# Man-Machine Interoperation in Training for Offensive Counter Air Missions

**Patrick L. Craven, Kevin B. Oden, Kevin J. Landers**
**Lockheed Martin Corporation**
**Orlando, Florida**

**Ankit J. Shah and Julie A. Shah**
**Massachusetts Institute of Technology**
**Cambridge, Massachusetts**

## ABSTRACT

The application of an artificial intelligence (AI) agent developed via Machine Learning (ML) was investigated for the purpose of automatically interpreting the execution of a simulated Offensive Counter-Air (OCA) missions flown by experienced fighter pilots. The agent demonstrated the ability to interpret the behaviors of human pilots flying missions in a synthetic task environment (STE) using a realistic desktop flight simulator to provide segmented behaviors useful to commanders in a mission debrief.

The objective was to be able to automatically parse the mission execution in order to ultimately build more effective technology tools for commanders. First, we defined the framework for a machine-learning capability to automatically decode mission execution. Next, we developed a realistic military flight scenario. We then developed a synthetic task environment (STE) in which two experienced fighter pilots flew a two-ship strike package, and we collected mission data as they performed several missions. This data included not only the behaviors of the aircraft and other scenario objects, but also the mission segmentation commonly performed during mission debrief. Data was extracted from the STE that is consistent with what can be extracted from a live aircraft, and following the missions the pilot's segmented the mission into distinct phases. Finally, we trained the ML model to perform the mission segmentation, and it learned to classify different parts of the mission into their respective phases.

The results showed similar classification accuracy for Linear SVM, random forest classifiers and feed forward neural networks (~ 75% accuracy). LSTMs and Kernel SVM performed more poorly, on average, but inconsistently demonstrated high classification accuracy for certain runs. Overall, the results demonstrated that mission phases could be correctly classified a majority of the time using snapshot techniques. Future work will expand the techniques to include time-series models that can better account for phase inertia.

## ABOUT THE AUTHORS

**Dr. Patrick L Craven** is a certified human factors professional with Lockheed Martin Rotary and Mission System where he focuses on developing advanced human-centered technologies, predominately for us in training systems. He has led efforts that developed human performance augmentation strategies, increased system usability, evaluated system and operator functionality, and enhanced interface design. He has experience designing and evaluating human-technology interactions including neurophysiological-based measures of cognition, human-autonomy interaction and teaming, command and control, handheld, aircraft, and intelligence analysis applications.

**Mr. Kevin Landers** is a Senior Software Engineer for the Advanced Simulation Center at Lockheed Martin and has been developing advanced technology solutions for five years. He is the lead software engineer for the LM-MIT program where he has implemented scenario review and evaluation capabilities, and has deep experience in implementing mixed-reality interfaces for DoD applications. He joined the Human Systems and Training (HST) Lab in 2016 and has experience in the design and implementation of data collection and visualization systems. Mr. Landers is graduate of The Ohio State University where he earned a BS in Computer Science and Engineering.

**Dr. Kevin Oden** Dr. Oden is the Principal Investigator of the Human Systems and Training (HST) Lab for the Advanced Simulation Center. In this role, he leads far-reaching R&D efforts that aim to accelerate the rate at which individuals and teams achieve expertise. Recent efforts have focused on the use of advanced sensing technologies to create objective measures of cognitive skills and emotional intelligence that are not directly observable. He partners with universities (MIT, Yale, and Drexel), small businesses and commercial companies to create new capabilities that improve and extend human performance. Dr. Oden holds a Ph.D. in Applied Experimental and Human Factors Psychology from the University of Central Florida, where he also earned a M.S. in Modeling and Simulation. A graduate from the University of Florida, Dr. Oden was also a Graduate Fellow Researcher at the Army Research Institute for Behavioral and Social Sciences with sponsorship from the Consortium of Universities located in Washington D.C.

**Mr. Ankit Shah** is an advanced PhD graduate student at MIT. His current research focuses on inferring formal logic specifications for complex tasks from demonstrations. His broader interests also include planning and learning within domains with hierarchical state descriptions. He is also interested in applying these algorithmic techniques towards enhancing the capabilities of human-robot teams. He has previously received his SM (2016) from the Department of Aeronautics and Astronautics at MIT, and his B.Tech. (2013) from the Indian Institute of Technology, Bombay (IIT-B).

**Dr. Julie Shah** is an Associate Professor in the Department of Aeronautics and Astronautics at MIT and leads the Interactive Robotics Group of the Computer Science and Artificial Intelligence Laboratory. Shah received her SB (2004) and SM (2006) from the Department of Aeronautics and Astronautics at MIT, and her PhD (2010) in Autonomous Systems from MIT. Before joining the faculty, she worked at Boeing Research and Technology on robotics applications for aerospace manufacturing. She has developed innovative methods for enabling fluid human-robot teamwork in time-critical, safety-critical domains, ranging from manufacturing to surgery to space exploration. Her group draws on expertise in artificial intelligence, human factors, and systems engineering to develop interactive robots that emulate the qualities of effective human team members to improve the efficiency of human-robot teamwork. In 2014, Shah was recognized with an NSF CAREER award for her work on “Human-aware Autonomy for Team-oriented Environments," and by the MIT Technology Review TR35 list as one of the world’s top innovators under the age of 35. Her work on industrial human-robot collaboration was also recognized by the Technology Review as one of the 10 Breakthrough Technologies of 2013, and she has received international recognition in the form of best paper awards and nominations from the International Conference on Automated Planning and Scheduling, the American Institute of Aeronautics and Astronautics, the IEEE/ACM International Conference on Human-Robot Interaction, the International Symposium on Robotics, and the Human Factors and Ergonomics Society.

# Man-Machine Interoperation in Training for Offensive Counter Air Missions

**Patrick L. Craven, Kevin B. Oden, Kevin J. Landers**
**Lockheed Martin Corporation**
**Orlando, Florida**
**patrick.craven@lmco.com, kevin.oden@lmco.com, kevin.j.landers@lmco.com**

**Ankit J. Shah and Julie A. Sha**
**Massachusetts Institute of Technology**
**Cambridge, Massachusetts**
**ajshah@mit.edu, julie_a_shah@csail.mit.edu**

## INTRODUCTION

Autonomous systems and robots are both heavily emphasized in technology roadmaps for the United States Air Force (Dahm, 2010). In Nov 2015, the Deputy Secretary of Defense spoke about how the Defense Science Board summer study on autonomy concluded that we are at an inflection point for AI and autonomy (Pellerin, 2015). In other words, there is recognition that currently developed technological systems are able to operate more intelligently and more independently, and their role in defense is about to become more prominent. In doing so, these autonomous systems will disrupt current practices that have been honed for human-dominated actions. Even when technology ultimately passes the tipping point of transitioning form automation into autonomy, these technological systems are unlikely to replace human decision making (Bradshaw, Hoffman, Woods, & Johnson, 2013; Murphy & Shields, 2012). As Bradshaw et al. describe it, "there's a need for the kinds of breakthroughs in human machine teamwork that would enable autonomous systems not merely to do things for people, but also to work together with people and other systems." Thus, it is anticipated that smart technological systems will serve in an advisory capacity in collaborate with humans who have the final authority on taking action.

The Observe-Orient-Decide-Act (OODA) loop was formulated by Col. John Boyd and describes the process by which fighter pilots engage threats (Feloni & Pelisson, 2017). This framework has been extended to many more activities, including the Plan-Brief-Execute-Debrief "Win Cycle." Pilots and practitioners of the win cycle are encouraged to find ways to get inside the OODA loop of an opponent through better information (observations), faster execution (flight controls, communication), etc. However, as the volume of available information relevant to a decision has increased the human decision-maker has become more encumbered and is challenged to make decisions efficiently. Consequently, there is a desire to augment human decision-making though artificial intelligence (AI) tools. The focus of the current work is on AI tools in the debrief process and, in particular, tools that allow for a commander to more rapidly process what occurred during mission execution and what lessons/learned (i.e., debrief focal points) should be identified.

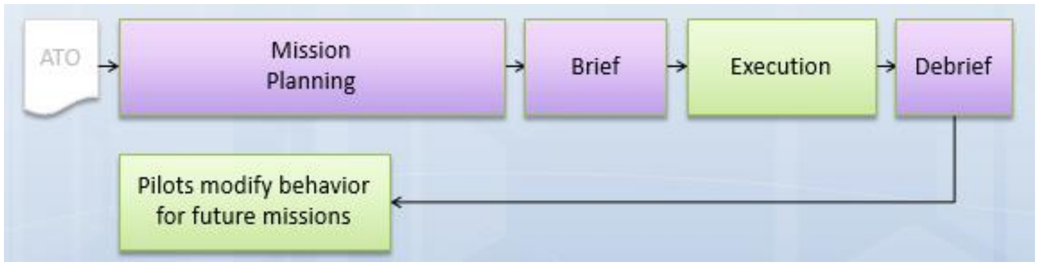


**Figure 1. The Full Mission "Win Cycle"**

The debrief process is composed of the following elements:

- **Mission Complete:** Return safely, maintenance debrief & intel collection
- **Debrief Premass:** Recreate truth data of specific actions during mission based on recorded data
- **Mass Debrief:** Jointly combine truth data from each flight/formation to create a mission overview. In the Mass Debrief the commander also relays debrief focal points (DFPs).

- **Element Debrief:** Flight lead identifies causal factors that relate to DFPs and the contributing roles of each pilot in the formation. Also, suggestions for instructional fixes to the formation members

As part of the current effort we have focused on technology and tools for enhancing the post-fight debrief process for large force exercise (LFE) training missions. Nellis Air Force Base hosts RED FLAG, a realistic combat training exercise with United States and allied forces. It was established in 1975, and provides an opportunity for a large group of mixed "blue" aircraft (attack, fighter, bomber, reconnaissance, electronic warfare, etc.) to fly missions against "red" air and ground threats (Nellis Air Force Base, 2012). At the conclusion of a training mission, the commander guides the participants through a recreation of the mission to create a ground truth from which lessons can be learned. A commander has to accurately process the activities of numerous manned (and unmanned) aerial platforms and link their mission execution to either accomplishing or failing to meet the mission objectives.

Advanced classification methods can be used to automatically code the pilot's actions in order to better match execution to objectives. Machine learning techniques can be used to model the process by which commander's review mission execution and relate that to the mission objectives. This model can then be used to provide tailored feedback to pilots (human- or autonomous-controlled) as part of a Mass Debrief in order to assist in ensuring that future missions are executed in accordance with objectives. Specifically, in the Mass Debrief portion an autonomous AAR combines truth data from each flight/formation to create a mission overview.

For this effort we focused on the Strike Package within an Offensive Counter-Air mission where two F-16s attack airfield infrastructure in order to damage threat offensive capability. Data was collected while experienced pilots flew simulated missions, and those data were used to train a series of AI classification models to automatically calculate when the aircraft switch from one phase of the mission to another.

## TECHNICAL APPROACH

The overall goal of this project is to develop a system that can observe a new mission execution, compare it with a model of a 'nominal' mission execution from its database, and determine the correspondence or discorrespondence between them. This requires having a computational model of the 'nominal' mission based on a database of previous executions of the mission. We propose building this model of a 'nominal' mission execution from data of demonstrated mission executions of a given mission type. In building this model, we begin with a supervised learning approach in which SME's provided training data input in the form of mission phase annotations. Based on the encouraging performance of supervised models, future research efforts will work to progressively relax the necessity for SME annotations. In this section, we define a nominal mission execution as a finite state machine with transitions constraints governing the transitions between modes of the state machine. Finally, we describe the manner in which trajectory segmentation algorithms are applied to automatically identify and segment the modes of a mission.

### Mission Representation

Complex combat missions can be seen as tasks that involve accomplishing multiple sub-goals in order to accomplish the overall goal of the mission. Thus the mission can be segmented into phases, where each phase corresponds to the part of the trajectory trying to achieve a particular sub-goal. The nature of most combat missions is such that a unique sequence of phases leading to mission completion may not exist, and there could be multiple equally valid sequence of phases that lead to the accomplishing the overall goal. Due to these characteristics, a finite state machine (FSM) is a suitable data structure to model a mission type, and to define a trajectory through the phases corresponding to a 'nominal' mission execution.

A FSM consists of a set of finite number of modes $S$ and a finite number of transition actions $\Sigma$. Each mode $s \in S$ corresponds to a phase of the mission. The transition actions $\Sigma$ encode the valid mode transitions and constraints on the aircraft and environment state governing the mode transitions. For a strike mission, the FSM can be defined as having the following interconnected states:

The goal of this work is to construct the FSM representing the nominal mission execution through data collected from mission demonstrations. This involves two steps. First, the system observes the trajectories and identifies the number of modes and the valid transitions between the modes. Next, the system learns the transitions constraints between the modes and the constraints apply within a phase. The discorrespondences between a nominal mission execution and a test trajectory can be identified as either a violation of the transition constraint, or a phase constraint. Discorrespondence could arise from an observed phase transition that does not correspond to one of the valid state transitions in the FSM. For the scope of this paper, we focus on the first of these steps, namely, supervised mission phase annotation to reconstruct the timeline of an individual flight element in terms of discrete mission phases.

**Trajectory Segmentation: A Data Driven Approach**

Central to the idea of representing a mission as a FSM is the notion that the test trajectory can be segmented into phases, and each phase can be classified as one of the phases defined by the FSM. This is an important problem in robotics and human activity modeling called trajectory segmentation. Trajectory segmentation allows breaking up of complex demonstrations into smaller similar phases which are shared across multiple trajectories. In this section, we first formally define the trajectory of a flight. Next, we define the trajectory segmentation problem, and cast it as supervised or unsupervised learning problem based on the availability of expert annotation.

Given a mission with a set of modes $\boldsymbol{M} = \{m^{(1)}, m^{(2)}, \dots, m^{(n)}\}$. $\boldsymbol{s} \in \mathbb{R}^{\boldsymbol{d}}$ is a vector of all the state variables which are relevant to the mission. It can include, the navigation data of the aircrafts, the system configuration of the aircraft, and the sensor readings recorded by the onboard systems of the aircraft. Some of these state variables may be discrete and the others may be continuous. Then, a mission execution $\boldsymbol{D}^{(i)}$ is defined as follows:

$$\boldsymbol{D}^{(i)} = [(\boldsymbol{s_i}, m_1), (\boldsymbol{s_i}, m_1), \dots, (\boldsymbol{s_t}, m_t), \dots (\boldsymbol{s_k}, m_k)]; m_t \in \boldsymbol{M}, \boldsymbol{s_t} \in \mathbb{R}^{\boldsymbol{d}} \quad (1)$$

The state $\boldsymbol{s}$ is always observable. The mission mode $m$ may be an observable or a latent variable depending on the machine learning paradigm selected. The supervised and the unsupervised paradigms of machine learning are described next.

**Problem Formulation**

A supervised learning formulation is trained on a dataset which contains demonstrations that includes both the state trajectories and the mode labels for each point in time. Thus the training dataset can be described as

$$\boldsymbol{D} = \{\boldsymbol{D}^{(i)}\}; i \in \{1,2, \dots, n_{demos}\} \quad (2)$$

Where each demonstration $D^{(i)}$ is as defined in Equation 1. The task of a supervised learning model would be to provide a predicted mode level for each time stamp in a new 'test' trajectory that only contains the state $\boldsymbol{s}$. Thus supervised learning algorithms learn a function $f$ such that, given a test trajectory

$$D_{test} = [\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_t}, \dots, \boldsymbol{s_{k_{test}}}] \quad (3)$$
$$f(D_{test}) = [\widehat{m}_1, \widehat{m}_2, \dots, \widehat{m}_t, \dots, \widehat{m}_{k_{test}}] \quad (4)$$

We begin by adopting a supervised learning to mission trajectory segmentation. For this phase of the project, we aim to develop a supervised learning model to predict the mode labels for a test trajectory based on annotations provided by the pilots. This would be valuable in identifying the features which are most indicative of mission phases and the level of complexity of the model necessary to label the modes. In this section, we first describe the dataset which was collected for training the supervised learning models. Next, we present the pre-processing steps taken on this data to make it suitable for supervised learning algorithms. We provide a list of machine learning algorithms that were used to construct the learning models and the metrics used to evaluate the models and finally we present the results of the data analysis.

## METHOD FOR TRAINING DATA COLLECTION

The research team designed a data collection event to log realistic data from pilots completing missions within the STE in order to create a ground-truth training data set for the development of the machine learning classification models. In this event, a pair of pilots (lead and wingman) flew mission scenarios in which their actions and aircraft data were logged. At the conclusion of the scenarios the pilots each scored the mission phases using a computerized tool.

Two pilots with a total of 3200 hours flying 4th and 5th generation fighters participated in this event, and they were both very familiar with this type of air-to-ground mission. In addition, they have experienced as mission commander at USAF's Red Flag training exercises and have participated in five Red Flag exercises in the last three years. The lead pilot did not participate in the development of the scenarios. The wingman developed the mission briefing and worked with the design and development team to verify the realism of the developed scenario as well as identify the relevant mission phase labels. However, key factors such as the exact threat locations as well as the existence and timing of pop-up threats were kept hidden from both pilots.

The wingman roleplayed the mission commander and briefed the lead pilot on the relevant mission aspects. Both pilots were given an opportunity to get familiar with the simulator environment by flying a sample mission. They were also given the list of task phase labels and given an opportunity to try the task phase labeling tool.

After completion of the scenarios, pilots were asked to use the Flight Data Annotation Tool (FDAT) (shown in Figure 3) to provide labels for different phases of the mission. The FDAT has been hard-coded with the labels and corresponding definitions for phases of the mission. These phases were developed in conjunction with pilot subject matter experts (SMES). The list of phases are:

1. ***Push***: Start of the strike route on time and in the correct formation
2. ***Legs***: Following the flight segment between waypoints during ingress.
3. ***Threat Identification:*** Observation of a threat in the vicinity
4. ***Threat Avoidance / Mitigation***: Actions taken to avoid an observed threat. Could include altering a planned route and/or counter-measures.
5. ***On-Target***: Period of time when the pilot focuses on making sure that the weapon is released on time.
6. ***Egress***: Return to base

## Synthetic Task Environment

The team used a variety of technological tools in order to immerse pilots in a realistic environment in which they were asked to perform a realistic task so that a corpus of ground-truth training data could be generated. The Synthetic Task Environment (STE) was built around Prepar3D v4 (Lockheed Martin, 2018), which is a visual simulation platform that allows users to create training scenarios across multiple domains but has its roots in aviation simulation . Prepar3D engages users in immersive training through realistic environments. Ideal for commercial, academic, professional, or military instruction. Prepar3D can be used to quickly create learning scenarios anywhere in the virtual world, from under water to sub orbital space.

The scenarios built for this are all two ship missions piloting F-16C aircraft armed with AIM-9X and GBU-32. Known ground enemy threats are distributed along the flight plan. Additionally, variations on the base scenarios involve pop-up threats that show up on radar but were unknown during the scenario brief. The pilots must recognize and react to these threats. Ultimately, each scenario has a target or set of targets that must be destroyed by the two-ship team.

Each pilot is set up on a laptop running Prepar3D with the data collection plugin. The laptops run over a Distributed Interactive Simulation (DIS) network. This allows the pilots to fly together in the scenario sharing targets, threats, etc. In order to support this effort a Data Collection Plugin was developed to extract flight data from Prepar3D. Additionally, the Data Annotation Tool was developed to allow the pilots to manually segment their flights. The output of this tool is truth data that could be used to train the machine learning algorithm.

## Task Scenarios

The scenarios all follow a basic air-to-ground attack as part of an Offensive Counter-Air operation. Previous sorties have cleared known enemy anti-aircraft capabilities and created a cleared corridor to approach the target. The pilots were asked to approach ISIS-controlled Raqqa, Syria from the east (Turkish and Kurdish controlled airspace). They were informed that a coalition operation was providing softening of ground targets and infrastructure to support Syrian Democratic Forces (SDF) and Kurdish militias in an effort to wrestle control of Raqqa from ISIS. Operations took place during dusk. The pilots controlled a simulated F-16C which was armed with four GBU-32 JDAM (1K lbs.) bombs and two AIM-9X air to air missiles. Threats included SA-2 and SA-6 anti-aircraft batteries.

Four major variations of scenarios were created in which the target was either an air base, fueling station, radar, or armored airport security vehicles. Within the four scenario, slight variations were introduced in order to make three versions of each scenario. The A version of the scenarios had only the known threats, whereas the B and C versions of the scenarios introduced pop-up anti-aircraft threats that would appear on the radar during egress.

**Simulator Data Recorder**

The data collection plugin called the Data Bridge was created using the Prepar3D Development Kit (PDK) and rests as an add-on inside of Prepar3D. This plugin tracks flight information on entities in Prepar3D. The plugin creates menu options in Prepar3D to allow users to interact with it. There are options to start and stop the plugin. Outside of these menu options, there is a configuration file that rests in Prepar3D's add-on folder with the plugin. This is a .csv file that stores configuration options for collection frequency, variable list, entities being tracked, etc. The plug-in features enabled it to:

- Configure the frequency of collection (defaults to every frame ~60Hz)
- Specify via configuration file which variables and units were being collected
- Configurable set of tracked entities
- User-initiated Start and Stop for collection
- Generate a timestamped CSV file

The plugin can be configured to collect custom variables for different entities in the simulation using information based on the list of simulation variables provided on the Prepar3D website. The data collected out of Prepar3D gets exported as a comma separated value (CSV) file by default. There is also an option to send the data to an external server by specifying an IP address and port number.

Each mission had a two ship friendly force formation with 329 streams of data from each of the friendly aircraft that included navigation data, fuel levels, flight instrument states and flight plan data. For each of the objects in the scenario including the SAM launchers, the targets and missiles, 15 streams of data were collected.

**Data Annotation Tool**

In order to train the machine learning approach, in Phase 1 we also created truth data for segments of the flight. After running through each scenario, the pilots would annotate the data by marking the different segments or phases of their flight.

This data was merged with the P3D flight data to provide truth data for segments of flight. The merged data combined with pre-mission brief and contracts are the inputs to the machine learning algorithm.

After the mission was flown in the simulation environment, the pilots were asked to annotate the phase of flight during each point of the mission execution. The available labels were: Push, Legs, On Target, Egress, Threat Identification and Threat Mitigation. As each of the pilots were individually asked to label the flight data and we have 11 separate mission executions, we have 22 annotated demonstrations. Of the data streams available, the ones selected for the supervised learning models were:

- Positions
- Body frame velocities
- Attitude
- Body frame angular rates
- Body frame accelerations
- Next waypoint index
- Waypoint bearing
- Waypoint range

The aircraft flown by the lead pilot is henceforth referred to as *ownship*, whereas the wingman's aircraft is referred to as *wingman*. For each of the objects other than the ownship, only the positions were added to the feature set. Based on this dataset, we can train two different types of models.

- Ownship models: models trained on the data that includes only the information of the ownship and the phase annotations provided by the ownship pilot.
- Ownship + wingman models: These models include the data of the ownship and the position of the wingman along with the phase annotations provided by the own ship pilots.

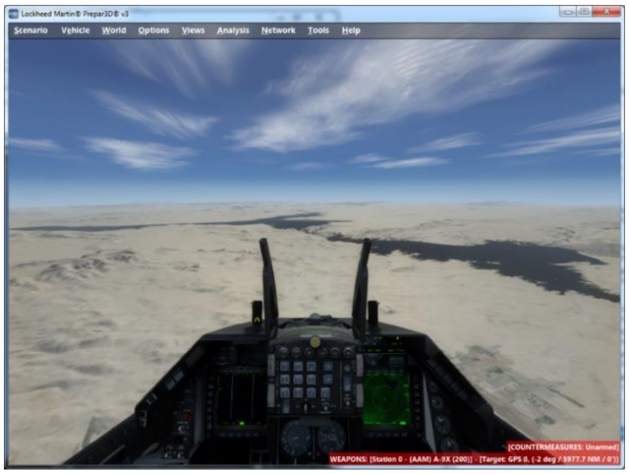
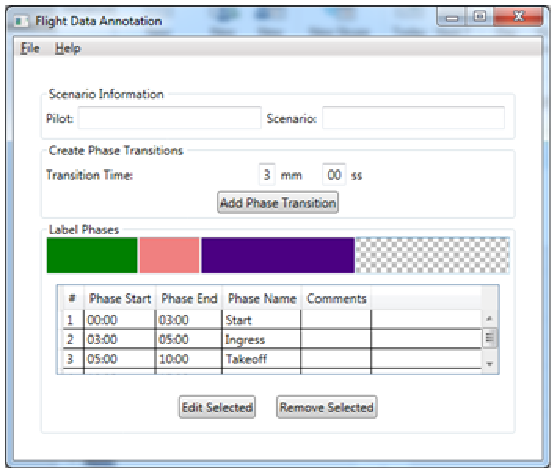

Lockheed Martin® Prepar3D® v3
Scenario Vehicle World Options Views Analysis Network Tools Help
[COUNTERMEASURES: Unarmed]

**Figure 2. An out-the-cockpit view from the F-16 as flown in the missions**



Scenario Information

Pilot:

Scenario:

Create Phase Transitions

Transition Time: 3 mm 00 ss

Label Phases

| # | Phase Start | Phase End | Phase Name | Comments |
|---|---|---|---|---|
| 1 | 00:00 | 03:00 | Start | |
| 2 | 03:00 | 05:00 | Ingress | |
| 3 | 05:00 | 10:00 | Takeoff | |

**Figure 3. GUI of the Flight Data Annotation tool used to code the mission phases**

## ANALYSIS METHODS

Supervised machine learning algorithms are trained by defining a hypothesis space of parametric classification functions and the process of training the model involved selecting the optimal values of the parameters. These parametric classification functions work best with separable and unbiased data. Thus the raw data needs to be transformed into an appropriate set of features which would function best with the supervised learning technique selected. We perform three different pre-processing steps. First the relevant data streams must be down-selected to eliminate redundant data streams. Next, in case of data dealing with positions of the objects, selecting an appropriate frame of reference can help with generalizing the result. Finally, the numerical optimization approaches used to select the parameter values work best when all the data sources are bound to similar values. Thus the data needs to be appropriately scaled to "equalize" the values in the data streams.

The frame of reference selected for our models were as follows:

1. Ownship:
    a. Position: Target centric navigation frame[1]
    b. Velocities: Body frame[2]
    c. Accelerations: Body frame
    d. Attitude: Navigation frame to body frame (Euler angle representation)
    e. Angular rates: Body frame
    f. Waypoint: Vehicle centric radial coordinate system

[1] Navigation frame is a frame with X aligned with local North, Z aligned with down and Y completes the right handed coordinate system.
[2] Body frame has its X axis with through the nose of the vehicle, Z is down through the underside of the vehicle and Y completes the right handed coordinate system.

2. Wingman positions: Vehicle centric frame aligned with the vehicle navigation frame
3. Scenario object positions: Vehicle centric frame aligned with the vehicle navigation frame

We also provide the index of the next waypoint as an input to the learning algorithms. This is a discrete variable that can take one of the many possible values. We encode that as an indicator vector of a dimension equal to the number of categories. The dimension corresponding to the discrete value of the variable is assigned value 1 and the other dimensions are 0. Care must be taken to provide the expected number of categories for all discrete variables.
Finally it is important that the feature values provided as input to the parametric classification functions are of a similar magnitude. In our scenario the magnitude of numbers representing positions, velocities and attitudes vary significantly. Hence we apply linear transforms to each of these data streams independently, such that each coordinate has a mean 0 and standard deviation of 1. With these pre-processing steps the dataset is well conditioned for applying the supervised learning algorithms.

**Supervised Learning: Snapshot Classifiers**

We consider two different types of supervised learning algorithms; snapshot classifiers and time series classifiers. Snapshot classifiers learn a classification model that takes a fixed size input and generates the most likely class label as an output. Snapshot classifiers are relatively simple models with fewer parameters. They make a simple causal assumption that the observed data is conditioned only on the current class labels, and that observations at different points in a time series are independent. While this is not an accurate assumption for time series data in general, it can be a reasonable assumption to make in certain feature spaces. The classifiers generated by this method are typically more interpretable as it is easy to query the feature vector that would most likely get classified into a target class. However the most important drawback of snapshot classifiers is that it does not utilize the correlations between successive observations. In our implementation of snapshot classifiers, we use a windowed state representation where instead of providing the model with the state value at a single time instant, we provide it with state observations in a window of time. This would allow the model to learn short period characteristics in the data. We use the following four varieties of snapshot classifiers.

**Linear Support Vector Machines (Linear SVM)**
Support vector machines are a linear classification technique which train a classification function of the form

$$H(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{b}) = \left[\left[\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b}\right]\right] \quad (5)$$

Where $\boldsymbol{x}$ are the features and $\boldsymbol{\theta}, \boldsymbol{b}$ are the parameters set by the classifier. SVMs try to identify a separating hyperplane in the feature space which serves as a decision boundary in selecting the target class. While the basic SVM classifier is a binary classifier, it can be adopted to multiclass classification either using a one vs one ($\binom{n}{2}$ hyperplanes) approach or a one vs rest (n hyperplanes) approach. Typically the power of linear SVM classifiers increases as additional features are added to the classification problem. However, the added features must not be redundant and care must be taken to not introduce features which might be spuriously correlated to a bias in the training data.

**Kernelized Support Vector Machines (Kernel SVM)**
Here the support vector machine model is learnt for a transformed feature space with a transform $\Phi(x)$, and the inner product in the transformed space is defined by the function $K(x_1, x_2)$ such that

$$K(\boldsymbol{x_1},\boldsymbol{x_2}) = \boldsymbol{\Phi}(\boldsymbol{x_1})^{\boldsymbol{T}}\boldsymbol{\Phi}(\boldsymbol{x_2}) \quad (6)$$

Where $K$ is called the kernel function. The SVM model trained on the transformed feature space

$$h(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{b}) = \left[\left[\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{b}\right]\right] \quad (7)$$

is equivalent to

$$h(\boldsymbol{x};\boldsymbol{\alpha},\boldsymbol{b}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x},\boldsymbol{x_i}) + \boldsymbol{b} \quad (8)$$

Where, $\boldsymbol{x_i}$ are all the data points in the training dataset. Thus kernel methods do not scale very well with large datasets as they need to store a classification function with one parameter per sample in the training set. Kernelization to a higher dimensional feature space can help kernel SVMs learn non-linear decision boundaries. However the choice of a kernel is an important design decision.

**Random Forest Classifiers (RF)**
Random forests classifiers are an ensemble algorithms that involves learning multiple decision trees and using boosting methods to provide the best target prediction given the prediction of multiple classifiers. The decision boundaries in this case are in form of bounding boxes in the feature space.

**Feed Forward Neural Networks (FFNN)**
Neural networks are function approximation techniques with a very large parameter space. They attempt to learn a classifier function of the form

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}) \quad (9)$$

For neural networks tasked with classification, $\hat{\boldsymbol{y}}$ represents a vector representing the categorical distribution of the classes. The predicted class is usually the most probable class label. Due to the large possibilities of defining the network architecture, and the type of layers and activation functions used, it is typically very hard to determine the entire space of functions that a given network can represent. However there is consensus that given a network with sufficiently large number of hidden units can accurately model any function. In our current work, we maintain a constant network architecture and vary the learning rate and number of epochs trained.

**Supervised Learning: Time Series Classifier**

Alternatively, time series classifiers have the ability to model variable length sequences of vectors as inputs to output the most likely class label. These classifiers are designed to learn temporal correlations in the data. There are also different type of models available in the literature which assume different statistical dependencies of the data. However these classifiers do not have interpretable classification functions. They are also hard to optimize as they have a larger number of parameters, and suffer from the problem of vanishing parameter gradients when propagated through time. We used LSTM as the time series classifier in this effort.

**Long-short Term Memory Networks (LSTM)**
LSTMs are a type of recurrent neural network (RNN) where the previous output of the network is used as one of the inputs to the next layer of the network. The weight of the network are shared across time. This allows RNNs to model sequences. With the output layer similar to the feedforward neural networks, RNNs can be used for sequence labeling tasks like the one in this work. LSTMs are modifications of simple recurrent neural networks that introduce a cell state and gated updates to the cell state to allow the network to remember the state for a long period of time or forget the state when given certain inputs.

The standard parameter settings used to train the different models are described next. We also include a list of tunable hyperparameters. The first set of results compares the accuracy of the classifiers when 'standard' hyperparameter values are used.

## RESULTS

Table 1 represents the standard hyperparameter values used for each of the different types of classifiers.
The metric used to evaluate the results was the leave-one-out cross validation technique. Of the 11 available scenarios, the models were trained on data from 10 scenarios and one of the scenarios was held out. This was repeated with each of the scenarios being held out for training once. The final value reported was the mean of the accuracies for all the cases. As the variation in the accuracies was high, we also report the largest and the smallest accuracy values.

**Table 1: Standard Hyperparameter Values**

| | |
|---|---|
| Linear SVM | Window size of 5 |
| Kernel SVM | Radial basis function (RBF) kernel<br>Window size: 5 |
| RF | Window size: 5<br>Number of decision trees: 5<br>Maximum tree depth: unbounded<br>Min samples in a leaf node: 25<br>Minimum samples to branch the tree: 50 |
| FFNN | 6 hidden layers with scaled exponential linear units (SELU)<br>Optimizer: Adam with learning rate 0.01<br>Trained for 20 epochs |
| LSTM | 3 hidden layers: dense layer followed by 2 LSTM layers<br>Optimizer: Adam with learning rates 0.005 followed by 0.0025<br>Trained for 50 epochs with each learning rate. |

The mean, maximum, minimum leave-one-out cross-validation (LOOCV) accuracies for ownship data and ownship + wingman data are shown in Table 2.

**Table 2. LOOCV Accuracies for Only Ownship and Ownship + Wingman**

| Classifier | Mean | Max | Min | | Mean | Max | Min |
|---|---|---|---|---|---|---|---|
| | *Model results using only ownship data* | | | | *Model results using ownship + wingman data* | | |
| Linear SVM | 76.5 | 98.6 | 50.1 | | 75.3 | 95.2 | 50.0 |
| Kernel SVM | 56.4 | 79.1 | 37.3 | | 66.8 | 95.7 | 42.0 |
| RF | 79.9 | 92.0 | 68.3 | | 79.5 | 94.0 | 65.6 |
| FFNN | 74.5 | 92.5 | 49.9 | | 73.0 | 94.1 | 49.8 |
| LSTM | 65.4 | 70.1 | 59.8 | | 65.4 | 70.2 | 59.5 |

A one-way ANOVA was performed on the 11 results for each of the five classifiers with each classifier having two possible training data sets (ownship, or ownship + wingman data). Models were generated from data that included either just ownship data identified as "[O]", or ownship + wingman data identified as "[OW]." The ANOVA showed a significant main effect ($F = 3.75$, $p < .001$). Dunnett's post-hoc for multiple comparisons showed the following pairs of classifiers were significantly different:

Kernel SVM [O] from RF [O], (p < .001)
Kernel SVM [O] from FFNN [O], (p < .05)
Kernel SVM [O] from RF [OW], (p < .01)
RF [O] from LSTM [O] (p < .01)
RF [O] from LSTM [OW] (p < .01)

**Table 3. Accuracies of Different Combinations of Ownship with Weapons and Communication Data**

| Classifier | Mean | Max | Min | | Mean | Max | Min | | Mean | Max | Min |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Model results using ownship and weapons* | | | | *…using ownship and communication* | | | | *…using ownship, weapons, and communication* | | |
| Linear SVM | 76.7 | 98.6 | 50.3 | | 76.8 | 98.7 | 49.8 | | 76.7 | 98.7 | 50.0 |
| Kernel SVM | 56.2 | 78.0 | 39.2 | | 56.1 | 78.0 | 39.2 | | 56.1 | 78.0 | 39.2 |
| RF | 80.1 | 93.1 | 66.8 | | 80.4 | 93.8 | 65.8 | | 78.7 | 93.2 | 67.4 |
| FFNN | 73.4 | 94.9 | 55.2 | | 75.6 | 95.0 | 56.0 | | 73.7 | 95.7 | 54.7 |
| LSTM | 70.1 | 93.9 | 38.4 | | 70.5 | 91.2 | 46.9 | | 70.3 | 92.4 | 40.2 |

Finally, we also explored the effect of adding indicator variables for weapon release and voice communications between the pilots. These variables were included in the training data along with the ownship data. The accuracies of these models are listed in Table 3.

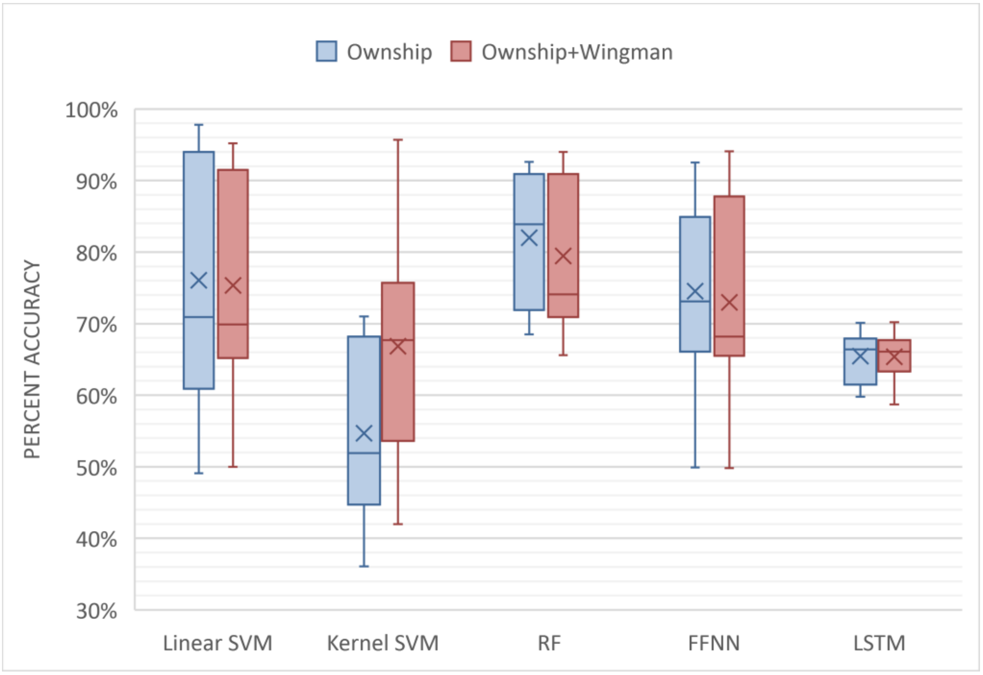


**Figure 4. Box-and-Whisker plot of accuracy by classifier technique.**

## DISCUSSION

These results indicate that Kernel SVM and LSTM classifiers were significantly weaker than RF and FFNN classifiers. It was noted that the performance for Linear SVM, RF and FFNN were very similar. LSTMs perform poorly as compared to the snapshot techniques in our first iteration of the model, however their performance is fairly consistent across all the held out scenarios. In contrast, the performance on the held out scenarios was highly variable in case of the linear snapshot techniques. This variability seen in conjunction with very high maximum accuracy indicates that some held out test trajectories were substantially similar to the training data. Further work is required to determine whether an increase in accuracy with time series classification approaches is possible with better parameter tuning and training methods. Few of the future directions are detailed below.

The addition of communications and weapons release signal does not seem to increase the classification accuracy for the snapshot techniques, but the classification accuracy does increase by a large margin for the LSTM classifiers. One potential reason for this increase could be the fact that these features are less important at a single time instant but more important in the context of a sequence of events. Thus radio communications were seen to occur at leg transitions or when the aircraft was inbound on its final leg to the target. We also observe a higher variability in accuracy of LSTM models as compared to the ownship features model. Further the best and the worst performing scenarios were consistently the same ones. The worst performing scenarios were the ones which had significant portions of flight in Threat mitigation phase.

It is also interesting to note that adding the wingman positions to the feature set did not significantly affect the accuracies. That result might be due to the decision to represent the wingman positions in local navigation frame. As a result, the positions for the lead and the wingman would be reflected about the origin with the same label. This representation may not be informative, and rather an alternate representation (e.g. representing the wingman position in the body frame) may be more informative. In this frame, the position of the wingman is described relative to the viewpoint of the pilot versus being aligned to a fixed orientation. Another possibility is that close physical coordination between the lead and the wingman was not primarily important to successful mission completion; instead another coordinative feature (e.g. information exchange) may have been more important.

This is the first known research study in which experienced fighter pilots conducted realistic mission within a simulator environment and AI was used to categorize the mission phase based on aircraft behavioral data. While rule-based algorithms may provide accurate phase estimations in some situations, real-world training mission executions can often have deviations that are not anticipated and hence not defined in a more strict rule-based approach. By using a supervised-learning AI approach an expert commander can provide examples for the AI to learn and assimilate the rule-structure using a bottom-up data-driven rather than top-down approach. This provides an enhanced ability to accurately categorize cases that would otherwise violate a strict definition-driven rule. In the current data set there were likely not sufficient examples to clearly differentiate linear from nonlinear classification methods (as evidenced by the high Linear SVM performance compared to RF), but it does show that the several of the nonlinear techniques can be as good as the linear approach while also providing growth potential for capturing cases that are not linearly separable.

**Lessons Learned**

Our pilot volunteers spent two full days working with us to understand the task environment, execute the missions, and label the data. While an attempt was made to familiarize both pilots with the labeling process, there is the possibility that they conceived of the segment labels in a different way. This likely introduced some variation in the training data set. Despite this variation, it is noteworthy that the models performed as well as they did. In the future we plan to intentionally measure inter-rater reliability of labeling of mission execution data and to mitigate the effects of variation with more explicit instructions to the pilots for performing mission segment labeling.

In these models the hyperparameters were set to the standard values with minimal hyperparameter optimization. While the results indicate that the features selected features are important indicators of mission phase, better hyperparameter optimization can lead to higher accuracies. We hypothesize that additional features such as time of weapon release, and an indicator of radio communications between pilots, may be important in detecting mode changes. These features need to be incorporated into the training models.

The effort to date has focused on implementing the supervised learning algorithms and training the models using standard pre-processing techniques to identify a fixed set of informative features. However, the set of informative features will likely vary as a function of mission phase. Training a learning model using a large superset of important features may result in spurious correlations due to biases in the dataset. Next steps in the research will investigate automated techniques for feature selection using ensemble methods to generate the final mode label.

## ACKNOWLEDGEMENTS

This work was supported by the USAF AFMC AFRL through the Learning for Man-Machine Interoperation and Training (LM-MIT) program under Contract Number (FA8650-16-C-6765). Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the Department of Defense. We would like to thank Dr. Kevin Gluck of the Air Force Research Labs (AFRL) for his sponsorship of the LM-MIT program. Additionally, we would like to thank Mr. Zach "Zen" Wallace and Mr. Scott "Carp" Carpenter for their aircraft and piloting expertise.

## REFERENCES

Bradshaw, J. M., Hoffman, R. R., Woods, D. D., & Johnson, M. (2013). The seven deadly myths of" autonomous systems". *IEEE Intelligent Systems, 28*(3), 54-61.

Dahm, W. (2010). *US Air Force chief scientist report on technology horizons: A vision for Air Force science & technology during 2010-2030*. Retrieved from

Feloni, R., & Pelisson, A. (2017). A retired Marine and elite fighter pilot breaks down the OODA Loop, the military decision-making process that guides 'every single thing' in life. *Business Insider*.

Lockheed Martin. (2018). Prepar3D Product Overview. Retrieved from https://www.prepar3d.com/product-overview/

Murphy, R., & Shields, J. (2012). The role of autonomy in DoD systems. *Defense Science Board*.

Nellis Air Force Base. (2012). 414th Combat Training Squadron "Red Flag". Retrieved from http://www.nellis.af.mil/About/Fact-Sheets/Display/Article/284176/414th-combat-training-squadron-red-flag/

Pellerin, C. (2015). Work: Human-Machine Teaming Represents Defense Technology Future [Press release]. Retrieved from https://www.defense.gov/News/Article/Article/628154/work-human-machine-teaming-represents-defense-technology-future/