

# Artificial Intelligence Approaches To UCAV Autonomy

Amir Husain\*, Bruce Porter†

January 26, 2017

amir@sparkcognition.com, porter@cs.utexas.edu

## 1 Abstract

This paper covers a number of approaches that leverage Artificial Intelligence algorithms and techniques to aid Unmanned Combat Aerial Vehicle (UCAV) autonomy. An analysis of current approaches to autonomous control is provided followed by an exploration of how these techniques can be extended and enriched with AI techniques including Artificial Neural Networks (ANN), Ensembling and Reinforcement Learning (RL) to evolve control strategies for UCAVs.

## 2 Introduction

Current UAVs have limited autonomous capabilities that mainly comprise GPS waypoint following, and a few control functions such as maintenance of stability in the face of environmental factors such as wind. More recently some autonomous capabilities such as the ability for a fixed wing UCAV to land on the deck of a carrier have also been demonstrated [1]. These capabilities represent just the tip of the spear in terms of what is possible and, given both the commercial and military applications and interest, what will undoubtedly be developed in the near future. In particular, flexibility in responses that can mimic the unpredictability of human responses is one way in which autonomous systems of the future will differentiate themselves from rules-based control systems. Human-style unpredictability in action selection opens the door to finding solutions that may not have been imagined at the time the system was programmed. Additionally, this type of unpredictability in combat systems can create difficulties for adversary systems designed to act as a counter.

---

\*SparkCognition Inc.

†Department of Computer Science, University of Texas at Austin.

The capability to compute sequences of actions that do not correspond to any pre-programmed input - in other words, the ability to evolve new responses - will be another area of future differentiation. There are many other such enhancements that will be enabled via autonomous systems powered by Artificial Intelligence. In the following sections we will outline some of the advanced capabilities that can be engineered, and design and engineering approaches for these capabilities.

### 3 Existing Control Systems

Some degree of autonomy in flight control has existed for over a hundred years, with autopilot inventor, Lawrence Sperry's demonstration in 1913 [5] of a control system that tied the heading and attitude indicators to a control system that hydraulically operated elevators and rudders. A fully autonomous Atlantic crossing was achieved as early as 1947 in a USAF C-54 aircraft [6]. However, much of the early work in automating control systems were mechanical implementations of rule-based systems drawing upon cybernetics and control theory. They demonstrated that with such techniques it was possible to automate a basic mission, including takeoff and landing.

Since the 1947 demonstration, considerable effort has been invested in developing autonomous flight capabilities for commercial and military aircraft. Modern flight control or autopilot systems that govern landings are segmented in five categories from CAT-I to CAT-IIIc [11], with capabilities varying based on forward visibility and decision height. Many of these systems use rule-based, or fuzzy-rule based control, incorporating sensor-fusion techniques such as Kalman filters [7]. They are capable of following a planned route and adjusting for environmental factors such as cross-winds, turbulence and so on.

The increased popularity of commercial drones, and the heightened utilization of military drone aircraft has, in parallel, created a new class of autonomous capabilities. From Open Source initiatives such as the Ardupilot[8] flight control software for low-cost drones, to higher levels of autonomy in military drones. Software such as the Ardupilot, for example, uses a combination of GPS positioning, additional sensors to gauge velocity and position, combined with basic flight control rules to autonomously navigate to a sequence of waypoints. Many of these map-input based waypoint following capabilities are also implemented in military surveillance and combat drones.

Another area of control innovation comes from swarm theory and related control algorithms. At the simplest level, these algorithms seek inspiration from the behavior of biological systems such as ant colonies or flocks of birds. They are collaboration algorithms that enable each individual system in the swarm to compute its future actions based on its own measurements, but also those of its neighbors. While basic swarm algorithms [10] are effective in providing coverage over an area, and automatically repositioning all nodes when one is lost to maintain coverage, they do not provide much guidance on how to divide mission responsibilities and burdens, and to effectively delegate them to

individual nodes. The concept of a “swarm” as found in biology will have to evolve into something entirely different - perhaps somewhat similar to a pack hunt - but even that analogy would only be marginal - in order for it to be an effective and useful system particularly in a military context. Some of the reasons why we propose this conclusion regarding the inadequacy of existing swarm algorithms is that most biologically inspired algorithms, such as Particle Swarm Optimization (PSO) [12] or Artificial Bee Colony Algorithm (ABC) [13], are search or optimization techniques that do not account for the role of an individual particle (or node) in the swarm. For example, PSO proposes the same meta-heuristic for computing positional updates for all points and does not incorporate a differential update mechanism based on the role of a particle. In a subsequent publication, we intend to propose a “Pack Hunt Optimization” (PHO) algorithm that we believe addresses the shortcomings of the existing swarm algorithms we have cited, and holds relevance to UCAV control applications.

The state of current control systems can be summed up as follows:

- Effective at basic navigation and path following
- Many existing techniques to fuse sensor data for accurate position identification
- Able to automatically take off and land if runways are properly instrumented
- Actions beyond flight control (such as weapons engagement) are presently manual
- Missions are pre-defined
- Swarm algorithms can provide additional value for relative positioning of multiple assets and distributed sensing

## 4 Advanced Autonomous Capabilities

The purpose of this section is to outline a few areas of potential advancement that can be expected of autonomous systems of the future. This list is neither exhaustive nor complete with regards to the author’s current conception of all such advanced capabilities. It is a subset of possible functions that is listed to illuminate the broad contours of what is possible in terms of applications of Artificial Intelligence to UCAV autonomy. Some features include:

1. Knowledge & Assessment Updates
  - (a) Identification of potential threats outside pre-programmed mission briefs
  - (b) Autonomous exploration and assessment of identified targets that autonomous control deems to be high priority

- (c) Enhancement and update to intelligence supplied as part of the mission brief and plan, based on actual observation
2. Autonomous Navigation and Swarm Coordination
    - (a) Ability to adjust to environmental conditions that cause system or any linked swarm systems to deviate from mission plan expectations
    - (b) Ability to adjust to loss of a Swarm asset, not just in terms of re-positioning, but including potential re-tasking (i.e. assumption of a new role on the part of an individual asset)
  3. Autonomous Evasion
    - (a) Automated update to mission plan based on sensor detection of probable manned aerial intercept
    - (b) Automated update to mission plan based on detection of unexpected sensor presence
    - (c) Autonomous evasion in the event of a RWR (Radar Warning Receiver) activation or MAW (Missile Approach Warning) system activation
  4. Autonomous Targeting
    - (a) Autonomous addition to target lists based on computer vision or alternate sensor based identification of threats to mission (including surface to air threats)
    - (b) Autonomous addition to target lists in the event that primary targets have already been neutralized
    - (c) Autonomous deletion of a target from target lists in the event it has been already neutralized, is found to violate a “hard” policy constraint or is low priority and its neutralization harms the overall achievement or success of the mission

## 5 The Need for a New Approach

In the preceding sections we explored the current state of autonomous systems and the rules-based approach that is often employed to develop these systems. Further, we also considered a number of advanced capabilities that would be desirable in future autonomous control systems. A fundamental challenge in developing these future capabilities is that the range of scenarios an autonomous system would have to contend with in order to effectively execute the required maneuvers are enormous. Tackling such a large range of possibilities with a rules-based system will be impractical not only because of the combinatorial explosion of possibilities that would require individual rules, but also because

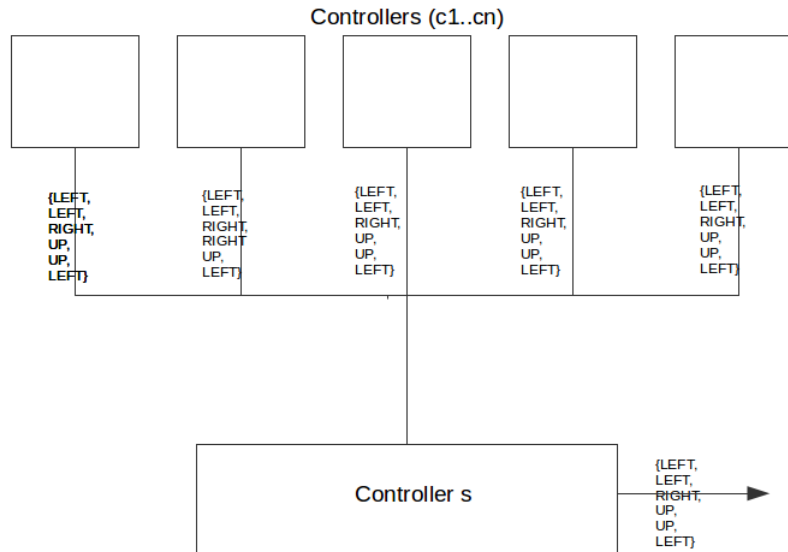
human designers of such a system may simply not be able to conceive every imaginable scenario the autonomous system could find itself in.

Another challenge is that rules-based systems are hard coded to measure certain criteria, or sensor values, and then act based on this pre-specified criteria. This hard coding means that each rule is tied to a specific set of sensors. If additional sensors are added to a system, or existing sensors are upgraded, a large number of rules would have to be re-written, creating an obvious cost and effort burden.

What we have described above is far from an exhaustive list of limitations in current autonomous systems, but we believe they are sufficient to motivate the need for a new architecture for autonomy. A future system that moves beyond rules-based systems, incorporates learning capabilities so that actions can be learned rather than hard coded, and can adapt to new information from new or better sensors, will represent a substantial advance. In the sections that follow, we define the contours of just such a system.

## 6 An Architecture for Advanced Autonomy

The fundamental architecture we propose in this paper is based on multiple independent control systems connected to an action optimizer neural network. Each of the multiple independent control systems can be neural networks or non-ANN rule based control systems that output a suggested vector of actions or control activations. The action optimizer ANN gates and weighs the inputs supplied by each independent control system.



Let  $c_k$  be an independent control system, and  $s$  be an action optimizer neural network to which  $c_{1..n}$  control networks are connected. Additionally, let the set  $E$  contain a collection  $e_{1..m}$  of environmental inputs that are supplied to  $s$ . Then, we denote the specific configuration of all environmental inputs at time  $t$  by  $E^t$  and the output of  $s$  under these environmental inputs and based on the inputs of all independent control networks, as follows:

$$s(E^t, C^t) = A^t$$

The goal of our system is to optimize the selection of action sequences  $A^{t..t+k}$  such that this sequences maximizes the performance of the system being controlled.

It is important to understand what we mean by, “performance” here. We define performance as a variable that is the output of a utility function  $U$  such that this output is high when the weighted achievement of all mission parameters is large, and low when the weighted achievement of mission parameters is small. In other words, we are attempting to locally maximize  $U$  at least locally:

$$\frac{dU}{dx} = 0$$

and:

$$\frac{d^2U}{dx^2} < 0$$

The question obviously arises, how do we build the function  $s$ ? Conventionally, control functions have been built in various ways, for example as fuzzy rule based systems [3]. However, we propose to implement the control function  $s$  as an Artificial Neural Network (ANN). As the application at hand will benefit from some knowledge of past actions, we specifically propose to implement the network as a Recurrent Neural Network (RNN).

## 7 Evolving Mission Specific Controllers

The actual training and evolution of the RNN represented by  $s$  is not the subject of this paper and will be documented in a subsequent publication. In summary, this can be done in a manner that combines real world and simulator environments. However, in a more detailed future exploration we intend to cover questions such as whether individual control networks,  $c_{1..n}$ , can be trained independently and how a training set that reflects key the wide range of scenarios theUCAV might experience would be compiled. For the purpose of the present discussion, our basic approach is to use Reinforcement Learning (RL) techniques [4] to train the RNN in a simulated environment until a basic level of competence has been achieved, and to then allow the evolved network to control a real craft. Collected data from the actual flight is reconciled with the simulated

environment and the process is repeated until an acceptable level of capability is demonstrated by  $s$ . This reconciliation would benefit from applications of Transfer Learning [14].

One of the benefits of this approach is that the simulated environment can introduce environmental constraints that  $s$  must respond to appropriately. For example, these can be navigation constraints such as avoiding certain pre-identified objects on a map. Work has already been done to use search algorithms such as A\* to find viable paths around objects to be avoided [2] and this type of constraint can be implemented by one of the independent control networks ( $c_k$ , as presented in the previous section). Other examples of existing work that could be leveraged in the form of an independent control network include collaborative mapping algorithms for multiple autonomous vehicles [15]. Of course, other constraints and optimizations would be represented by other ensembled control networks, forcing  $s$  to weight them and choose from them carefully, in a way that maximizes  $U$ .

Thus, the controller can be evolved to optimize operation in different types of environments, and under different constraints. It may then become possible to simply “upload” the optimal controller for a particular environment, or a particular mission type, into the same craft and achieve mission-specific optimal performance.

## 8 Semantic Interpretation of Sensor Data

Sensor data in autonomous systems does not have to remain limited to environmental measurements or flight sensor readings. It can include a variety of image feeds from forward, rear or down-facing cameras. Additionally, radar data and Forward Looking Infra Red (FLIR) sensor data is also a possibility. In order to utilize all this diverse data to make decisions and even deviate in small but important ways from the original mission plans, all of this data has to be interpreted and semantically modeled. In other words, its meaning and relevance to the mission and its role in governing future action has to be established.

For the purpose of understanding how such data can be interpreted and what its impact on decisions can be, we classify sensors and data sources into the following categories:

1. Internal Sensors
  - (a) System Health (e.g. Engine Vibration, Various Temperature and internal system Pressure)
  - (b) System Performance (e.g. Velocity, Stress)
2. External Sensors
  - (a) Navigational Aides (e.g. Level, Wind speed, Inertial navigation gyroscopic sensors)

- (b) Environmental Mapping (e.g. Camera, Radar, LIDAR, FLIR, RWR, MAWS)

In an example table below, we show the types of impact that information received from these sensors can potentially have on mission plans and vehicle navigation.

Actions	Health	Perf.	Nav.	Envir. Mapping
Terminate Mission	X	X		X
Update Mission Achievement				X
Add New Target				X
De-prioritize Target	X			X
Change Course	X		X	X
Add Obstacle (Constrain Path)				X
Engage Weapon System				X
Evasive Maneuvers				X
Engage Countermeasures				X

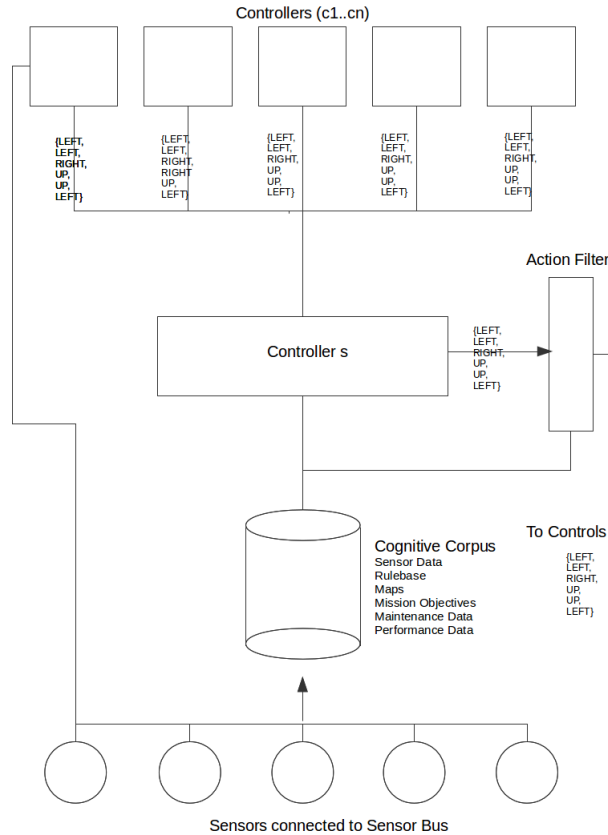
In order to support the types of advanced autonomy outlined in Section 4 of this paper, many of the actions highlighted in the table above will likely need to be combined based on sensor input to form a chain of actions that update the internal state and maps used by the autonomous asset. Sensor data may be an input required by any controller  $c_k$  or by the controller  $s$ . Thus, a sensor bus connects all sensors to all controllers.

For many sensor types, instead of the sensor providing a raw output, we transform the output to reflect semantic constructs. For example, instead of a raw radar signal input, we may transform the signal into a data structure that reflects the position, speed, heading, type and classification of each detected object. This transformation of raw sensor data into semantic outputs that use a common data representation for each class of sensor enables replaceability of underlying components so that the same controllers can work effectively even when sensors are replaced or upgraded.

The semantic output of individual sensor systems can be used by controllers, and is also stored in a Cognitive Corpus, which is a database that can store mission information, current status, maps, objectives, past performance data and not-to-violate parameters for action that are used to gate the final output of the controller  $s$ .



## 9 Knowledge Representation For Advanced Autonomy

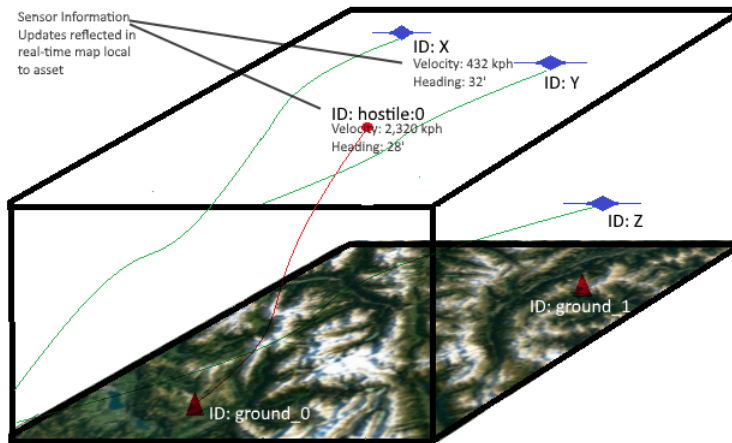


As the more complete diagram of the proposed Autonomy Architecture illustrates, the controller  $s$  receives input from a set of controllers  $c_{1..n}$  and is also connected to the sensor bus and the Cognitive Corpus. A state map stored in the Cognitive Corpus reflects the full environmental picture available to the autonomous asset. For example, it includes an estimate of the asset's own position, the positions of allied assets, the positions of enemy assets, marked mission targets, paths indicating preferred trajectories at the time of mission planning, territory and locations over which to avoid flight and other pertinent data that can assist with route planning, objective fulfillment and obstacle avoidance.

This state map forms another important input to the controller  $s$  as it chooses the most optimal sequence of actions. The image below shows a visual representation of what the state map might track. Here, it shows the location of multiple allied assets, for example systems that might be part of a swarm with the UCAV that is maintaining this map. There is also a hostile entity identified with additional information regarding its speed and heading. Locations on the

ground indicate sites to be avoided. Sensor information carried in the set (or vector)  $E$  result in updates to the state of each object in this map. Note that the state map is maintained by each autonomous asset and while the underlying information used to update it may be shared with, or received from other systems, each autonomous asset acts based on its own internal representation, or copy, of the state map.

While the details of an implementation are beyond the scope of this paper, we propose that the information exchange between autonomous systems occur using a blockchain protocol [9]. Benefits of this approach include the fact that in the event communication is interrupted and updates are missed, information can be reconstructed with guarantees regarding accuracy and order. Further, the use of a blockchain store ensures that a single or few malicious participants cannot impact the veracity of the information contained therein.



While the figure shows a graphic representation of the map, it is possible to represent such a map as a vector or matrix. By so doing, it can readily be supplied to the controller  $s$  as an input.

## 10 Conclusion

Sophisticated autonomy requires control over a wider range of action than rule based systems can support. The subtle changes in flight patterns, identification of new threats, self-directed changes in mission profile and target selection all require autonomous assets to go beyond pre-ordained instructions. Machine Learning and AI techniques offer a viable way for autonomous systems to learn and evolve behaviors that go beyond their programming. Semantic information passing from sensors, via a sensor bus, to a collection of decision making controllers makes provides for plug and play replacements of individual controllers. An artificial neural network such as an RNN can ensemble and combine inputs

from multiple controllers to create a single, coherent control signal. In taking this approach, while some of the individual controllers may be rules-based, the RNN really evolves into the autonomous intelligence that can consider a variety of concerns and factors via control system inputs, and decide on the most optimum action. We propose delinking control networks from the ensembler RNN so that individual control RNNs may be evolved and trained to execute differing mission profiles optimally, and these “personalities” may be easily uploaded into the autonomous asset with no hardware changes necessary. One of the challenges in taking this advanced approach may be the inability to guarantee what exactly a learning, evolving autonomous system might do. The action filter architecture proposed in this paper, which provides a hard “not to exceed” boundary to range of action, delivers an out-of-band method to audit and edit autonomous behavior, while still keeping it within parameters of acceptability.

## References

- [1] Vinson (2013). X-47B Makes First Arrested Landing at Sea, Navy.mil.
- [2] Casteli et. al. (2016). Autonomous navigation for low-altitude UAVs in urban areas, arxiv.org.
- [3] Ansari & Alam (2011). Hybrid Genetic Algorithm fuzzy rule based guidance and control for launch vehicle, Intelligent Systems Design and Applications (ISDA) Conference.
- [4] Wang et. al. (2016). Learning to Reinforcement Learn, arxiv.org.
- [5] HistoryNet (2006). Lawrence Sperry: Autopilot Inventor and Aviation Innovator, HistoryNet.
- [6] Chicago Tribune (1947). Reveal 'Robot' C-54 Zig-Zagged Way To England, Chicago Tribune Sept. 24, 1947.
- [7] Welch & Bishop (2001). An Introduction to the Kalman Filter, SIGGRAPH 2001.
- [8] Bin & Justice (2009). The Design of an Unmanned Aerial Vehicle Based on the ArduPilot, Indian Journal of Science & Technology April 2009.
- [9] Ferrer (2016). The blockchain: a new framework for robotic swarm systems, arxiv.org.
- [10] Hexmoor et. al. (2005). Swarm Control in Unmanned Aerial Vehicles, ICAI 2005.
- [11] Federal Aviation Administration. Flight Operation Branch, Category I/II/III ILS information, www.faa.gov.

- [12] Kennedy & Eberhart (1995). Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948.
- [13] Karaboga (2005). An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department 2005.
- [14] Pan & Yang (2009). A Survey on Transfer Learning, IEEE Transactions on Knowledge and Data Engineering.
- [15] Luotsinen (2004). Autonomous Environmental Mapping In Multiagent UAV Systems, Masters Thesis, University of Central Florida.