Autonomous Robot Control via Autonomy Levels

Lawrence A.M. Bush and Andrew Wang

Autonomous systems need to exhibit intelligent behaviors while performing complex operations. These systems will be deployed in clusters to perform collaborative missions with human supervisors. Autonomous systems will take on expanded roles, requiring higher-order decision-making capabilities supporting autonomous mission planning, resource allocation, route planning, and scheduling and execution of coordinated tasks.

In the future, unmanned systems will gain decision-making intelligence that enables them to autonomously operate in clusters to perform collaborative tasks. For successful field deployment of unmanned systems, operators will need confidence that autonomous decision making leads to optimal behaviors in uncertain environments. Adjustable autonomy technologies, concepts, and simulation environments evaluating teaming behaviors will enable researchers to develop these systems. Network and sensing advances have created the opportunity for increased mission performance, but at the expense of greater complexity in sensor coordination and analysis. Current unmanned systems are typically teleoperated and are labor intensive, relying on human operators and their decision-making capabilities to perform mission tasks.

Today, mission and sensing complexity that are managed through increased automation of lower-level functions (e.g., mechanical-system controls) help operators focus on higher-level decisions. The lower-order decision-making algorithms under development include those for waypoint following and collision detection and avoidance. Some of these algorithms have been incorporated in operational platforms.

Deployment

A team of unmanned aerial and ground vehicles might be deployed in a natural disaster relief scenario as depicted in Figure 1. In this example, a major earthquake has damaged buildings, roads, and bridges, and disrupted communication, power, and water distribution services. A nuclear

energy facility also requires an immediate response. Relief convoys need to deliver supplies throughout the affected area. A team of autonomous aerial and ground scouts supervised by operators in a mission logistics vehicle is dispatched to survey the damage. The algorithms need to determine the safest path for the relief convoy to travel to reach its destination in the minimum amount of time.

Figure 2 presents an architecture for functions that a multiagent autonomous team would need to perform in this scenario. Human mission operators in the logistics vehicle would enter high-level goals, system constraints, and policies into the system. Resource allocation algorithms would be employed to develop a system composition based on the mission objectives and the appropriate available resources, including which platforms to deploy and their sensor payloads, processors, and communication capabilities. Planning algorithms would develop platform route plans for optimum survey coverage, and scheduling algorithms would determine flight or road plans for autonomous scout vehicles to follow. The execution manager algorithm would see that the mission is performed and goals are met.

It is the job of the logistics planner algorithm to choose the actual sequence of waypoints so that it balances and reduces the risk among each component of the mission. However, to make well-informed decisions, the planner will need the scouts to gather additional data on areas the logistics vehicle may cross in the future. The scout dispatcher algorithm determines where to send the scouts, given the plans currently considered by the logistics planner algorithms. The execution of each plan carries with it some risk uncertainty, which is transformed into map uncertainty. In other words, the scout dispatcher determines map locations that contribute

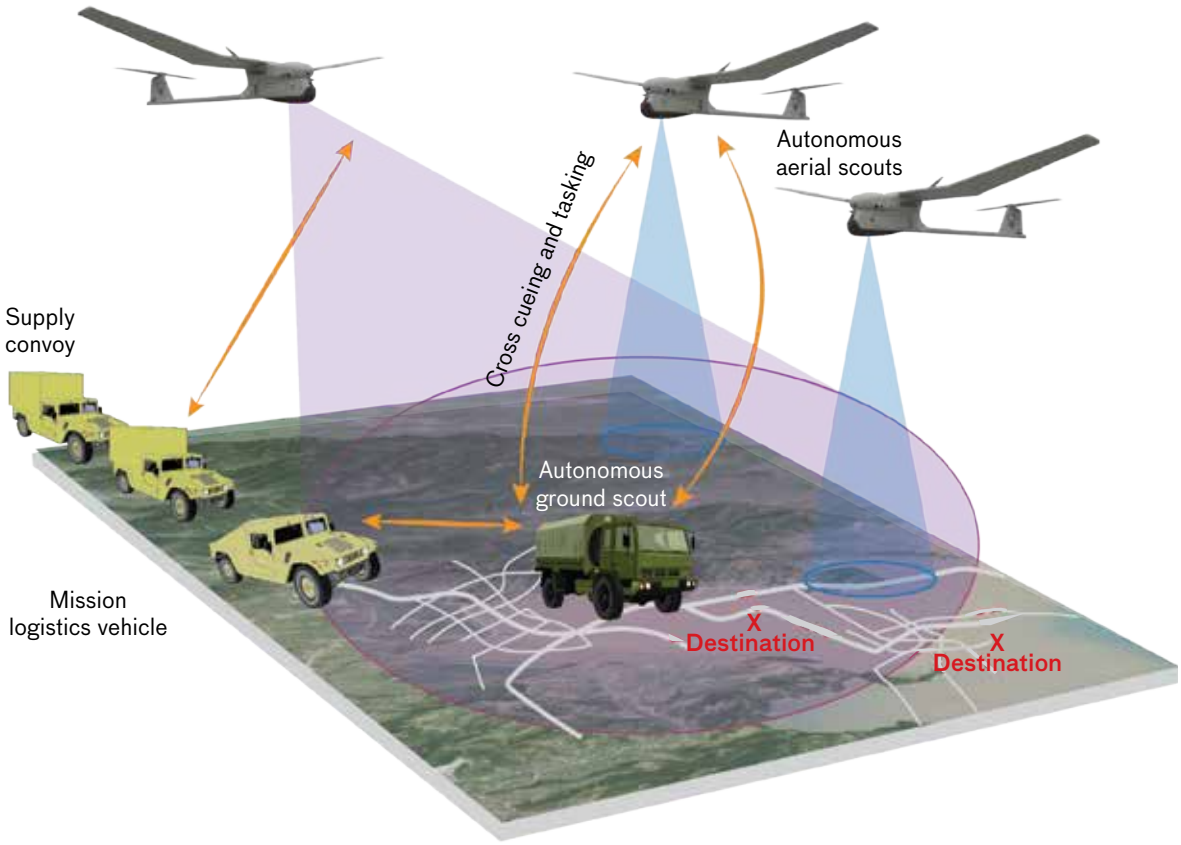


FIGURE 1. In this earthquake relief scenario, a convoy needs to deliver supplies to those in need by the safest path. A team of aerial and ground scouts supervised by operators in a mission logistics vehicle is dispatched to survey the damage and provide real-time route safety information.

most to uncertainty. It then tasks scouts to survey these areas to disambiguate candidate plans. Each scout's planner accepts as inputs these areas and a time limit for reporting results on each area. The executive must also receive the current risk estimate or "belief" for the relevant area. The planner runs an adaptive sampling algorithm that is trained to calculate the flight path that achieves the highest expected information gain within the time allotted. As sensor measurements are analyzed and shared, the belief update module incorporates them into the risk belief, and at the end of a sensing task, the scout reports the updated risk belief to the logistics executive.

For successful system field deployment, operators need confidence that autonomous decision making leads to optimal behaviors, especially when carried out in uncertain environments. A number of concepts and technologies are the subjects of current research to optimize planning in uncertain environments. As shown in Figure 2, one strategy is to equip functional modules with risk-assessment capabilities. This strategy would allow adjustment of the system's autonomy levels according to individual risk acceptance. At any time, an operator can monitor autonomy algorithm decisions, augment or modify algorithm inputs, or take over full manual control of selected logistics vehicles.

Another strategy is the incorporation of rigorous verification processes within the autonomous system algorithm architecture. Algorithm results or plan feasibility would be verified against operator risk acceptance as well as mission resource costs and system performance or autonomous behavior expectations. If conditions are not met, the system may request new plans or request/task subgoals to reduce uncertainty,

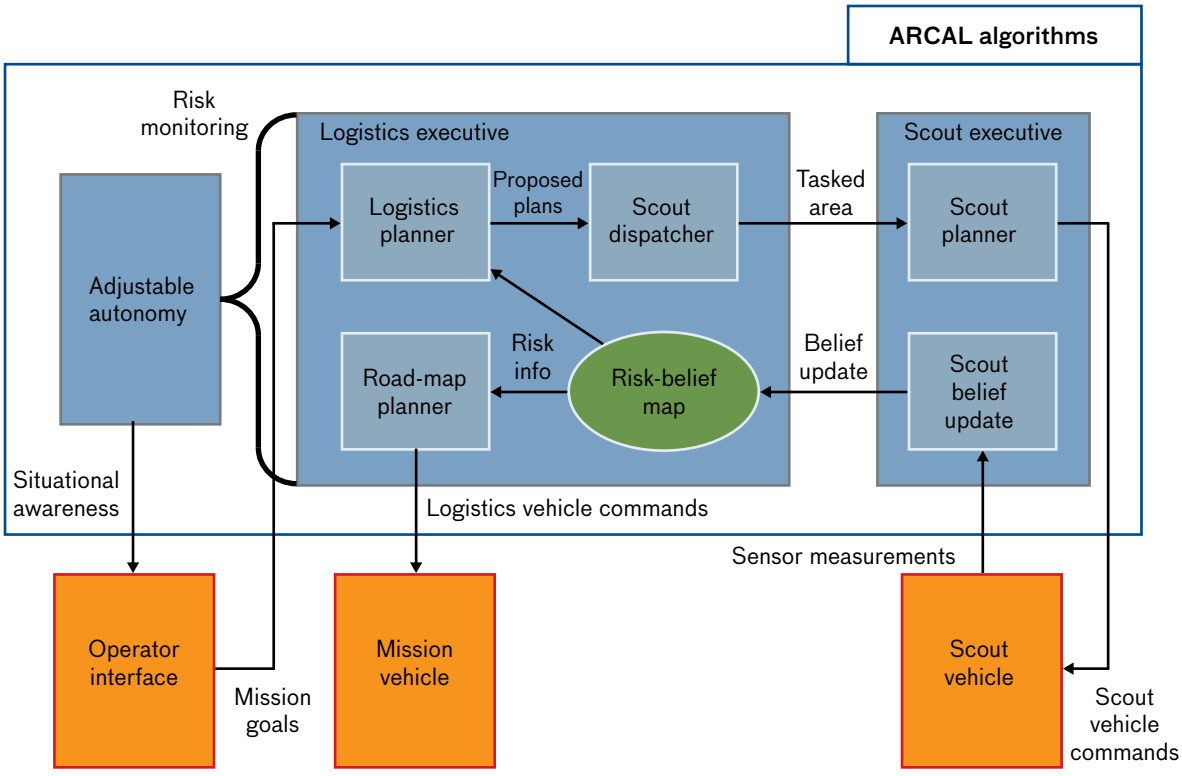


FIGURE 2. Within the generalized multiagent autonomy architecture of the Autonomous Robot Control via Autonomy Levels (ARCAL) system, the logistics executive contains several submodules. Two of them are the high-level logistics planner and the low-level road-map planner, each containing a risk-assessment functionality that operates on the risk-belief map. Together, these submodules determine the course of action for the logistics vehicle. The logistics planner accepts mission goals from the operator and generates sequences of waypoints, producing a high-level road map that will achieve the mission goals. Then, the road-map planner finds the actual path taken between waypoints.

including tasking additional scout runs for surveillance information, satellite imagery, or other sensor data.

The performance of unmanned systems in collaborative tasks has not been thoroughly tested in uncertain environments. In fact, such testing requires entirely new methods. Accurate behavioral simulation and metrics (currently unavailable) are both vital to completing a successful mission. The Autonomous Robot Control via Autonomy Levels (ARCAL) project seeks to establish a robust planning architecture for collaborative, multivehicle autonomous systems by testing system performance in uncertain environments.

Autonomous Robot Control via Autonomy Levels

ARCAL brings together researchers and engineers from MIT campus and Lincoln Laboratory. Researchers from the Model-based Embedded and Robotic Systems (MERS) Group at MIT have developed risk-based adjustable autonomy and task-directed adaptive sensing systems—two fundamental components of ARCAL—that can autonomously coordinate multivehicle missions with an overriding human operator. Engineers from Lincoln Laboratory's Tactical Networks Group developed a simulation environment to evaluate autonomous collaborative behaviors and to determine how well adjustable autonomy operations meet operator expectations.

The project specifically tests risk-based adjustable autonomy with task-directed adaptive sensing technologies and concepts to determine how tasks can be completed at different levels of autonomy. ARCAL utilizes a novel simulation environment to test collaborative autonomous algorithms and team behaviors prior to field deployment. Adjustable autonomy algorithms and functions drive simulated unmanned systems in three-dimensional (3D) platform models that include dynamic environments similar to real-world conditions.

Adjustable Autonomy with Risk Assessment

Adjustable autonomy hopefully combines the best elements of human intuition with computational pragmatism. Challenges in creating a truly synergistic relationship between humans and computers and sensors, given human variability and the limitations of computer logic, have tended to obscure an exact definition of adjustable autonomy as a concept. In its most basic form, an adjustable autonomy system makes two kinds of decisions: what future actions are optimal and how can the human operator best be engaged? Both of these decisions depend on risk estimates and mission objectives, with risk explicitly incorporated in the planning process. Given the mission's logistical plan, risks posed along each step of the plan are probabilities integrated over each mission goal. The configuration and distributions of these risks should inform optimal human engagement. An adjustable autonomy architecture optimizes the risk assessment and mission planning process to provide situational awareness (SA), keeping the human involved at the appropriate level of detail for each mission component.

ARCAL's contribution to adjustable autonomy is to encode risk throughout the decision-making process. In practice, scout aerial vehicles and other sensors can improve risk awareness throughout the mission. Scouts are specifically deployed to improve risk mapping and refine decision making. Algorithms guide scouts toward high-value information that will help identify the low-risk pathways for future components of the mission. The scouts are first tasked with informational reconnaissance relative to the logistics planning. ARCAL performs some tasks offline (learning and simulation) to minimize the amount of online optimization needed.

Architecture

Algorithmic modules within the artificial intelligence architecture enable the incorporation of risk information and the involvement of a human operator. Modules include the logistics executive, the scout executive, and the adjustable autonomy module. These components interact with the logistics vehicle, the scout vehicles, and the human operator, as depicted in Figure 2.

The logistics executive planner chooses the actual sequence of waypoints to reduce risk within each component of the mission. To make well-informed decisions, the planner needs the scouts' information on potential vehicle paths. The scout executive dispatcher determines where to send the scouts, given plans currently under consideration by the logistics planner. Scouts then survey these areas to disambiguate candidate plans. The planner runs an adaptive sampling algorithm trained to traverse the path that achieves the highest expected information gain within the time allotted. As sensor measurements arrive, the belief update module incorporates them into the risk belief.

In a nonadjustable autonomy architecture, the human operator would interface directly with the logistics executive; here, the adjustable autonomy module mediates their interaction. This module continuously monitors the risk associated with each mission component according to the entire state of the logistics executive. It tracks the possibility that each component's risk might exceed user-specified thresholds. As these risks evolve because of additional planning and updated risk beliefs, adjustable autonomy may request human intervention for particular mission components. Thus, while the human operator still specifies mission goals to the logistics planner, he or she now has an interface to override different components of the logistics executive at varying levels of control. Together, all of these modules provide a rational, risk-based operator interface.

Risk Assessment

A key capability of our system is assessing risk relative to the overall mission goals. Here, risk is defined as the likelihood that a logistical plan will or will not achieve each and every goal, where a goal may involve driving an emergency, utility, or personnel transport vehicle to a needed location. Plan success is provisionally defined as the probability of success in all parts of a plan. The risk assessment problem then becomes as follows:

> Given a path plan that nominally achieves overall mission goals and a belief map of the environment, we compute a distribution over a path's success probability, that is, the probability that a ground vehicle can successfully traverse that path. We cannot know the true path-success probability because we do not have a true map of the environment. However, we possess a belief map that models the location of features and obstacles within the environment as well as our uncertainty about them. We may know, for example, that a certain type of obstacle exists in a general vicinity but not know its precise location and threat level. Thus, we must compute, and our algorithms must operate on, a probability distribution over the success probability, i.e., a risk distribution.

Given this definition of risk distribution, we describe below how to represent risk in a belief map. With this definition, we can build paths over the map and devise a risk distribution for each path. Finally, we explain how the scout measures and updates the risk-belief map. For further information on mission time constraint risks, see the appendix titled "Temporal Risk Assessment."

The belief map is represented by a grid of square cells. Each cell contains a distribution over the probability of success if the vehicle traverses that cell in any direction, independently of all other cells. This interpretation allows us to use the Markov assumption (described below) when constructing paths from sequential cells. In our belief map, we parameterize each cell with a mean and variance to represent a beta distribution. Not only does the beta distribution admit an intuitive interpretation, but its parameterization is also appealing for real-time calculation.

The belief map's form makes it relatively straightforward to compose paths from cells; the distribution of the resulting path is an approximation. We rely on the Markov assumption that the probability of successfully traversing a certain cell is independent of the probabilities for other cells. Then, given a path of cells for which successful traversal is a random variable, the success probability for the entire path becomes the product of each of the independent cell traversal probabilities. Unfortunately, the true distribution for the entire path is not a beta distribution and cannot be analytically computed, so we approximate it as a Gaussian distribution, parameterized by a mean and variance.

The testing simulation must also incorporate environmental obstacles into the belief map. The sensor has algorithms for detecting and characterizing features of the environment. The scout's camera, for example, would be interfaced with a pattern-recognition application for road fissures that would then communicate the fissure parameters to the probability of success estimate. If the camera's resolution is characterized by a variance, then the fissure's risk distribution can be characterized, and the information is encoded into the occupied grid cells, effectively distributing the fissure's risk over the area it occupies.

In summary, the simulation formulates risk as a distribution over a path, given a risk-belief map. The map is gridded into cells, each of which contains a beta distribution. Paths are sequences of adjacent cells, with risk distributions represented as truncated and scaled Gaussians.

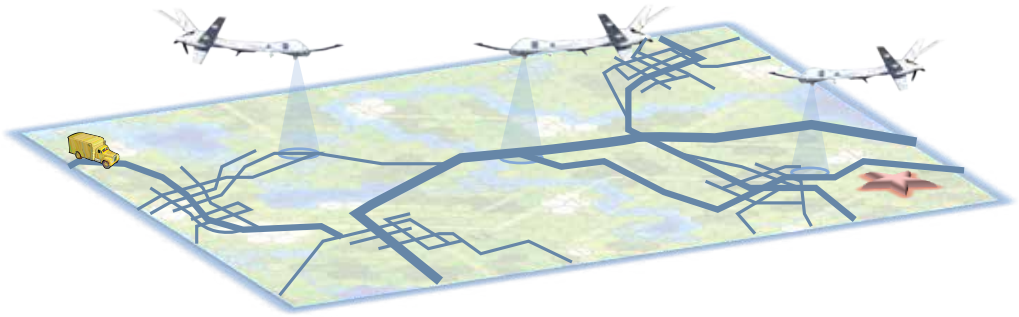


FIGURE 3. In this rescue scenario, the logistics executive tasks a three-unit aerial scout team to identify traversable paths from a set of possible paths. The scout planner algorithm allocates resources in response to the need for locating a safe route and quantifying the path risks to the rescue vehicle and the mission.

Scout Executive and Planner

The scout executive obtains more detailed scans of certain areas that could yield safe routes for the logistics vehicle, as illustrated in Figure 3. While the logistics executive tasks the scouts with examining certain areas, it would be inefficient for scouts to equitably traverse each area (i.e., by spending the same amount of time in each area). For example, a human operator would immediately fly/drive a vehicle to the most uncertain areas in order to gain the most value from reconnaissance. The scout has only limited time to complete reconnaissance and report back to the logistics executive. The scout planner algorithm incorporates scout observations and directs scouts to collect data that optimally reduce risk uncertainty for the logistics vehicle.

Figure 4 highlights the scout planner portion of the ARCAL architecture and various components of the scout planner algorithm. The scout planner dictates the policy that governs paths the scout should take. The policy is typically encoded as a value function. A typical scout scenario, however, is so computationally intensive that the value function would require the processor to have unreasonable volumes of storage space. The iteration process thus approximates the value function to yield nonoptimal but reasonable solutions. Calculations are performed offline, and the approximate solution is stored in an approximate value function. When the time comes for the scout to execute online actions, it further reoptimizes the value function according to its particular situation, given computational constraints.

ARCAL's scout planning problem is formulated as follows:

- The scout dispatcher tells the scout which subset of the full map needs to be surveyed to reduce uncertainty in the risk belief.
- This subset is represented as a set of grid cells. Each grid cell is associated with a prior risk distribution.
- The scout's goal is to fly a path over the area in an allotted time such that it maximizes the total reduction in variance over these grid cells. (The total variance reduction is the sum of all variance reductions in each grid cell.)

ARCAL uses the general framework of the Markov decision process (MDP) to model the problem and approximate dynamic programming (ADP) to solve it [1, 2]. MDPs operate on discrete time steps. When an MDP executes an action from a "current" state, there is a probability of transitioning to a "next" state in the next time step, and the expected reward associated with that transition is calculated. The ADP algorithms generate policy solutions that assign an action to each state of the MDP. The value of a state under a specific policy is the expected sum of rewards obtained when the policy is followed. The objective is to find an optimal policy that maximizes the value of every state.

For a policy to be optimal, it must choose actions that maximize the expected value of the subsequent state. In other words, the optimal action moves to the next-best state, and then plans from that new state. The optimal policy derives from solving for the optimal value function. For the scout planning problem, we define the following components of an MDP:

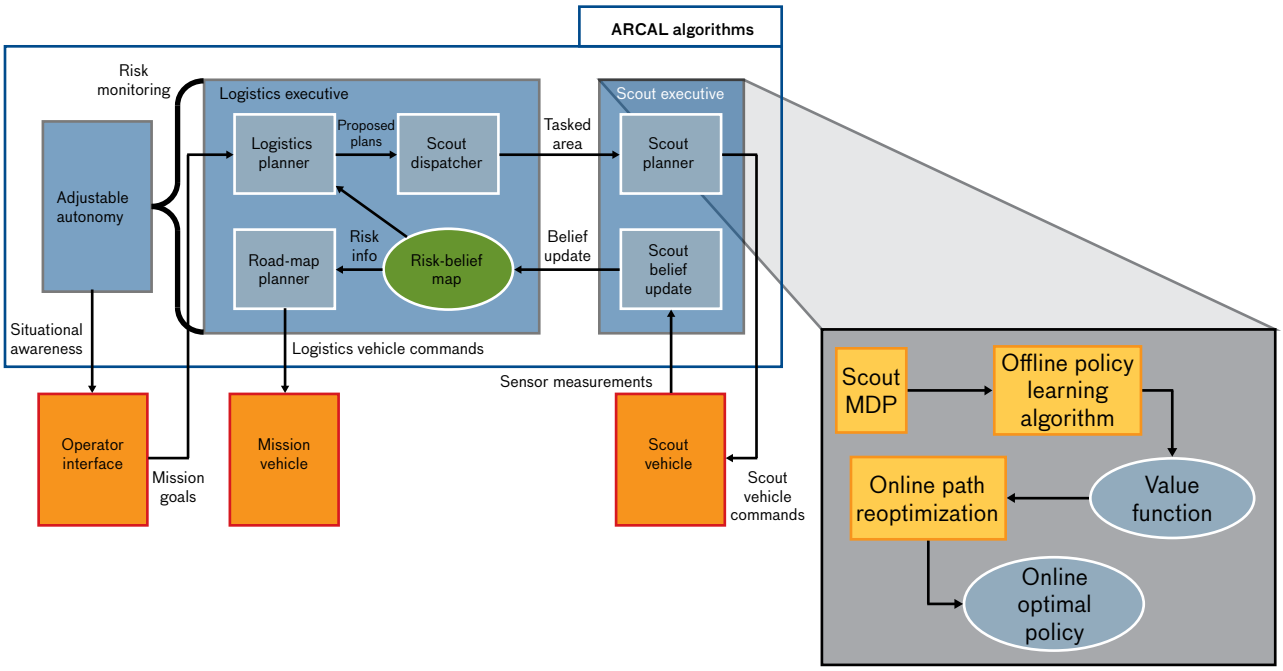


FIGURE 4. The scout executive architecture shows that the goal is to identify paths that maximize information. Scouts model the problem and goal as a Markov decision process (MDP) and learn an offline policy in the form of a value function. Policy dictates what the scout will do next. The policy may be improved online during the mission.

- The state includes the vehicle location and belief map (i.e., risk distributions over the relevant grid cells).
- The action set refers to scout vehicle movement options. In our problem, the available actions are left, right, and straight for any grid cell.
- Reward is defined as the total reduction in uncertainty for the relevant grid cells. It is calculated by taking the sum over all reductions in variance resulting from the Kalman filter–based updating of the state from the scout's observations.

It should be noted that the state space includes the belief map in addition to the location and pose (three location dimensions plus the "pointing" direction of the scout). This information is a necessary part of the state because the reward in transitioning between states is solely defined by the reduction in variance. Given high initial uncertainty, traversing new cells decreases variance more than moving between cells whose uncertainty is already low. Including the belief map makes the state space continuous.

Figure 5 shows the offline scout algorithm. The scouts use ADP to create a policy for acting in the world. A policy interlaces states and actions by instructing scouts in given situations. The scout's state space includes not only location, pose, and risk but also uncertainty in the risk-belief map. Computing the value function is computationally intensive, so ARCAL approximates it offline through value iteration before the mission starts. The approximation simulates scout reconnaissance of high-value areas and saves snapshots of the simulation as data points in a table (Q table or Q function). We then generate an approximation architecture on each iteration by regressing over these data points, taking representational uncertainty into account [3].

Action-Selection Algorithm

A central question is how to choose which actions to reevaluate. As stated previously, the offline algorithms generate a state-action value function and a tree of paths. The simulation then decides how deeply (the number of steps forward) and broadly (the number of actions) and the number of samples per action the reevaluation should be. We assume that we do not have enough time to reevaluate every action over the planning

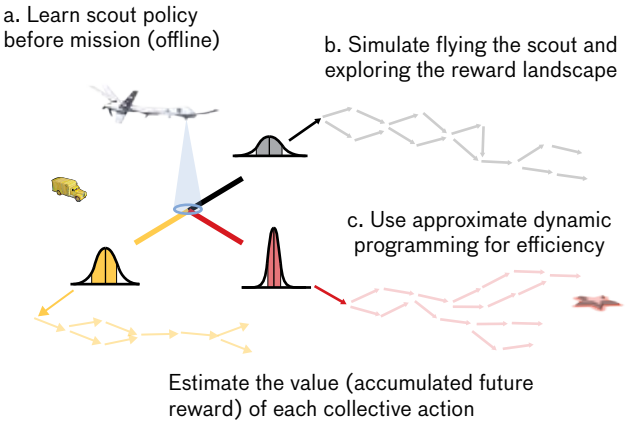
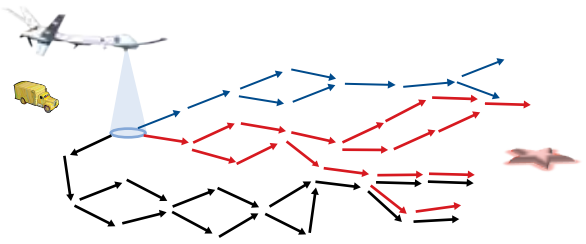


FIGURE 5. The scouts use approximate dynamic programming to learn a policy by simulating reconnaissance and updating the value of each action. Here, a decision to turn left would follow the black arrow that might eventually follow the gray route; going straight would follow red paths and turning right would follow the yellow paths.

horizon and thus only evaluate actions with promising outcomes, given uncertainty about those outcomes. This selection uses the offline state-action value function. The function provides an estimate of the value (future cumulative reward) for each action, assuming the current state. The estimate actually includes a distribution described by a mean and variance. The distribution captures how well we know a given value: a high variance distribution means that we do not know the value very well, and a low variance distribution means that we know it precisely. We select a sample from these distributions, one for each action, and then choose the action with the highest sample value. If one of the action distributions consistently produces a high sample value, we know we have little reason to evaluate other options. However, if there is a state-action value distribution with an especially high variance, the action will sometimes produce a sample with the highest value even though its mean is lower. This phenomenon mirrors the probability that said action is the best, given what we know. In other words, we explore the actions in proportion to their optimality and to our certainty about this parameter (i.e., we determine which actions to reevaluate by representing the uncertainty about their true value). This uncertainty distribution is used to select actions (Figure 6) for reevaluation that appear to be good, but uncertain. We may also evaluate less optimal plans whose true value is subject to high variance.

To summarize, our method allows us to use offline knowledge and processing to guide our scouts online—the offline policy informs the additional online processing. We can exploit both online and offline control processing in a complementary way.

The action-selection process describes a family of algorithms because changing the search horizon and branching factor fundamentally changes the algorithm. For example, if we use a very small branching factor with a long horizon, the algorithm closely resembles the rollout algorithm. Rollout, a long-standing algorithmic method originally developed to evaluate moves in the game of backgammon, was repurposed to evaluate MDP policies in general [4, 5]. On the other hand, using a short horizon with a large branching factor closely resembles model predictive control. The optimality of different configurations depends on different applications and the stage of the mission. For example, toward the end of a mission, it could be helpful to use a wider search (larger branching factor) to make sure that we appropriately consider the end goal. Further extensions to our algorithm may include using different branching factors at different levels of the search tree. For example, it is easier for a function to capture long-term objectives than short-term details. This small extension would



1. Which collection path contributes the most to keeping the rescue vehicle safe?
2. Are we sure?
3. Reevaluate uncertain and viable alternatives.

FIGURE 6. Some steps need to be evaluated in a timely in-flight manner online. The online process describes a family of algorithms that change the search horizon and branching factor and determine which steps need to be reevaluated online.

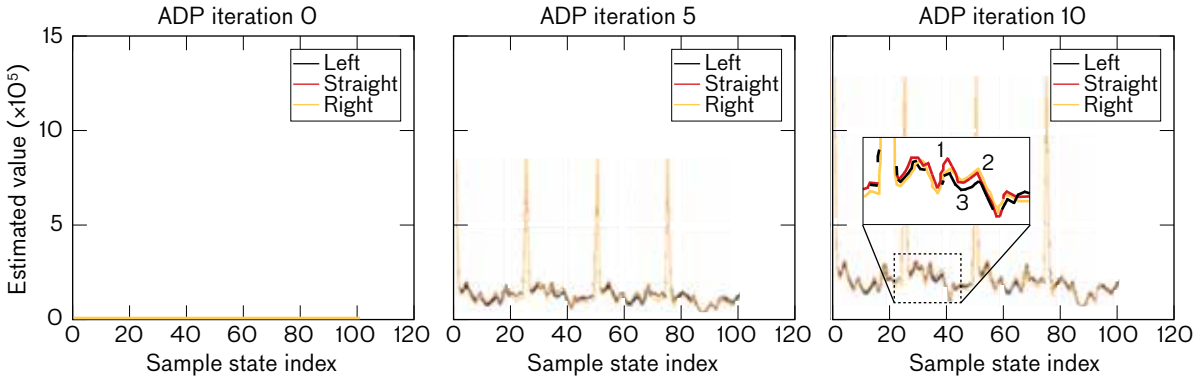


FIGURE 7. The approximate dynamic programming (ADP) iterates through the state-action value functions. The above graphs, taken at the zeroth, fifth, and tenth time points (of the ten iteration steps) in the policy learning process, depict the value of each action (left, straight, right) in a series of policy-driven trajectories through the state space. The objective of the repetitive reinforcement learning process is to generate a function that attributes (maps) a value to the three actions for every state in the state space. However, only a subset of the states can be sampled because of the very large state-space size. The middle and right graphs show four spikes for four trajectories. The values are higher at these mission start points because there is more information to collect and lower at the ends of missions because there is less information remaining. In the inset, position 1 indicates that the unmanned system should move straight (red), position 2 shows a slightly better choice of moving right (yellow) over straight, and position 3 clearly indicates that left (black) is not the best direction.

therefore allow the algorithm to rely more on offline knowledge in the middle of the mission (when long-term evaluation is sufficient) and more on simulation at the beginning and end of the mission (when short-term information is critical). Configuration parameters thus determine the appropriate algorithm among the possible alternatives. Selecting algorithms in this way enables us to design systems that are broadly applicable to many situations.

Demonstration of Results

At this stage, we are currently designing and implementing ARCAL's logistics executive and adjustable autonomy components. We have implemented the scout path-planning algorithm, which includes the path-planning problem, the offline value iteration procedure, and the online search for decision making. Overall, the scenarios task the scout to reduce variance within a 10-by-10 gridded area, but the areas of high variance differ across scenarios. To show how our scout finds efficient sensing paths, we focus on scenario A, which includes high variance across the entire map, thus simulating a setting of incomplete prior information for a region.

Scenario A runs with a mission length of 25 time steps. Within the scenario's context, we first illustrate the evolution and convergence of the approximate values function during value iteration. We use 100 samples to represent the state subset. The scout can be within any of 100 unique grid cells, with four possible orientations in each, and an infinite number of possible map beliefs. Our value function representation is thus extremely sparse relative to the actual state space. We display paths constructed during online execution to show how the scout chooses to survey areas with higher uncertainty that are within the time constraints. The tree search algorithm is limited to 20 node traversals of computation, but searches down to a depth of 7 nodes.

State Space Sampling

Figure 7 depicts the state-action value function evolving over 10 sets of value iteration. The x-axis represents different sampled states in our lookup table, and the y-axis shows the values associated with those states. When querying the value of a state, we are not actually querying but rather representing the estimation architecture that interpolates over the sample states in the table. However,

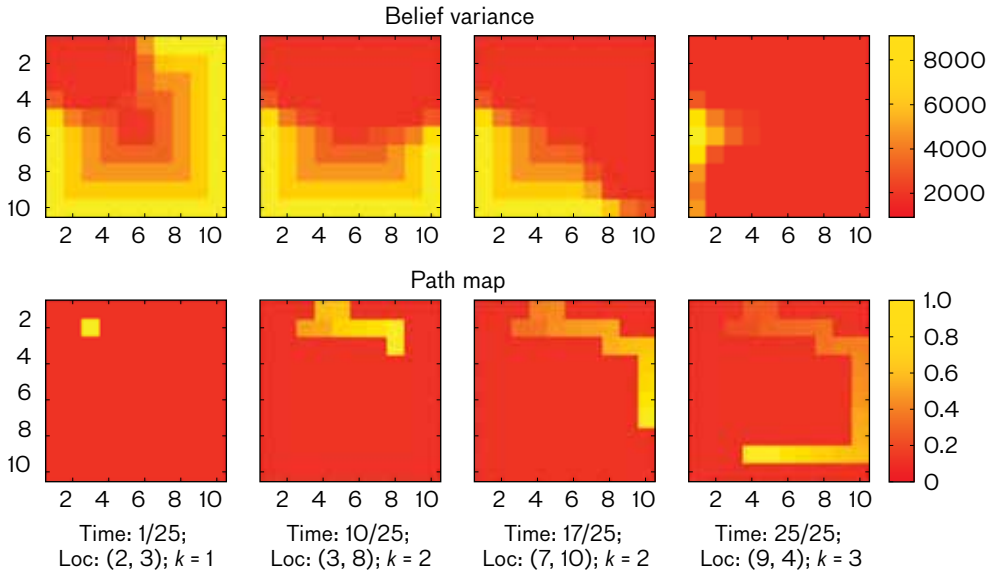


FIGURE 8. Row one and two respectively show the belief uncertainty and scout path for our scenario. The top row visualizes the belief uncertainty (variance) map at time steps 1, 10, 17, and 25; the yellow reflects a higher uncertainty, and red reflects a lower uncertainty. The bottom row visualizes the data-collection path. Loc indicates the current location of the scout.

to aid conceptual convenience and transparency, we will refer to the plots as the value-function plots. The sample states were constructed by initializing a simulated scenario four times and letting the scout fly a predetermined raster pattern that sweeps across the area for the 25-step length of the mission [6]. To avoid gathering the exact same data each time, we introduced stochasticity into the path and increased it with each subsequent pass.

Our method samples a state "trajectory," which is a path through the state space during a representative mission. Give a mission starting point, we employ a default, or initial, policy in order to choose initial actions. This "on-policy" approach tends to explore states that are likely to occur. Injecting some random decisions into the policy allows the system to explore actions (and the associated states) that are "outside of the envelope."

The result of this sampling is shown as a sequence of value function plots in Figure 7. At first, the values for each sample state are initialized with low random noise (not visible at the scale shown). In subsequent iterations, the values accrue at each step because each state "looks ahead" to the next best state and adds that state's value to its own reward (i.e., variance reduction) for taking the action leading into that state. The values gradually converge (i.e., the increase at each step gets smaller) since accrual is increasingly discounted over subsequent iterations. However, the most interesting parts of these plots are the four peaks that correspond to when the scout passes over high-variance areas and realizes large rewards. The reward subsequently decreases for these areas. The plots illustrate how the value function effectively encodes and exploits the structure of belief variance.

Figure 8 shows the algorithm's path construction at time steps 1, 10, 17, and 25. The upper plot in each frame shows belief variance with a color scale, while the lower plot shows the scout's path for the given time step. The diagram shows that the scout travels south into the area of highest uncertainty and traverses it until the end of the mission. Note how the scout systematically whittles away the belief variance in the top plots. We rescaled the colors so that areas with the highest remaining variance always appear yellow, and thus show how those locations guide and attract the scout. The scale changes significantly by the end of the mission, demonstrating the extent of variance reduction.

This example shows that our scout planning algorithm finds a path through an area such that it purposefully surveys the most uncertain features, thus generating the most valuable data for the logistics planner through a combination of the offline value iteration and online

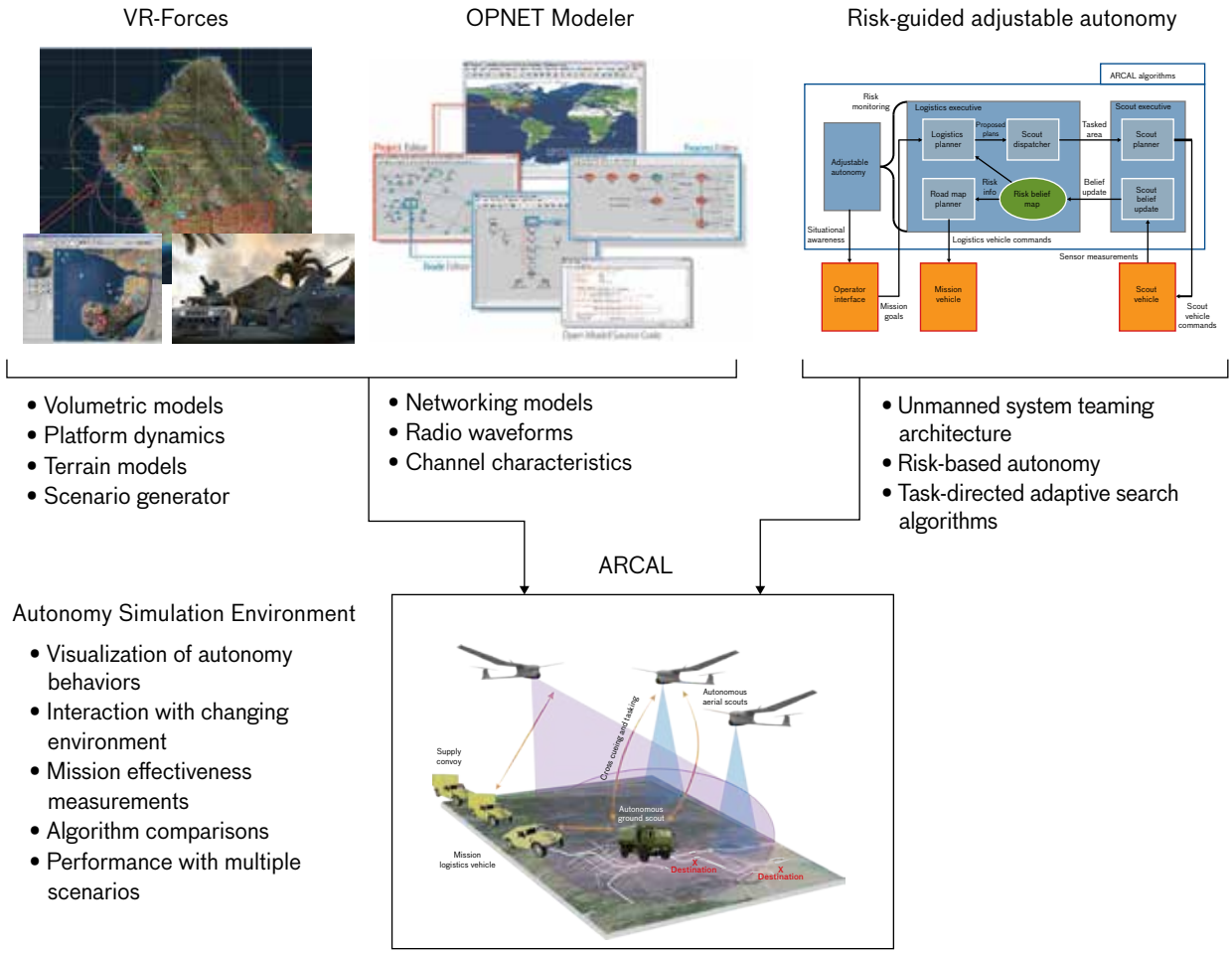


FIGURE 9. The autonomy simulation environment architecture leverages and integrates two powerful simulation tools: MÄK Technology's VR-Forces simulation framework for computer-generated forces and OPNET's Modeler simulation for high-fidelity communications. A well-defined interface is then specified to drive autonomous vehicles and to simulate communication and other interactions within a dynamic environment.

search procedures. The flexibility of our algorithm arises from the MDP framework, which easily adapts to any given scenario. An important detail of our approach is that, because we cannot exactly represent the value function, we acknowledge this problem by introducing stochasticity into our decision making. Thus, our nondeterministic solutions, while rarely optimal, are robust in the presence of this uncertainty. When integrated within the ARCAL system, the scouts can thus effectively contribute to real-time logistical planning.

Autonomy Simulation Environment

To further assess autonomous behaviors in changing environments, the ARCAL project is also developing an autonomy simulation environment (Figure 9). This environment includes a software infrastructure in which collaborative autonomy algorithms and system behaviors can be evaluated according to physical, environmental, and network effects. Accelerated 3D visualization of the simulation provides demonstrative context for the candidate algorithms. Metrics are being developed to assess proper autonomous decision-making behavior in multivehicle and hierarchical configurations.

Environment Framework and Components

MÄK Technology's VR-Forces (VRF) is a simulation framework for computer-generated forces (CGF), allowing for scenario generation and behavioral

modeling of ground, air, and sea entities (ground vehicles, aerial vehicles, autonomous ocean sensors) [7, 8]. Entities in VRF have 3D volumetric representations that interact with a specified terrain or the overall environment. Each entity has a parameter database that describes its physical and behavioral characteristics. The simulation engine uses these parameters as it interacts with the terrain and other entities. Scenario generation consists of generating a terrain and all simulation entities, which are assigned a plan consisting of small tasks. Similar to robotic systems, VRF uses a component architecture made up of sensors, controllers, and actuators. These components combine to form behavioral systems. The Laboratory's framework uses a customized application programming interface (API) to provide input to the autonomy algorithms. The object states of VRF are outputted over a distributed high-level architecture (HLA) for computer simulation systems. This HLA is similar to a Common Object Request Broker Architecture (CORBA) middleware that allows for federated applications using the same simulated objects. VRF includes a detailed graphical user interface front end that subscribes to objects over an HLA and renders them, along with the terrain, in an accelerated 3D environment.

It is important to model accurate network effects because of their relevance in algorithm design. Leveraging the fact that, in VRF communication, effects can be exported to an external server, we were able to integrate OPNET Modeler, a separate discrete-event simulator that excels in network simulations [9–11]. Communications between entities are sent over an HLA to an OPNET simulation whose timing is synchronized with VRF. By using VRF with customized component systems to provide behavioral modeling of unmanned aerial vehicles (UAV) within a 3D terrain, along with a front-end graphical visualization engine and OPNET Modeler to provide communication effects, we have a high-fidelity combination of software technologies and APIs with which we can test and evaluate candidate autonomy algorithms [12].

In order to support the ARCAL search and rescue scenario description, we needed the ability to discriminate terrain by using low-flying platforms. We assume that our UAVs are equipped with sensors that are able to discriminate between various terrain types (e.g., paved road, shallow to deep water, grass, boulders). In addition to identifying these terrain patches, we must determine whether a given area has been disturbed from its previous terrain type. We introduce a terrain "flag" that represents a drastic change to a cell's terrain. The sensors on the UAV detect terrain conditions and upload them to their overhead view of the area.

Sensor Modeling

We equip the model in VRF with a custom coarse-grained sensor that uses the terrain API to read what type of terrain exists in each cell as the UAV flies over it. For each terrain patch, two values are stored in a matrix: a "prior," which represents the risk of a given terrain (e.g., paved road is low risk, deep water is high risk), and a "variance," which represents the sensor's certainty of detection. A fine-grained sensor is integrated on a second UAV. It detects disrupted terrain and supplies updated variance values to further reduce uncertainty.

The UAV with the lower-resolution sensor flies in a simple raster pattern over an area of interest, forming the initial matrix of priors and variances. This low-resolution reconnaissance is a bootstrapping phase that can be cached. The scenarist can then apply calamity effects to the terrain by dropping a flag that represents a disruption of that terrain patch. Alternatively, a random application of calamity effects can be generated and overlaid onto the terrain. At this point, the priors matrix is split up into units for further refinement. For visualizing the sensed area, each sensor has a visual cone angle that captures the ground state as the UAV flies over the terrain. For the fine-grained sensor, terrain cells are initialized with values that represent a function of their priors and variances. The dynamic program uses the priors matrix to instruct UAVs with high-resolution sensors on where to go to improve the priors matrix, narrow variances, and otherwise detect the terrain's real state.

As the UAVs refine the priors matrix, communication occurs by sending data from the UAVs to the OPNET simulation, subjecting the data to the wireless effects of the configured channel and terrain, and to the communication effects of the configured radio. We are currently using a single candidate autonomy

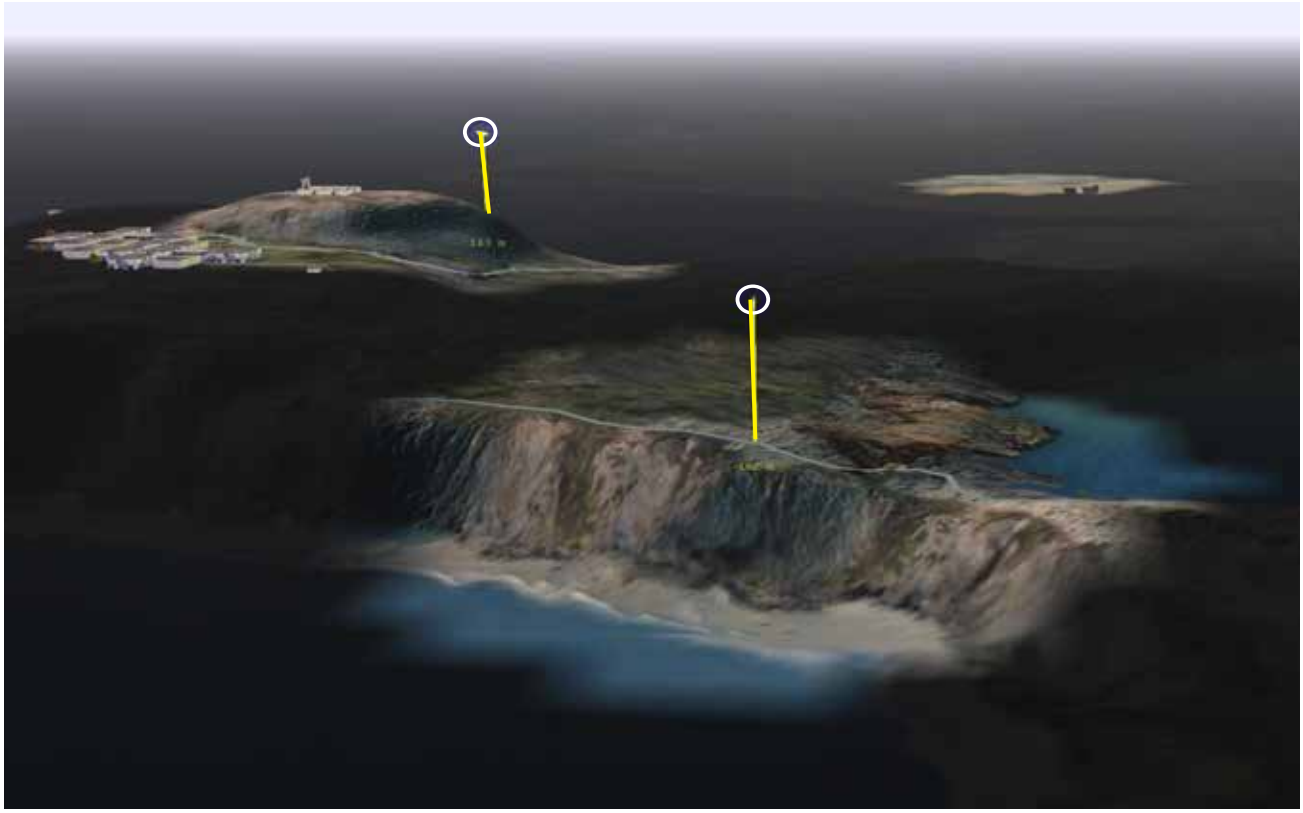


FIGURE 10. In this simulation, high-resolution sensors "paint" the terrain as the variances are reduced via data collection. The circles mark the locations of the UAVs and the yellow lines indicate the direction of the UAV sensors.

algorithm in the ARCAL simulation framework, but most of the framework development itself is agnostic with respect to the candidate algorithms.

As development of the Lincoln Laboratory ARCAL simulation framework continues, we are putting together an initial demonstration of some of the capabilities that will ultimately be integrated into the full infrastructure. Offline processing that uses the MIT scout path-planning algorithm inputs terrain characteristics into the ARCAL simulation environment to generate algorithm decisions. The demonstration will showcase the ability of the coarse-grained sensor to create a priors matrix for a given terrain and the fine-grained sensor's ability to further reduce variances by flying paths determined offline by the dynamic program of the algorithm developed by MIT's MERS Group.

The simulation assumes a single UAV rastering a given swath of terrain. The priors matrix is written to a file processed by a MATLAB script provided by MERS. The texture value enumerations that were recorded correspond to one of several supported surface types (e.g., asphalt, grass, deep lake, boulder). Next, the algorithm is run on the priors matrix to determine flight paths for the UAVs. These flight paths are then imported back into VRF. The high-resolution sensor will follow these paths with the "terrain-painting" customized graphical user interface plugin enabled (Figure 10). The terrain will be mapped in order to reduce variances. Ultimately, this processing will integrate with the simulation itself, and the aforementioned communications processing with OPNET will be used to exchange information among entities.

Looking Forward

In the future, unmanned platforms will gain higher-order decision-making intelligence, form teams, and perform collaborative tasks. The ARCAL project represents two complementary areas of research that increase operator confidence in future autonomous

system behaviors. The first incorporates concepts of risk-based adjustable autonomy with risk verification within system functions and task-directed adaptive search techniques. An operator can adjust the autonomy level, employ autonomy functions, or revert to fully manual control at any time. The second involves new methods to effectively test and evaluate collaborative autonomous team behaviors prior to field deployment via a high-fidelity, interactive simulation environment with 3D visualization.

Natural disaster relief scenarios were used to develop and validate the concepts and technologies used in this project. In these simulations, a team of unmanned aerial and ground scouts is dispatched to determine the most efficient path for a logistics vehicle. The autonomous algorithms, concepts, and technologies can be used with the autonomy simulation environment and for other situations.

This article also introduced the theoretical basis for adjustable autonomy used during control and supervision of a team of scouts performing a collaborative task. Task-directed search algorithms for UAV scout path planning improved knowledge of the risks to the mission. An algorithm test battery was developed and used to run tests on the scout path-planning algorithms. The ARCAL system's architecture incorporates autonomy algorithms that were tested with the natural disaster recovery scenario. Initial results show that the algorithm performs effectively.

The autonomy simulation environment integrates commercial state-of-the-art simulation and modeling products. It serves as a software infrastructure for evaluating candidate collaborative autonomy algorithms and system behaviors under very specific physical, environmental, and network conditions. An initial concept capabilities demonstration under development uses task-directed search algorithms for planning paths for unmanned scout vehicles to follow to update a terrain risk-belief map. This map is used in the ARCAL system architecture so that adjustable autonomy algorithms can plan efficient pathways.

This technology requires risk-based reasoning developed through formal models and algorithms. The architecture for such reasoning (in a context of path planning) must include characterization of temporal risk. Temporal coordination is an essential aspect of any mission that requires multiple activities to be executed in sequence or simultaneously, with future tasks depending on the completion of earlier tasks. The problem of assessing temporal risk is scheduling the activities such that they are probabilistically robust against scheduling uncertainty. This article formally described two algorithmic approaches to this problem, demonstrating one approach through experiments.

Today, both mission and sensing complexity are managed through increased automation that allows operators to abstract away from lower-level functions and focus on higher-level goals. The operator specifies goals at a certain level of abstraction and then relies on automation to achieve them. The result is a significant collaboration between humans and automation. Decisions traditionally made by humans are now automated and significantly improve the probability of a successful mission.

Sensing advances have increased mission performance in terms of faster execution and greater complexity. By providing relevant data with great immediacy, the sensors can immensely accelerate mission planning and execution. However, improved mission performance also requires greater sensing complexity in sensor coordination and analysis. Sensing an environmental feature may, for example, require a network of sensors operating in a coordinated manner. Trade-offs between coverage and resolution must be considered, and further trade-offs for resource scarcity are magnified if heterogeneous sensors are involved. To interpret the data from multiple sensors, data would need to be integrated and analyzed. Finally, rapid response requires real-time sensor coordination and data analysis.

Networked sensing systems are enabling unprecedented levels of mission performance through significant collaboration between human operators and advanced automation. In the last decade, advances in low-cost computation and networking have transformed single-instrumented sensors into networked systems of mobile elements. For example, aerial surveillance is progressing from being a mission conducted by a single, piloted aircraft to a continuous operation maintained by teams of smaller and less expensive UAVs. The result has been a dra-

matic increase in the observational capabilities and response times of sensing systems. The multitude and mobility of sensors can yield not only greater coverage but also greater depth and precision than achieved previously. Sensor networks can also offer redundancy and immediacy (i.e., repeated "looks" at and quick response to critical areas). Rather than being a single point of failure, sensing becomes a service whose performance improves or degrades gracefully with the number of sensing assets.

Acknowledgments

To carry out this work, researchers and engineers from MIT campus and MIT Lincoln Laboratory collaborated. Researchers (Lawrence Bush, Andrew Wang, and Prof. Brian Williams) from the MERS Group at MIT developed the theoretical basis for risk-based adjustable autonomy and task-directed adaptive sensing in order to autonomously coordinate multivehicle missions while keeping the operator in the loop. Jeffrey McLamb of Lincoln Laboratory developed the simulation environment to understand and evaluate autonomous collaborative teaming behaviors, with an emphasis on facilitating operator confidence for supervisory control. The authors would also like to acknowledge Michael Boulet, Bernadette Johnson, Jerry Jaeger, and Matthew Kercher for their encouragement of this research. ■

References

1. D.P. Bertsekas, *Dynamic Programming and Optimal Control,* vol. 1, ed. 3. Cambridge, Mass.: MIT Press, 2005.
2. D.P. Bertsekas, *Dynamic Programming and Optimal Control,* vol. 2, ed. 3. Cambridge, Mass.: MIT Press, 2007.
3. J.N. Tsitsiklis and B. Van Roy, "Feature-Based Methods for Large Scale Dynamic Programming," *Machine Learning,* vol. 22, no. 1, 1996, pp. 59–94.
4. D.P. Bertsekas and D. A. Castanon, "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heuristics,* vol. 5, no. 1, 1999, pp. 89–108.
5. G. Tesauro, "Temporal Difference Learning and TD-Gammon," *Communications of the ACM,* vol. 30, no. 3, 1995, pp. 58–68.
6. R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, Mass.: MIT Press, 1998.
7. "VR-Forces: Computer Generated Forces and Simulator Development," http://www.mak.com/products/simulate/vr-forces.html.
8. H. Gao, Z. Li, and X. Zhao, "The User-defined and Function-strengthened for CGF of VR-Forces," *Computer Simulation*, vol. 6, 2007, pp. 212–215.
9. "OPNET," Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/OPNET#Corporate_history.
10. G.F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. Reed, "OPNET Modeler and Ns-2: Comparing the Accuracy of Network Simulators for Packet-Level Analysis Using a Network Testbed," *WSEAS Transactions on Computers,* vol. 2, no. 3, 2003, pp. 700–707.
11. B. Meenakshi, R. Rajput, and G. Gupta, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," The Internet Society, 1999.
12. I.S. Hammoodi, "A Comprehensive Performance Study of OPNET Modeler for ZigBee Wireless Sensor Networks," *Proceedings of the 3rd International Conference on Next Generation Mobile Applications, Services and Technologies,* 2009, pp. 357–362.

Appendix

Temporal Risk Assessment

In many transport scenarios, it is critical not only to fulfill the goals but also to complete them "on time." The risk assessment tools for autonomous robot control discussed in the main article must be extended to assess the timeliness of proposed paths. Robots need to be sent to locations where information on route accessibility is missing and to pathways where the traverse times are uncertain.

Scenario

A transport convoy responding to a natural disaster consists of teams of ground vehicles with trained responders that must traverse unknown terrain to reach people in need. The mission goals are to

- Deliver provisions, administer medical care, transport victims to hospitals or shelters, repair infrastructure, and set up field stations;
- Transport responders throughout the affected region, performing each task according to its priority;
- Task robotic aerial scouts to study terrain.

The role of automation is to coordinate these goals by planning navigation for ground vehicles while human responders focus on their tasks.

Figure A1 illustrates a disaster relief scenario. A team waiting at a depot has been tasked with picking up a patient at location A, transporting him to the hospital, and unloading supplies at the shelter at B along the way. The navigation planner generates possible paths along roads from the depot to A, A to B, and B to the hospital. The risk threshold, which pertains to the patient transport task, may not exceed, for example, 5%. The total risk specifically includes the risk of successful traversal across damaged roads on a one-hour deadline. The depot, A, and the hospital are on one side of a river while B is on the other side. The only feasible paths to and from B (given the deadline) traverse bridges that may have been critically damaged. A longer path that avoids these bridges leads from the depot to B and incurs the least traversal risk. The final stage from B to the hospital may require aerial scouts to survey three candidate bridges.

While the transport team is picking up the patient at A, scouts relay the information that only one of the bridges is traversable and, furthermore, the roads from this bridge to B pose a 10% risk to the patient. Operators can either accept the additional risk or take the patient to the hospital first and then visit the shelter at B, but the longer path and nightfall will make it harder

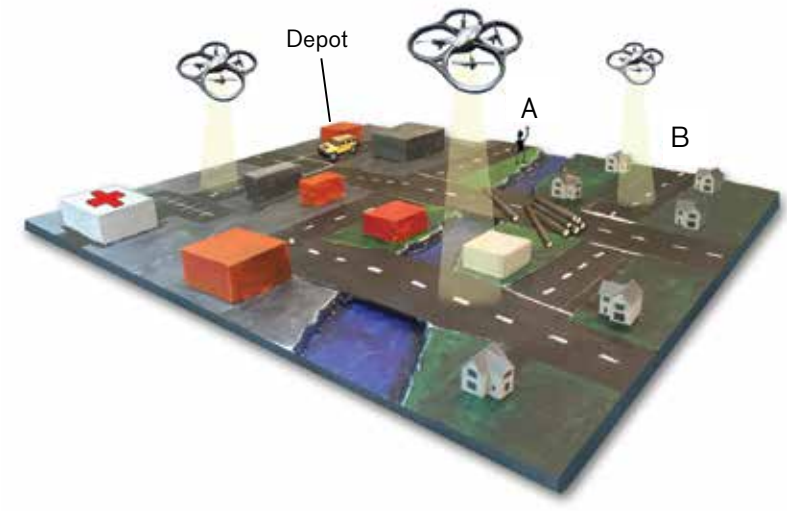


FIGURE A1. A transport scenario may include multiple tasks with associated risks. For example, a patient at location A needs to reach a hospital within one hour, while people at location B need supplies and access to shelter before nightfall.

to drive safely through damaged roads.

The architecture can also incorporate unforeseen developments (both positive and negative) that affect the risk distribution. A person on the response team, for example, may be able to perform some sort of triage on the patient en route, giving the team a longer window to reach the hospital. With the deadline extension, the automation is able to find a plan that addresses the task at B and results in 6% risk to the overall mission completion.

Scheduling

In assessing the mission risk, the automation must consider (1) the uncertainty of whether one can traverse roads successfully and in a timely manner relative to other mission segments, and (2) the possible repercussions if roads are not passable within the risk tolerances. For instance, two of the bridges in the scenario are untraversable, and the third has traffic or requires more care in traversing and therefore more time. The simulation considers an off-road alternative in the absence of traversable roads. This iteration of the simulation would send scouts to assess the off-road terrain in a timely manner, so that the risk map is current by the time the patient at A is picked up. Multiple vehicles were not considered in this scenario, but in a plan that involves a rendezvous point, vehicles would ideally converge simultaneously to conserve resources, streamline other tasks, and minimize risk.

The main article describes how to plan and model the success probability of a planned path through a landscape affected by a natural disaster or crisis. The algorithm calculates the risk-belief distribution of a path according to risks encountered. Once risks for each potential path are known and the optimal path has been determined, responders still face the challenge of minimizing and prioritizing response times for different crisis-related tasks. Response time and the coordination of tasks according to temporal constraints are essential aspects of the planning architecture. A mission may call for multiple activities to be executed in sequence or simultaneously. Temporal risk management requires the coordination of tasks in a way that is probabilistically robust against temporal uncertainty.

Previous research has shown that an iterative risk reallocation algorithm capitalizes on the structure of the desired temporal coordination (e.g., identifying scheduling conflicts and trying to solve them) to better serve the operator in real-time temporal risk management. Risk reallocation requires an algorithm that makes local, iterative adjustments but has global guarantees of identifying a schedule that meets the risk criteria. As with other forms of risk, the adaptive sampling algorithm can reduce temporal risk by evaluating it within the mission model and evaluating risk estimates according to overall mission risk thresholds. The planner determines a task ordering (i.e., schedule) that satisfies the mission's temporal goals with a certain probability.

Risk Allocation

Calculating the exact risk becomes increasingly difficult with more complex mission structures. Finding the optimal schedule is not typically a tractable approach. The most practical approach finds a feasible schedule that obeys some minimum failure rate. First, we formally define the scheduling problem through a specification of chance-constrained temporal goals and temporal uncertainty. Then, we restrict the solution space to scheduling strategies that are strongly controllable (i.e., a complete schedule that is robust against future uncertainty).

After defining the problem, we reformulate it as a temporal problem with uncertainty. This reformulation decouples chance and temporal constraints, and maps the strong controllability condition into strong controllability for a previously studied problem. The principle behind our reformulation is to allocate temporal risk to each activity's duration. Each allocation reduces an activity's probabilistic model of temporal uncertainty into an interval bound. Thus, satisfaction of the chance constraint depends wholly on the risk allocation, while temporal constraints are evaluated solely on the structure of the interval bounds for duration. We choose an interval-bounded reformulation because it transforms the structure of temporal uncertainty into one that is addressable by efficient, controllability-checking algorithms. A strong controllability version of these algorithms is described and exploited by our algorithm. The problem can then be reformulated into a solvable form.

Given a simple temporal network, find a schedule that satisfies its chance constraint:

- Drive to A. Pick up patient.
- Drive to B. Deliver shelter supplies.
- Drive to hospital. Unload patient.

We assume that the transit components of each task

are uncontrollable because of road conditions, while the rest of the tasks are controllable but have specific temporal requirements. Operator-imposed temporal constraints begin upon arrival at an accident scene. In the ongoing operational example, the constraint might be that once the patient has been moved, the transport team must reach an emergency room within 60 minutes or the patient dies, immediately nullifying the value of the operation.

Figure A2 illustrates the encoding of this scenario as a probabilistic simple temporal network (pSTN). Uncontrollable events are squares and represent arrival times. Uncontrollable durations are represented by dotted arrows; controllable durations and temporal constraints are represented by solid arrows. Each arrow is labeled with its respective constraint (in minutes). Each controllable duration and simple temporal constraint has a lower and upper temporal bound (in the figures, this is unknown) as follows:

- Start from location D' (depot).
- Picking up and loading the patient at A takes at least 5 minutes.
- Delivering shelter supplies at B takes at least 10 minutes.
- Unloading the patient at H (hospital) takes at least 1 minute.
- Finally, the patient must reach the hospital within 60 minutes of the pickup time.

Each uncontrollable duration has a continuous probability distribution built from the awareness of road conditions. Constraint satisfaction problems can be represented as graphs with variables as vertices and constraints as edges. Graphs elucidate the dependency structure among constraints based on the constraints' shared variables.

Risk allocation distributes the chance constraint's specified failure probability over the various sources of uncertainty. In our case, these sources are the uncontrollable durations. Assuming an interval bound as a duration's domain effectively assigns risk to that duration (i.e., the probability that the realized duration will fall outside the interval). The combination of each duration's assumed interval then becomes the macro interval under consideration.

This type of risk allocation enforces structure, which enables evaluation of both variables and constraints. Specifically, the structure becomes a rectilinear parallelepiped, with the axes aligned within the uncontrollable outcome space. The assumption that all durations are independent of each other means that conditions may be evaluated as the product of probabilities for each duration and strong controllability can be easily verified. Placing interval bounds on each uncontrollable duration reformulates the pSTN as a simple temporal network with uncertainty (STNU). In contrast to probability distributions with infinite domains, these hard-bounded assumptions of temporal uncertainty simplify the controllability-checking criteria. Thanks to previous research, efficient algorithms exist to check both the strong and dynamic forms of STNU controllability. Risk allocation effectively restricts the solution to a series of small components (i.e., durations) that are easy to adjust to satisfy the conditions of the scenario.

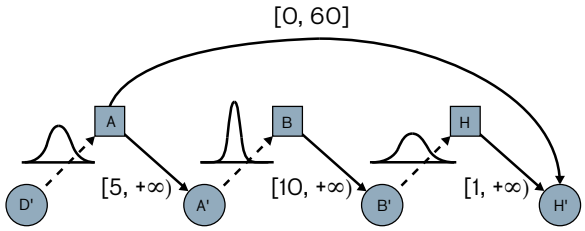


FIGURE A2. The probabilistic simple temporal network (pSTN) encodes the disaster relief scenario and the timing risks in seconds. The process comprises a set of uncontrollable events (squares) between locations (circles), uncontrollable durations (dotted arrows), and controllable durations (solid arrows). In a pSTN, each uncontrollable duration is described by a probablistic distribution. The brackets indicate the fixed time limit—no longer than 60 minutes total time—and the known minimum time for a given path. The parentheses indicate that the maximum times are currently unknown.

Figure A3 shows the disaster relief scenario in reformulated form. Note that the temporal structure remains virtually unchanged. Events, nodes, and durations remain in their original locations. However, each uncontrollable duration now has a lower-bound variable and an upperbound variable. The highlighted probability mass is the likelihood that the duration will land in between these bounds or between the inverse of the risk assigned to that duration.

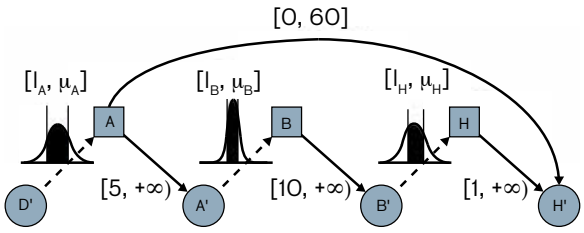


FIGURE A3. The original pSTN scenario is now reformulated as a simple temporal network with uncertainty (STNU). An STNU problem representation bounds each individual uncontrollable duration (with probable time l and uncertainty μ), reducing the effect of the broad uncertainty on route selection.

Controllability can then be checked by using a specific grounded disaster relief scenario. Checks can identify events that become uncontrollable under certain constraints, thus removing those events from consideration within scenarios. After cycling through certain constraints, the overall STNU can itself be evaluated for controllability.

About the Authors

Lawrence A.M. Bush is a senior perception software engineer at Autonomous Solutions Inc., where he is responsible for providing technical direction on the development of the design and algorithms for an autonomous ground vehicle perception system. He leads the research and development of algorithms for indoor and outdoor GPS-denied localization (e.g., cleaning and orchard applications) and develops algorithms for object and void detection and for terrain characterization for mining, drilling, and military convoy systems. He designs fail-safe system verification processes for autonomous road vehicles. Previously, at Lincoln Laboratory, he led machine learning algorithm development and autonomous system design work. Specifically, he conducted research in the areas of sensor data analytics for decision support; battle management, command, and control; human-machine interaction; and vehicle autonomy. He developed and led a human-in-the-loop experiment investigating the utility of decision support algorithms, integrating radar data and unmanned aerial vehicle–based optical imagery. He holds a doctorate in autonomous systems from MIT. His doctoral thesis research, performed in the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) in collaboration with the Monterey Bay Aquarium Research Institute and the NASA Ames Research Center, focused on autonomy for sensing missions. He has published papers on underwater vehicle control and autonomous air combat. He also advises the Utah State University Autonomous Vehicle Competition team.

Andrew Wang is a doctoral candidate at MIT, studying automated reasoning. His work focuses on modeling and assessing the risk of plans to improve the safety of robotic task execution. He is interested in scheduling algorithms that anticipate upcoming failures during execution and respond with dynamic updates to preserve safety. His work has been applied to designing space probe missions, underwater robotic surveys, and urban transportation logistics. He earned bachelor's degrees in aerospace engineering and in electrical engineering and computer science (EECS), and a master's degree in EECS, all from MIT. He has received fellowship offers from the Department of Defense and NASA.