information broadcast over the system, ultimately seeking to improve global performance. Additionally, AMO emphasizes improving system-wide performance in terms of the usage of common resources shared among the component UAVs. AMO also incorporates structures that support distributed multiobjective decision-making.

This paper is organized as follows. Section II describes AMO's core concepts. Section III develops the UAV reconnaissance scenario, which is the main subject of investigation. The section also provides a detailed description of the simulation environment that is the basis for the evaluation of system costs and benefits. Section IV provides results and analysis of experiments in which AMO is compared to the naive system under a variety of operating circumstances. Finally, Section V offers conclusions and suggestions for future work.

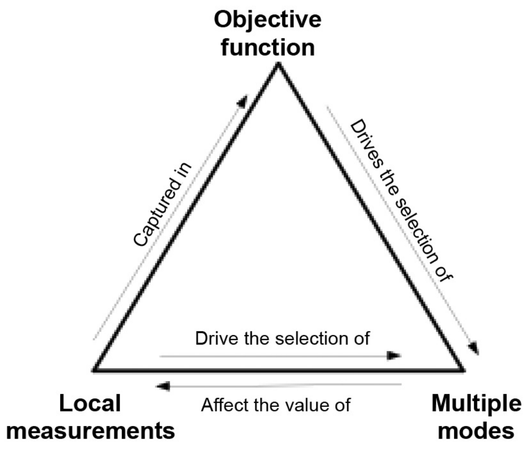


Fig. 1. Relationships between key AMO features.

II. Adaptive Multiscale Optimization Key Concepts

AMO seeks to distribute among system components the task of collectively optimizing the use of shared resources while still meeting user demands. By enabling components to independently make adjustments in pursuit of better resource usage, this optimization scheme is able to adapt as the system varies in terms of scale and state. That is, additions to the system and their interactions during operation do not necessitate fundamental changes to the underlying optimization strategy or system structure; those new components must simply satisfy AMOs measurement and adaptive requirements. An AMO-enabled system contains three core features that allow it to accomplish the stated collective optimization. First, individual components of the system must be made aware of the overall state of the system. This is done through shared local measurements, where components collect relevant data about their current operational state and disseminate this information to the rest of the system. These measurements may include such items as the estimated service costs for a given shared resource or the present working status of a component. Second, a user-defined function to be optimized must be defined and broadcasted to the whole system. Combined with the shared system measurements, this objective function captures the current desired tradeoff between resource consumption and performance subject to constraints placed by the current overall state of the system. Optimizing this objective function yields the best resource-consumption levels, while satisfying the users' needs and current resource availability. This optimization is accomplished by means of the third feature, multimode components. Components utilize the information provided by the system measurements and user tradeoffs to select the appropriate local operational mode that improves the objective function. As the various components perform this guided mode selection, this combined action in aggregate moves the objective function toward its optimal point, improving system performance in terms of the desired costs and benefits as specified by the user. The relationships between these three features are shown in Fig. 1.

As the system operates under AMO, the above multidirectional feedback loop regulates the utilization of shared system resources. Should system performance degrade under current utilization patterns for the shared resources (where performance metrics and acceptable values are defined by the user through the objective function), components respond by dynamically adjusting their individual usage policies to alleviate demand on those common resources. This dynamic optimization stands in contrast to simple greedy resource consumption, where components attempt to use those common assets without taking into consideration the costs and benefits of doing so. However, implementing this logic incurs additional overhead in itself. The following sections discuss the three major interacting features of AMO and the tradeoffs each one presents when introduced to a system.

A. Local Measurements

Shared local measurements in an AMO-enabled system may take on a variety of forms. For one, these measurements could convey information about the current availability of a particular shared resource. Components would use this information to determine whether or not using the shared resource at a given time would incur additional costs such as queuing delay or otherwise adversely affect the ability of other components to properly carry out their functions. Shared measurements may also contain estimates or projections about components' future operational states, allowing other parts of the system to make proper adjustments in anticipation of those changes. For example, if a certain piece reports that it stands a high probability of experiencing downtime in the near future, then the other parts of the system could respond by diverting requests to that asset to alternatives in an attempt to gracefully handle the impending failure. Finally, local measurements could be as simple as the current operational state of a given component, such as whether it is busy or free. Such records would be useful to dependent components that need to determine if it is worthwhile to spend time requesting access to a potentially busy shared resource.

While such local measurements enable systems components to make sound decisions about asset usage, sharing this information incurs some overhead in itself, as this core AMO function necessitates the presence and usage of certain resources. At a minimum, producing and disseminating these measurements require a means to collect the relevant data and to transmit it to interested components. Components must either have measurement capabilities (e.g., record their current state) or local computational facilities to produce accurate estimates and forecast. Once collected, components must share these measurements with one another, requiring the presence of a communications network linking the appropriate systems' parts. If this communications' capability is already a key shared resource in the system, running the AMO information-sharing function on it decreases the number of network slots available for the system to carry out its primary function.

Designers implementing AMO into existing systems must consider three key characteristics of local measurement sharing and their tradeoffs, as described below.

1) *Measurement frequency* is the time between local measurements. Less time between measurements allows for more information representative of the component's current state, but cuts into time spent for component function.
2) *Sharing frequency* is the time between when measurements are transmitted. Less time between updates allows for more representative information, but decreases network availability.
3) *Sharing length* is the time required to transmit measurements. More measurements facilitate finer decision-making, but decreases network availability.

Note that the network tradeoffs are nonexistent if the AMO information-sharing occupies a communications link dedicated to that one purpose.

B. Objective Function

Operators of AMO-enabled systems need a globally visible mechanism for specifying their preferences for individual measures of system performance (e.g., latency and accuracy). In principle, any scheme for expressing preferences among multiple attributes could be used. Without loss of generality, we may consider user preferences to be scalarized into a single objective function in which the variables are individual measures of system performance. In practice, the objective function might be accompanied by constraints (and so define an optimization problem) and might express simple satisficing notions, such as maximizing benefit subject to an upper bound on cost, or minimizing cost subject to a lower bound on benefit.

The objective function must be made available to all components in the system to enable them to make appropriate decisions regarding resource usage. Thus, the objective function also calls for the need to have a communications channel present between all the systems' components. However, the function can be assumed to be constant throughout system operation.

C. Multiple Selectable States

The above two sets of information—shared local state measurements and the user-specified objective function—drive the selection of appropriate operational state at the component level. As such, designers of AMO systems must build components that both have computational capabilities to operate on the objective function and received measurements, and have a set of operational modes from which to select the best way of utilizing resources given for user demands and system state. These modes are specified during the design of the system and must be defined in such a manner that each one has different effects on the consumption of shared assets. For example, appropriate operational state options for a simple generic producer–consumer system would vary the frequency and quantity at which the producer generates. Other potential operational state options involve enabling components to carry out secondary functions otherwise reserved to shared dedicated resources.

Components select modes based on the user's preferences and data gathered from the shared system measurements. This necessitates the presence of some degree of computational capability on the relevant devices in the system to carry out this decision-making. In a simple producer–consumer system, where the producer is a sensor and the consumer is a recording device, applying AMO would require the sensor to have a local controller capable of regulating its output as to prevent the recording device from being overwhelmed, all the while meeting user-set performance preferences.

III. Baseline Scenario and Simulation Model

To demonstrate and evaluate AMO in simulated practice, an appropriate use scenario must be defined involving a system that can be easily formulated in both native and AMO-enabled configurations. As AMO is a distributed decision scheme that manages components' use of resources, interesting examples have a number of independent units consuming shared resources in pursuit of a user-set objective. The key notion of AMO is to manage the consumption of resources to serve the system objective rather than the greedy interests of components. In constructing and analyzing a use scenario, the goal is to determine the tradeoffs AMO presents in terms of system complexity, coordination overhead, and improved resource usage under a variety of operational parameters.

We formulate and implement a simulation of a multi-UAV automatic target recognition scenario in which an arbitrary number of aerial drones collect 3-D LADAR point clouds at various detection rates and transmit the data to a common ground processing station through a shared wireless link. This simulation is implemented in discrete-time, where each step corresponds to 1 s of real time. In the naive non-AMO baseline case, the three major parts of the system—wireless network, ground station, and UAVs—simply perform a single task each: data transmission, point cloud processing, and point cloud generation, respectively.

Tables I, II, III, and IV define the notation used in the simulation model.

A. Shared Wireless Channel

In the baseline scenario, a single unidirectional wireless channel serves as the shared information conduit between UAVs and the ground processing station. The wireless network's sole purpose in this case is merely to carry LADAR data down to the ground. On its own, this wireless channel performs no additional functions necessary for wireless communications such as collision resolution and data retransmission; UAVs independently manage their own network usage to prevent collisions, and it is assumed that the network uses a protocol such as TCP where all data bits are guaranteed to eventually arrive at their intended destinations.

UAVs implement a simple retry-until-succeed back-off protocol to avoid interfering with existing transmissions on the network. When a UAV collects a LADAR image and prepares to send it to the ground station, it first checks the status of the network; i.e., whether it is free or busy. If the network is currently idle, then the UAV continuously transmits without interruption until it finishes. A transmitting UAV holds a monopoly over the channel while it still has data to send to

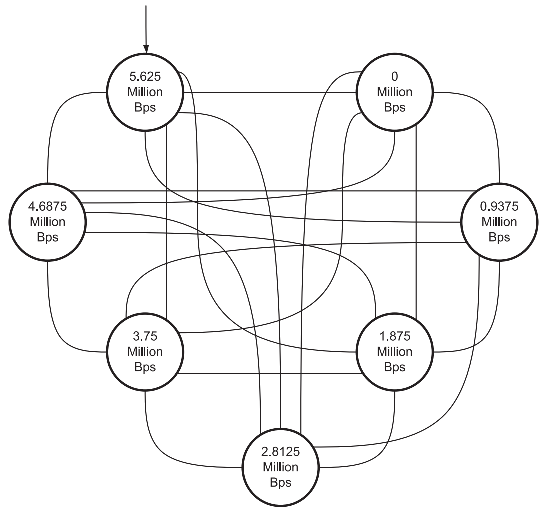


Fig. 2. Markov chain of all seven possible transmission rates.

the ground. If a UAV needs to transmit while the network is occupied with another transfer, then that drone idles for 1 time step (1 s) before retrying. Subsequent conflicts repeat this one-time-slot idle-and-retry scheme until the UAV secures a transmission slot. UAVs do not generate additional LADAR data while transmitting or waiting for the network to become available.

Note that actual wireless networks implement an exponential back-off scheme similar to that specified in the IEEE 802.3 CSMA/CD standard, where each conflicting transmitter randomly idles for an exponentially growing maximum number of time steps based on the number of previous conflicts [27]. A simplified retry-until-succeed protocol is implemented in the simulator instead of the standard exponential back-off scheme, because the 802.3 standard calls for data slot times of length in the order of microseconds, while the simulator's smallest time step is 1 s. Continuously retrying on each time step roughly approximates the worst-case exponential back-off solution.

The shared wireless network exhibits variable transmission rates to capture adverse effects on its availability such as jamming and external congestion. A predefined Markov chain specifies these different transmission rates and the transition probabilities between each possible value, with the initial state set to be the network's maximum transmission rate. The simulation specifies the network's maximum transmission rate as 5 625 000 bps, which is the maximum transmit rate for the tactical common data link network used by the US military for its nonfighter-jet airborne communications [28]. The remaining six states in this Markov chain each correspond to 0/6 to 5/6 of this maximum rate (i.e., equally spaced proportions). The transition probabilities between the values are left as a simulation parameter. Fig. 2 shows the different possible transmission rates and the transitions between each of them (self-transitions and probabilities are omitted for clarity).

The network may then be specified with the symbols.

This system is evaluated under a number of different transition probabilities for M and its effects on t_{qk}, t_{nk}, and other performance metrics are observed.

TABLE I
NETWORK SYMBOLS

Symbol	Definition
m_0	Initial Markov chain state, maximum network transmit rate (fixed to $m_0 = 5625000$ Bps)
m_t	Current transmit rate at simulated time t as specified by M
t_{qk}	Total back-off time incurred during the transmission of the kth LADAR point cloud in the scenario
t_{nk}	Network transit time incurred during the transmission of the kth LADAR point cloud in this scenario

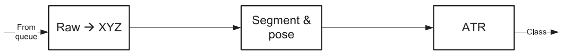


Fig. 3. UAV-based ATR processing stages: XYZ point cloud collection, object segmentation and pose detection, and target recognition.

TABLE II
GROUND STATION SYMBOLS

Symbol	Definition
P_n	Number of independent processing elements
Q_t	Current queue size (in terms of number of LADAR shots) at simulated time t
L	Target library (i.e., set of 3-D object models used to classify LADAR data)
P_x	LADAR points converted to XYZ per second (fixed to $P_x =$ 500 points per second, as per [29])
P_s	LADAR points processed segmented per second
P_c	LADAR points processed with the MPE classifier per second (fixed to $P_c = 7700$ points/s)
B_r	Bytes per raw LADAR point (fixed to $B_r = 2100$ bytes, per [29])
B_f	Bytes per double-precision floating-point number (fixed to $B_f = 8$ bytes, as per [30])
t_{uk}	Time spent in the queue by the kth LADAR shot
t_{pk}	Time spent in active processing by the kth LADAR shot

B. Ground Processing

LADAR images from the UAVs are sent to a ground station to be processed into information useful to human operators and decision-makers. LADAR data arriving through the network from the airborne drones first enter a work queue before they are processed. The ground station is assumed to have multiple processing elements, with the actual number left as a simulation parameter. Each of these computational nodes independently operates on a LADAR data set assigned to it, handling all the processing from its receipt to its classification. At each time step, idle processors check if there are LADAR data sets in the work queue, and if so, remove them from the queue to begin operating on them. This ground computation occurs in three stages: conversion of raw LADAR data to an XYZ point cloud, vehicle segmentation and pose estimation, and automatic target recognition, as shown in Fig. 3. Each stage requires at least one simulated time step to finish, so the minimum amount of time spent in processing is equal to the number of stages.

Note that the simulation does not actually carry out these computations at runtime. Instead, the field-collected LADAR data are already in the different forms needed: whole-scene XYZ, segmented and properly oriented vehicle subclouds, and

TABLE III
UAV SYMBOLS

Symbol	Definition
N	Number of UAVs
λ	Exponential distribution rate (i.e., mean target acquisition time)
k	ID number of the kth LADAR data set generated
p_{ck}	Number of points in the whole-scene LADAR shot
p_{vk}	Number of points after vehicle segmentation
b_p	Bytes per raw LADAR point (fixed to $b_p = 2100$ bytes, as per [29])

TABLE IV
STATE MEASUREMENT SYMBOLS

Symbol	Definition
t_{uk}	Ground queue delay. Dependent on: number of successful transmissions; umber of processing elements; time to process each queued job
t_{pk}	Processing time. Dependent on: processing time; data processing rate; data size
t_{qk}	Transmission retry delay. Dependent on: number of UAVs simultaneously attempting to use the network; data generation rate; time to complete a transmission
t_{nk}	Transmission time. Dependent on: network transmit rate; data size

per-point likelihoods as computed by the MPE classification algorithm. The simulated ground station merely idles for the amount of time necessary to carry out each stage based on input size and processing rate, and then uses the offline precomputed values to produce results. Doing this greatly simplifies the simulation, as the irrelevant details for the implementation of each processing stage are completely abstracted away, while leaving intact measurements of interest to this study: processing delays and correct classification rates.

As with the network, the symbols may be used to fully describe the ground station and relevant measurements.

C. UAV Data Generation

The UAV component in this system handles the generation of LADAR data later used to distinguish and classify vehicles relevant to the mission as set by the user. In this simulation, these LADAR data are derived from a set of images gathered from an airborne sensor: roughly 3200 field-collected whole-scene and corresponding segmented range images. The "generation" of LADAR data is implemented in the simulator as a uniform random draw with replacement over this collection of possible range images.

Multiple UAVs may be present at a given time, with the particular number being a simulation parameter. Each UAV behaves independently to others; no communication between UAVs occurs during any of the three states the drone can assume: idle, target acquisition, and transmission. However, UAVs do transmit data to the ground station through the shared wireless network. Network conflicts are resolved locally using a retry-until-succeed back-off scheme.

At each time step, idle UAVs currently without LADAR data to transmit have a chance to generate data. An exponential distribution with mean interarrival parameter λ governs the random times between detections. Although each drone may have its own mean detection interval, for the purposes of this simulation, it is assumed that λ is the same over all UAVs; i.e., all drones are identically specified.

The symbols specify the entire collection of data-generating UAVs as a whole.

D. AMO Extension

This section details how AMO concepts would be applied to the multi-UAV LADAR automatic target recognition system introduced at length in the previous section. Where the initial non-AMO system follows a simple pass-through structure (i.e., each component has the one task of passing LADAR data to the next one in the line, and eventually presenting classification results to the user), an AMO-augmented variant of this system designates additional responsibilities to components. These additional functions include reporting their current state to the rest of the system or selecting an appropriate data-handling policy, among others, in an attempt to more effectively utilize shared network and ground-computation resources. These new capabilities each fill the role of one of the AMO key features: local measurements, global objective function, and multiple operational states, as discussed in the following sections.

1) Local Measurements: Network Traffic and Ground Queue: There are two resources shared between the UAVs present in this multi-UAV ATR system: the common wireless data link and the ground processing station. Utilizing these assets incurs costs in the form of the following data handling delays, each depending on a variety of system factors.

These resource-utilization delays vary over the course of the operation of the system, so under the AMO scheme, multiple-mode components must have up-to-date information regarding these costs to decide which functional state would be the best response to changing resource availability. Accomplishing this requires both the ground station and network to periodically broadcast their respective local measurements (e.g., network rates and ground service times) to the UAVs. Additionally, as wireless networks do not have transmission queues from which network wait-times can be measured, UAVs need to compute an estimated network retry delay using locally available information before formulating an onboard processing and transmission solution to reduce retry delay and transmission time.

Computing the expected ground station queue and processing times is straightforward enough, as they can be derived from the parameters specifying the ground station and the processing work queue. The following equation gives the local measurements the ground station reports under AMO, depending on the current processing level F of the given LADAR data

$$t_{pk}(\text{F}) = \begin{cases} \frac{p_{ck}}{P_X} + t_{pk}(\text{xyz}) & \text{if F = raw} \\ \frac{p_{ck}}{P_S} + t_{pk}(\text{seg}) & \text{if F = xyz} \\ \frac{p_{ck} * \text{size}(L)}{P_c} + t_{pk}(\text{seg}) & \text{if F = seg.} \end{cases}$$

When the ground station receives a raw LADAR shot to process, the time required to do so is the sum of the time spent in each of the stages from the generation of XYZ points from the raw LADAR returns to the computation of the most likely match in the target library. Presenting partially processed data to the ground station reduces the amount of time the ground station spends to produce a classification match. Indeed, if the ground station receives a fully classified LADAR point cloud, then the ground processing time is 0, as given by $t_{pk}(\text{atr})$. An estimate of the expected ground queue delay is given by $t_{uk}(Q_t) = 1/P_n \sum_{k_i}^{Q_t} t_{pk_i}(F(k_i))$. This delay is defined as the total time required to finish processing each of the LADAR shots in the queue, assuming that this work is evenly distributed among each of the independent ground-processing elements.

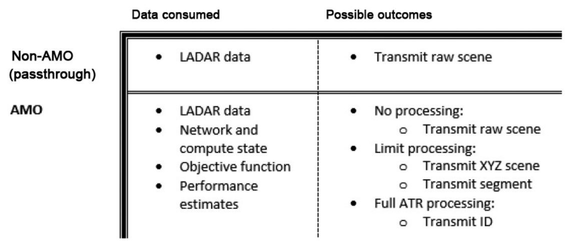
The network records and shares its current transmission rate with the rest of the system. For simplicity, it is assumed that this transmission rate remains constant until the next update. The time to transmit a LADAR data may be computed as follows:

$$t_{nk}(\text{F}) = \begin{cases} \frac{p_{ck} * B_r}{m_t} & \text{if F = raw} \\ \frac{p_{ck} * 3B_f}{m_t} & \text{if F = xyz} \\ \frac{p_{vk} * 3B_f}{m_t} & \text{if F = seg} \\ \frac{B_f}{m_t} & \text{if F = atr.} \end{cases}$$

The network transmission time depends on the level of processing already performed on the data set using the computational resources available onboard each UAV. Network transmission is not carried out in consecutive phases, so only noncumulative transmission times need to be calculated. Note that each stage of the processing sequence on the UAV reduces the amount to transmit in terms of bytes: from large raw LADAR returns to an XYZ 3-D point cloud, to the object of interest cropped out by a segmentation algorithm, to just its computed maximally likely classification.

Computing the expected network retry delay requires certain assumptions regarding the format and size of the data currently being transmitted, as well as the behavior of the other UAVs competing for transmission slots on the shared network. A worst-case scenario is assumed where other UAVs with LADAR data to send are all attempting to transmit whole raw scenes—the largest possible format to transmit. All these raw LADAR scenes are equal in size to the one the given drone has just acquired. A transmitting UAV uses the latest network transmission rate report in computing transmit times. In short, UAVs calculating network delay estimates use the most current local data available to piece together a pessimistic forecast of the expected time required to secure a network slot. Building this forecast is then done in two parts: estimating the number of other UAVs competing for the network and computing the expected delay. A simple internal simulation performed on a transmitting UAV estimates the number of other UAVs attempting to simultaneously use the network. An estimate of how many of the $N - 1$ other UAVs also produced LADAR images in that time span is made by simulating the condition of each vehicle using a draw from an exponential distribution.

Once collected, local state measurements must be broadcasted to the network. Broadcasting frequency t_{bf} and length t_{bl} are system parameters left to the designer to specify. Because only a single-shared wireless network is available in the multi-UAV scenario in question, the distribution of network and ground state information to the UAVs requires the use of some portion of the constrained common network resources AMO is



	Data consumed	Possible outcomes
Non-AMO (passthrough)	• LADAR data	• Transmit raw scene
AMO	• LADAR data • Network and compute state • Objective function • Performance estimates	• No processing: o Transmit raw scene • Limit processing: o Transmit XYZ scene o Transmit segment • Full ATR processing: o Transmit ID

Fig. 4. Summary of data used in non-AMO and AMO cases.

trying to manage. As such, tradeoffs exist between the communications overhead for distributing local measurements, the maximum time between measurements are made, and when a decision is based on them. Measurement updates are assumed to take 2 s, while the frequency at which they occur is left as a system parameter.

2) Objective Function: Global User-Set Tradeoffs: Drones decide their data-handling policies based on the user's cost, and performance tradeoffs codified in the global objective function. This objective function conveys the user's demands for maximum cost and minimum performance, as well as the operator's preference to guarantee either the desired level of cost or performance. This objective function relates the user's desired tradeoff between data handling delay and the probability of correctly recognizing the object in the LADAR image. Also recall that the objective function remains constant through the operation of the system, so unlike the measurements collected by shared resources, the objective function requires no further updates after its initial definition.

For a given detection k, the system will attempt to either 1) minimize delay subject to a lower-bound constraint on accuracy or 2) maximize accuracy subject to an upper bound on delay. The user chooses which problem will be solved as part of the state-selection decision logic that occurs on UAVs after a detection. Note that there may be cases where time-constrained objective functions cannot be satisfied under any data-handling policy. Then the system will attempt to satisfy the user's primary preference as closely as possible.

3) Multiple States: UAV Onboard Processing: Drones gather all the above information received from the rest of the system and use it to decide which is the best combination of onboard computation and network utilization given the current availability of those resources. These onboard computational options form the set of operational states UAVs may assume to best meet user-set delay and accuracy requirements in the face of dynamic resource utilization costs. Fig. 4 summarizes how data are used on non-AMO UAVs compared to the state-selection operation that occurs on AMO-enabled UAVs upon acquiring LADAR data.

AMO-enabled UAVs are capable of performing the same operations as the ground station, albeit at a slower rate to reflect the fact that more computational resources can be placed and maintained on the ground than onboard a drone. Performing some or all the processing onboard, the drone reduces both network usage by effectively decreasing the size of the data to transmit and ground processing demand by introducing more computational capabilities to the system as a whole. Some of

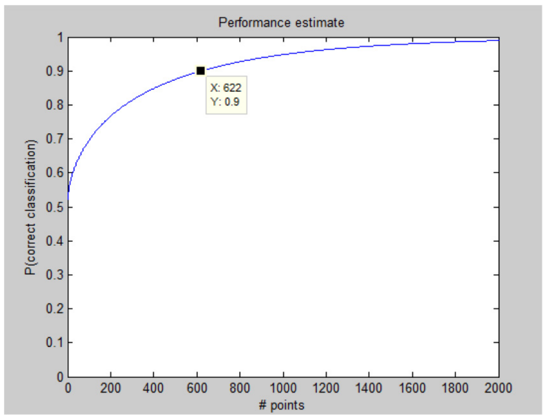


Fig. 5. Example performance estimate: 600+ LADAR measurements to correctly classify a given vehicle versus a given alternative with 90% accuracy.

the computation and communication tradeoffs are evaluated in the experiments detailed later in this section.

Upon acquiring a LADAR image, the AMO-enabled UAV decides the level of onboard computation to perform based on the size of the data, the latest network and ground resource reports, the user's objective function, and precomputed ATR-performance estimates. The first step to accomplish this is to refer to the ATR performance estimate to determine the number of vehicle points to use. The performance estimate relates in the general case the number of points processed to the expected probability of correctly classifying the object in the point cloud. If the user demands a probability of correct classification of 0.90, then 622 vehicle points should be processed according to the performance curve, as shown in Fig. 5.

The next step in the decision-making process is the calculation of the estimated delays associated with each of the four possible operational states the AMO-enabled UAV can assume. These calculations are simply the sum of the same delays that would be encountered after every detection in the non-AMO formulation: network retry delay, network transmission time, ground queue delay, and ground processing time, where the input is a raw LADAR image, in that order. Note that carrying out onboard processing before attempting to transmit increases the length of the time window in which other UAVs may attempt to use the network, so it is necessary to take this into account when estimating the number of competing UAVs.

The last step in the decision-making logic is to choose the policy that corresponds to the minimum amount of delay encountered while satisfying user preferences. If the user places more importance on accuracy than on delay, then the policy with the lowest delay is selected and carried out, as the accuracy constraint is already met through the number of vehicle points obtained from the performance curve. If the user prioritizes latency, then the same policy with the lowest delay is also selected. However, before implementing that policy, the number of vehicle points to process is increased and the associated delay recalculated to determine the maximum point cloud size that would result in a delay just below the user's threshold. This optimizes accuracy subject to a time constraint, as more points processed results in a higher probability of correct recognition.

IV. Experiments

This section explores the performance characteristics of a simulated AMO-compliant multi-UAV sensing and object-recognition application compared to a baseline naive non-AMO variant under a series of different operational paramters. Experiment 1 examines AMO's ability to handle the effects of increasing computational and network load on data-processing efficiency. Experiment 2 tests AMO's response to sudden reductions in shared network capacity. Experiment 3 looks at the effects of varying levels of contention in common ground processing resources.

Experiments 4 and 5 evaluate features exclusive to the AMO-enabled variant of this particular example system: onboard processing on the UAVs and system-wide broadcasts of local measurements. In particular, Experiment 4 determines the level to which the introduction of airborne processing accounts for the benefits of AMO. Experiment 5 establishes the point at which the overhead associated with AMO's measurement-sharing begins to significantly degrade system performance.

A. *Experiment 1: Effect of Increasing Data Generation and Contention*

This test compares the simulated naive baseline multi-UAV system to the AMO-enabled system under different levels of resource contention arising from increasing numbers of UAVs sharing a fixed quantity of ground processors and network throughput. We measure both the end-to-end latencies (i.e., time from detection to results delivered to the ground) and the correct recognition rate. We use an object library of 54 military vehicles arranged in broad functional groups such as mobile armor, aircraft, and missile launchers. The following parameters characterize this experiment: 3000 s window in which UAVs may detect objects, exponentially distributed detections with mean time $\lambda = 30$ s, four independent ground processors, 100 s periods between AMO measurement updates, 2 s to perform each AMO measurement update, and AMO airborne processing operating at 10% the speed of the ground station.

In Fig. 6 (and similar performance charts for subsequent experiments), values along the horizontal axis correspond to the kth detection in the system. The vertical axis represents the end-to-end processing latency for a given detection.

We infer which resources are saturated from the delay trends in Fig. 6. Four components contribute to varying degrees to the processing latency for each detection: network backoff/retry, network transmission, ground processing queue, and ground processing. Network transmission and ground processing times are functions of the amount of individual data to be handled. Network backoff/retry and ground processing queue are functions of the total amount of data over the system. As the total quantity of data in the system scales up (i.e., the number of UAVs increase), we expect network and ground processing idle times to dominate end-to-end processing times.

In these results, note the well-defined floor in latency and increasing delay variability in the baseline case as we sweep from 16 to 128 UAVs. The increased latency variability on top

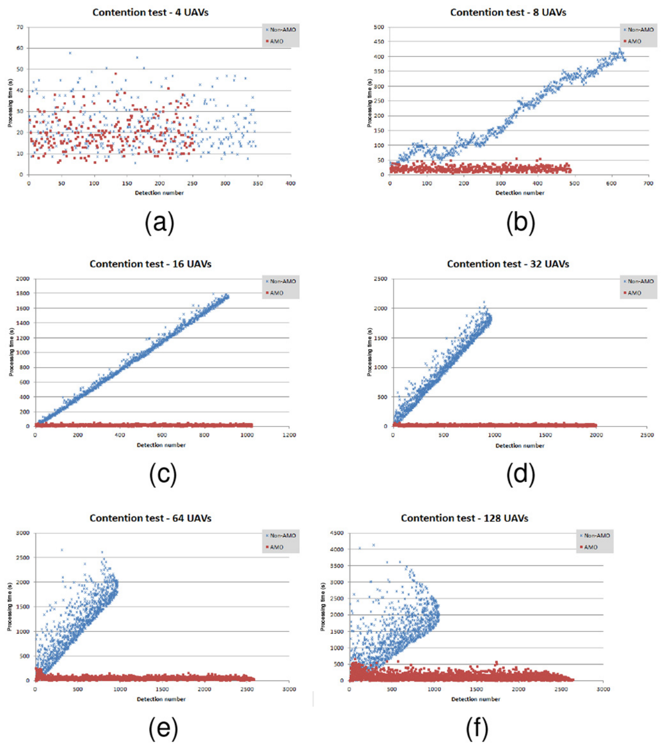


Fig. 6. Baseline versus AMO, increasing AMO update frequency. (a) Baseline classification rate 77.4%; AMO classification rate 78.0%. (b) Baseline classification rate 80.0%; AMO classification rate 73.4%. (c) Baseline classification rate 74.0%; AMO classification rate 75.4%. (d) Baseline classification rate 78.6%; AMO classification rate 79.1%. (e) Baseline classification rate 75.3%; AMO classification rate 77.0%. (f) Baseline classification rate 79.0%; AMO classification rate 78.1%.

of that results from a higher level of network contention as more UAVs attempt to transmit data and results using the same amount of network throughput. The increase in minimum delay is a sign of ground station saturation as the ground processors queue up received data more quickly than they can operate on them. Experiment 3 further explores and verifies this ground queuing explanation.

Now consider AMO's comparative performance with the baseline system in mitigating increasing demands as the quantity of shared network and computational resources remain constant. AMO's update overhead limits the total number of detection events as UAVs enter an idle state while system measurements are broadcast, something that the baseline does not need to do. As the system scales up, however, these measurements drive the selection of alternative processing paths, resulting in the better use of limited resources. The lack of increases in minimum delays over time (as found in the baseline case) indicates no ground queuing delays in the AMO system. Additionally, we observe no tradeoff in improved AMO performance and lower recognition accuracy in the data.

B. *Experiment 2: Effect of Network Limitations*

In this experiment, we simulate a sudden, significant, and permanent reduction in network capacity in the middle of the system's operation and measure its effects on the end-to-end

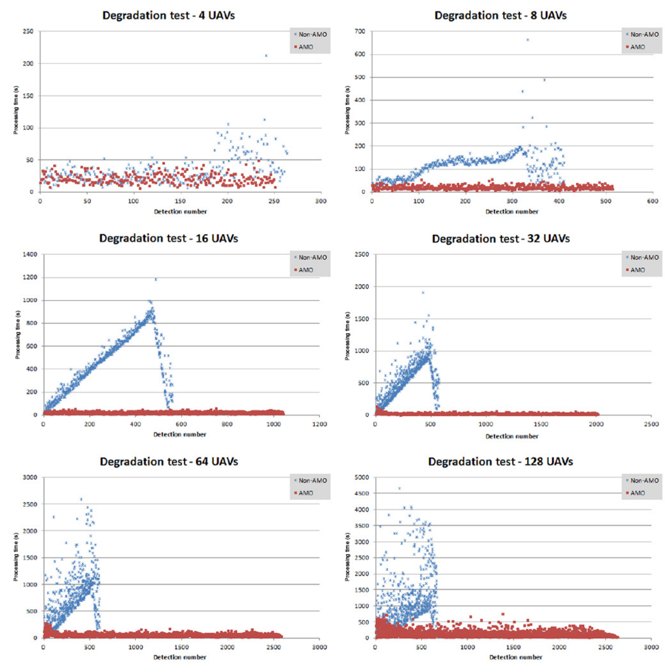


Fig. 7. Baseline versus AMO delays, network degradation after 1500 s.

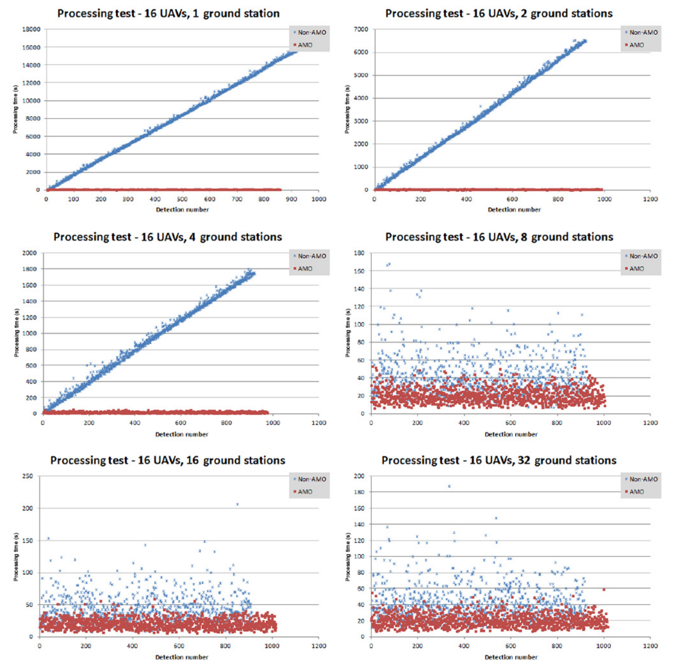


Fig. 8. Baseline versus AMO delays, 16 UAV case, increasing ground processors.

latency. We specify this degradation as a drop in network throughput from its maximum value of 5 625 000 to 937 500 bps (i.e., 1/6 of the maximum) halfway through the system's detection window at $t = 1500$ s. A sweep over a range of UAV counts is performed to gauge how the AMO system compares to the baseline as the degradation occurs under different levels of network contention. The system otherwise behaves identically to that used in Experiment 1: 3000-s detection window, 30-s average exponentially distributed time between detections, four independent ground processors, 100-s system measurement update interval, 2-s update length, and airborne processors with 10% the power of the ground station.

In the baseline 4 UAV case shown in Fig. 7, delay variability increases somewhat after the dropoff in network throughput as the same number of UAVs randomly backoff and retry in an attempt to secure a transmission slot. The decrease in network throughput reduces the number of opportunities to transmit for a given period of time, resulting in more UAVs idling before securing one. Increasing the number of detecting UAVs in the baseline system produces a peculiar effect: before the network degrades, a queue builds up in the ground station (evidenced by the linear increase in end-to-end latencies), but the queue clears after the degradation. In this case, the reduced network capacity has the effect of reducing the rate at which point clouds arrive at the ground station for processing, allowing the ground processors to operate on queued jobs more quickly than new jobs arrive.

Compared to the baseline system, the results show that under these parameters, the AMO formulation is not at all affected by the sudden reduction in network capacity. UAVs in the AMO configuration are able to choose alternative processing options to adapt on the fly to the new level of network availability; whether through partial or all-airborne processing or the same all-ground processing. Further tests limiting network speed and onboard processing still produce outcomes favorable to AMO over the baseline.

C. Experiment 3: Effect of Varying Ground Processing Resources

In this experiment, we examine the effects of different quantities of ground processing units on queue delay and processing latency. We determine whether expanding the ground station's computing capability is more advantageous over implementing AMO in this regard. These tests evaluate 16 and 128 UAV instances of the multi-UAV ATR application, as previous experiments show the effects of ground saturation and network connection begin to emerge starting with the 16 UAV cases. A sweep over a selection of ground processing unit counts for both instances allows us to gauge the effects of increasing processing throughput. The remaining test parameters are identical to the previous experiments: 3000-s detection window, 30.0-s average exponentially distributed time between detections, 100-s intervals between AMO system measurement updates, 2 s to perform those updates, and airborne processors with 10% the throughput of a ground processing unit.

In this particular scenario with the given parameters, the baseline results in Figs. 8 and 9 illustrate that 8 ground processors provide sufficient throughput to handle all the incoming raw ATR data without a ground queue developing; the elimination of the linear increase in end-to-end latencies from the 4 UAV baseline case to the 8 UAV supports this. Instead, an increase in latency variability in the ≥ 8 UAV cases implies that this is the point at which the network is the bottleneck, where the network cannot deliver UAV data down to the ground

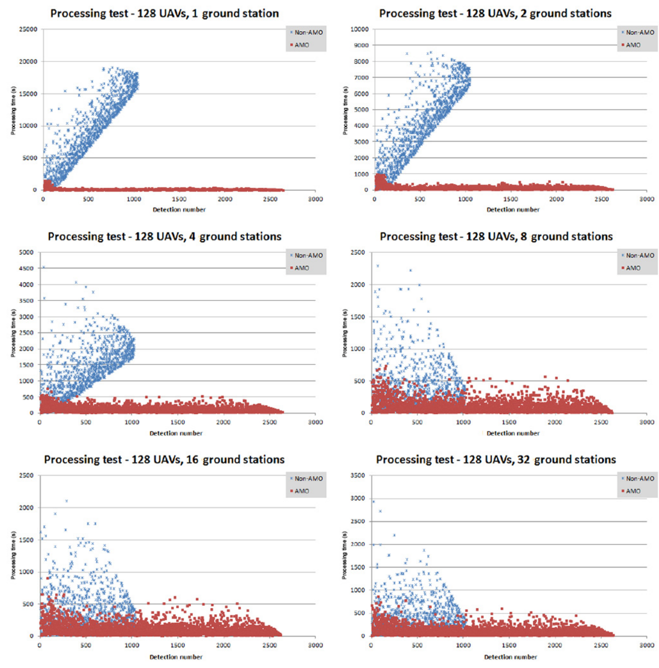


Fig. 9. Baseline versus AMO delays, 128 UAV case, increasing ground processors.

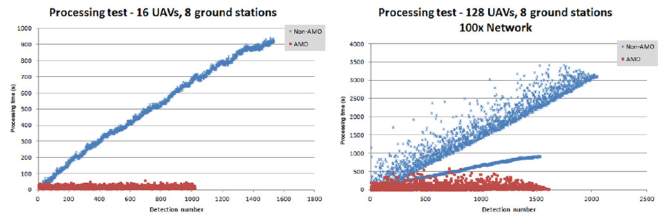


Fig. 10. Baseline versus AMO delays, increasing ground processors, 100× maximum network rate.

rapidly enough to saturate the ground processors. In all cases, however, AMO outperforms the baseline system in terms of both ground queuing delays and network idle delays because AMO further augments this increase in ground processing by reducing network usage through the selection of partial or full airborne data processing modes. This holds true even when the network's throughput is increased 100x, as shown in Fig. 10.

D. *Experiment 4: Effect of Airborne Computational Resources (AMO Only)*

This experiment explores the degree to which the introduction of airborne processing elements contribute to the favorable performance of AMO over the baseline observed so far. That is, we determine whether merely reducing network usage by performing all processing onboard the UAV is sufficient to achieve the same gains as employing a full AMO implementation (complete with system measurements and state selection). For this comparison, we set the baseline as the previously outlined AMO variant of the multi-UAV application. The proceeding results then compare AMO performance to that of a system that conducts all processing on the UAV, eliminating the need for ground processing and only transmitting a significantly smaller data set down to ground operators (i.e., transmit just the result, not the point cloud).

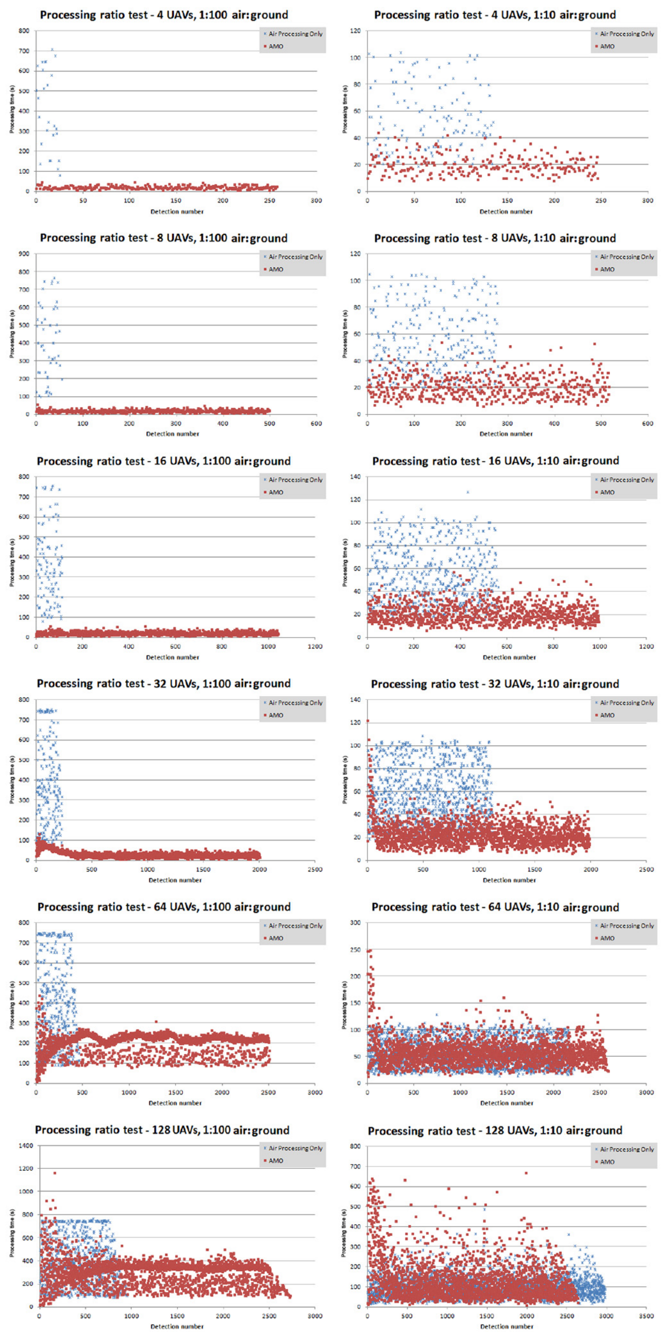


Fig. 11. AMO versus air-processing-only delays.

We test varying UAV counts and airborne data processing throughput rates; a doubling sweep from 4 to 128 UAVs to produce different levels of network contention, and airborne processors with 10% and 1% of the (unused)ground station's processing rate. Other system parameters remain the same as in previous experiments: 3000-s detection window, exponentially distributed detections with mean interdetection time $\lambda = 30$s, 100-s intervals between system measurement updates, and 2 s to perform those updates.

In Fig. 11, note that the naive all-airborne processing variant of this system outperforms the AMO implementation in the 128 UAV cases with air processors operating at a rate 1% to that of the ground units. The mean end-to-end latency in the all-air case is 100 s, while AMO exhibits a mean latency of 130 s. Additionally, the worst-case delays for the all-air case is roughly 500 s, while we observe roughly 700 s for AMO. While this appears counterintuitive to AMO's stated purpose of establishing better global resource usage policies in response to varying levels of resource availability, this result highlights a practical shortcoming of AMO: moving the global system performance toward this improved resource-utilization level requires accurate resource-availability measurements at any given time. The performance characteristics in the higher-UAV cases in Fig. 11 suggest that AMO performs quite poorly before the first system-measurement update once the contention level has reached some steady state; in this case, UAVs initially choose to perform high-contention full-point cloud transmissions to the ground station, because there is no data yet that would drive the mode selection elsewhere.

We draw two key conclusions from these results. First, there exists a "sweet spot" within which the dynamic AMO approach does produce better outcomes than a naive fixed-action approach. This preferred operational range for AMO can be generalized as one where utilizing just exclusive local resources (e.g., the limited onboard processing aboard each UAV) does not always produce the best performance, but is a viable means by which to reduce contention elsewhere in the data processing chain (e.g., in a congested network or a backlogged ground processor). The other conclusion, which addresses the initial motivation for this experiment, is that within this favorable operational environment for AMO, performance gains are indeed the result of AMO's mode-selection logic as a whole, as opposed to merely the increased availability of computational resources in this system.

E. Experiment 5: Effect of Update Overhead (AMO Only)

This experiment seeks to determine the cross-over point where the AMO system state update negates AMO's performance gains over the baseline system. We examine the 128 UAV cases, because it has the highest level of resource contention among the different cases examined and so should best highlight the broad costs of increasingly frequent measurement updates. As before, we specify this scenario with the following common parameters: 3000-s detection window, four independent ground processing stations, and 2 s to complete each system-wide measurement update. We initially set the update period to 50 s (i.e., 4% of the network cycles allocated to AMO overhead) and sweep over key values corresponding to more frequent updates.

The results in Fig. 12 show that in this application, the AMO system measurement update overhead produces only a trivial degradation in system performance until the update begins to consume a majority of network cycles. In the initial case of 50 s elapsing between 2-s system updates (4% of network cycles used for AMO), AMO still demonstrates a significant advantage in overall point cloud processing times over the simple baseline non-AMO case, allowing for faster classifications and more detections within the same 3000 s window. This advantage in

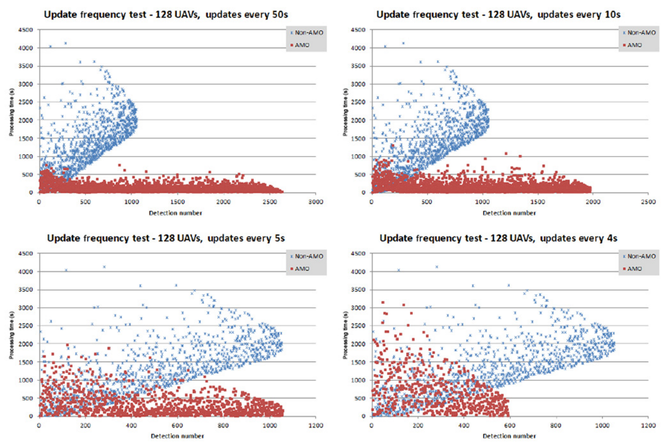


Fig. 12. Baseline versus AMO, increasing AMO update frequency.

processing times still holds at 10-s intervals between updates (20% of network cycles used for AMO). However, the AMO system detects fewer objects than in the previous case, because transmitting UAVs idle more as the measurement update interrupts more frequently, while other UAVs more often select to perform in-air processing using their comparatively limited onboard computing capability.

At 4-s update interval (i.e., 50% network cycles used for AMO), the AMO implementation results in significantly fewer detections than the baseline model while increasing the variability in the end-to-end processing times. Such a high level of AMO overhead effectively reduces network capacity available for actual work. In this environment, the AMO system behaves as if only onboard processing was available, with UAVs spending much time randomly idling until they secure a rare transmit opportunity. This extreme case demonstrates the system-wide cost of AMO and implies that AMO updates must be at a reasonably manageable level ($\leq 20\%$ in this application) to ensure improved performance over the baseline.

V. Conclusion and Future Work

This paper presents a formulation of a resource-management scheme called AMO that enables systems to respond to changing resource availability by means of components selecting appropriate operational modes, where these decisions are driven by periodically updated measurements shared by other components and the user's desired tradeoffs between costs and performance. A multi-UAV automatic target recognition system is specified and used to demonstrate and evaluate these concepts in simulated practice. Experiments comparing the original formulation of the system to its AMO-augmented variant reveal that AMO does indeed produce better detection-to-classification times over the baseline, all without sacrificing recognition accuracy. This also holds true when the availability of shared resources suddenly and drastically decreases; AMO gracefully adapts to such events.

Designers wishing to implement AMO on existing systems must realize that although it does accomplish its stated goal to great effect as it did in this example system, similar success in other cases may be attainable more simply by expanding current resources instead of performing a whole AMO overhaul.

Additional inquiry into the costs and benefits of AMO are required; however, as this is just one particular system formulated to show how such concepts can be applied in practice. A finer-grained simulation or an actual working implementation of such systems are necessary to evaluate the realism and implications of certain assumptions used in this work. Further evaluations of AMO applied to different systems (e.g., autonomous underwater vehicles [31], high-throughput commercial communications, cloud computing load balancing, surveillance with alternative sensors like synthetic aperture radar, etc.) may also very well prove that its tradeoffs and benefits vary greatly on a case-by-case basis, as the individual implementations of its key features—local measurements, objective function, and mode selection—depend on the details of the existing system to be augmented, and the inclinations of the systems' designer performing this task.

References

[1] I. Cisco, "Cisco VNI forecast," Cisco Systems Inc., 2013.

[2] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," in *Proc. 11th ACM Int. Conf. Multimedia*, 2003, pp. 582–591.

[3] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1204–1217, Sep. 2007.

[4] R. Min *et al.*, "Low-power wireless sensor networks," in *Proc. 14th Int. Conf. VLSI Des. (VLSID'01)*, 2001, pp. 205–210.

[5] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl. (WSNA'02)*, 2002, pp. 88–97.

[6] F. Darema, "Dynamic data driven applications systems: A new paradigm for application simulations and measurements," in *Computational Science*. New York, NY, USA: Springer, 2004, pp. 662–669.

[7] F. Darema, "Grid computing and beyond: The context of dynamic data driven applications systems," *Proc. IEEE*, vol. 93, no. 3, pp. 692–697, Mar. 2005.

[8] F. Darema, "DDDAS computational model and environments," *J. Algorithms Comput. Technol.*, vol. 5, no. 4, pp. 545–560, 2011.

[9] P. Grier, "Drone aircraft in a stepped-up war in Afghanistan and Pakistan," *Christ. Sci. Monit.*, 2009 [Online]. Available: http://www.csmonitor.com/USA/Military/2009/1211/Drone-aircraft-in-a-stepped-up-war-in-Afghanistan-and-Pakistan

[10] S. Tan, J. Stoker, and S. Greenlee, "Detection of foliage-obscured vehicle using a multiwavelength polarimetric LiDAR," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS'07)*, 2007, pp. 2503–2506.

[11] L. Wallace, A. Lucieer, C. Watson, and D. Turner, "Development of a UAV-LiDAR system with application to forest inventory," *Remote Sens.*, vol. 4, no. 6, pp. 1519–1543, 2012.

[12] H. Jiang, Y. Su, Q. Jiao, J. Zhang, G. Lixia, and Y. Luo, "Typical geologic disaster surveying in Wenchuan 8.0 earthquake zone using high resolution ground LiDAR and UAV remote sensing," in *Proc. SPIE Asia Pac. Remote Sens.*, 2014, p. 926219.

[13] T. C. Bybee and S. E. Budge, "Textured digital elevation model formation from low-cost UAV LADAR/digital image data," in *Proc. SPIE Defense+ Security*, 2015, p. 94650H.

[14] A. Bird, S. A. Anderson, M. D. Wojcik, and S. E. Budge, "Small swap 3D imaging flash LADAR for small tactical unmanned air systems," in *Proc. SPIE Defense+ Security*, 2015, p. 946008.

[15] Y. Jin, Y. Liao, A. Minai, and M. Polycarpou, "Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, no. 3, pp. 571–587, Jun. 2005.

[16] W. Guo, Z. Zhu, and Y. Hou, "Bayesian network based cooperative area coverage searching for UAVS," in *Frontiers in Computer Education*. New York, NY, USA: Springer, 2012, pp. 611–618.

[17] M. L. Cummings and P. J. Mitchell, "Predicting controller capacity in supervisory control of multiple UAVS," *IEEE Trans. Syst. Man Cybern. A, Syst. Humans*, vol. 38, no. 2, pp. 451–460, Mar. 2008.

[18] Y. Zhao, S. D. Patek, and P. A. Beling, "Decentralized Bayesian search using approximate dynamic programming methods," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 38, no. 4, pp. 970–975, Aug. 2008.

[19] A. Hubenko, V. A. Fonoberov, G. Mathew, and I. Mezic, "Multiscale adaptive search," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 41, no. 4, pp. 1076–1087, Aug. 2011.

[20] Z. Wang *et al.*, "DDDAMS-based crowd control via UAVS and UGVS," *Proc. Comput. Sci.*, vol. 18, pp. 2028–2035, 2013.

[21] T. Chen, R. Bahsoon, and G. Theodoropoulos, "Dynamic QoS optimization architecture for cloud-based DDDAS," *Proc. Comput. Sci.*, vol. 18, pp. 1881–1890, 2013.

[22] G. M. Y. Wei and M. B. Blake, "An operation-time simulation framework for UAV swarm configuration and mission planning," in *Proc. Int. Conf. Comput. Sci.*, 2013, pp. 1947–1958.

[23] R. R. McCune and G. R. Madey, "Swarm control of UAVS for cooperative hunting with DDDAS," *Proc. Comput. Sci.*, vol. 18, pp. 2537–2544, 2013.

[24] R. McCune *et al.*, "Investigations of DDDAS for command and control of UAV swarms with agent-based modeling," in *Proc. Winter Simul. Conf. Simul. Making Decisions Complex World*, 2013, pp. 1467–1478.

[25] A. M. Khaleghi, D. Xu, A. Lobos, S. Minaeian, Y.-J. Son, and J. Liu, "Agent-based hardware-in-the-loop simulation for UAV/UGV surveillance and crowd control system," in *Proc. Winter Simul. Conf. Simul. Making Decisions Complex World*, 2013, pp. 1455–1466.

[26] A. M. Khaleghi *et al.*, "A DDDAMS-based UAV and UGV team formation approach for surveillance and crowd control," in *Proc. Winter Simul. Conf.*, 2014, pp. 2907–2918.

[27] *ETHERNET*, IEEE Standard 802.3, 2008.

[28] J.I. Group, *Jane's C4I Systems*. London, U.K.: Jane's Information Group, 2010.

[29] P. Cho, H. Anderson, R. Hatch, and P. Ramaswami, "Real-time 3D LADAR imaging," pp. 62350G–62350G-12, 2006.

[30] *Standard for Binary Floating-Point Arithmetic*, IEEE Standard 754, 2008.

[31] K. Song and P. C. Chu, "Conceptual design of future undersea unmanned vehicle (UUV) system for mine disposal," *IEEE Syst. J.*, vol. 8, no. 1, pp. 43–51, Mar. 2014.

Irwin O. Reyes received the M.S. degree in systems engineering under the supervision of Peter Beling from the University of Virginia, Charlottesville, VA, USA, in 2011.

He is a Staff Member with Dell Research, investigating concepts in usable security. He worked on the analysis and development of the Aegis Ballistic Missile Defense system at the Johns Hopkins Applied Physics Laboratory, Laurel, MD, USA. His research interests include systems simulation and evaluation, machine learning, biometric security, and election security.

Peter A. Beling is an Associate Professor of Systems and Information Engineering with the University of Virginia (UVa), Charlottesville, VA, USA. His research interests include the area of decision-making in complex systems, with emphasis on adaptive decision support systems and on model-based approaches to system-of-systems design and assessment. His research has found application in a variety of domains, including mission-focused cybersecurity, reconnaissance and surveillance, prognostics and diagnostics systems, education and training, and financial decision-making.

Barry M. Horowitz is the Walter Munster Professor and Department Chair of Systems and Information Engineering at the School of Engineering and Applied Sciences, University of Virginia, Charlottesville, VA, USA. Prior to joining the university in 2001, he was the Founder, Chairman, and CEO of Concept Five Technologies, Mclean, VA, USA, an e-business systems development company, and served in a variety of positions including CEO with the Mitre Corporation, Dranesville, VA, USA, from 1969 to 1996.

Prof. Horowitz was elected as a member of the National Academy of Engineering in 1996 and was the recipient of the U.S. Air Force Exceptional Service Award in 1992.