FINAL REPORT
Colorado Advanced Software Institute

MODELING STUDENT PILOTS FOR INTELLIGENT TRAINING

Principal Investigators:	Charles Anderson, Ph.D. Associate Professor Bruce Draper, Ph.D. Assistant Professor Department of Computer Science Colorado State University, Fort Collins
Graduate Student:	David Peterson Department of Computer Science Colorado State University, Fort Collins
Collaborating Company:	SymSystems Thomas Donohue, Vice President Representative

Project Title:	MODELING STUDENT PILOTS FOR INTELLIGENT TRAINING
Principal Investigators:	Charles Anderson, Ph.D., and Bruce Draper, Ph.D.
University:	Colorado State University, Fort Collins
Collaborating Company:	SymSystems
Collaborating Company Representative:	Thomas Donohue, Vice President

As authorized representative of the collaborating company, I have reviewed this report and approve it for release to the Colorado Advanced Software Institute.

Signature

Date

Contents

MODELING STUDENT PILOTS FOR INTELLIGENT TRAINING

Abstract

SymSystems is developing pilot training systems for NASA. Based on their experience, student pilots not only learn at different rates, but in different manners. Some students, for example, tend to overcompensate early in their training while others experience problems with take-off and landing. The challenges presented to each student must be tailored to their unique learning experiences. This requires an intelligent training regime that exploits a model of each student that predicts where the student's performance will be deficient. This project developed artificial neural network algorithms for predicting student actions. Results show that modular neural networks can discover when a student acquires the various skills necessary to fly the simulated aircraft. This provides a basis for future intelligent training systems.

1 Problem

In 1995, the FAA and NASA formed a consortium to advance the concept of a small aircraft transportation system. The consortium's primary focus is the Advanced General Aviation Transport Experiments (AGATE). The goal of this government-industry-university partnership is to develop technologies for safety, affordability, and ease of use for single pilot, single engine, near all weather transportation aircraft and related training, airspace, and ground infrastructure systems. Targeted training technologies include cockpit-based onboard training systems, computer-based training desktop devices, simulation, virtual environment systems, and expert systems. It is in this field of advanced training concepts for General Aviation (GA) pilots that this research is targeted.

Teaching a person to fly a GA airplane is a complex and time consuming task. The approach of using one-on-one training with an instructor in the co-pilot's seat of a real airplane has been used for decades. While real aircraft offer the ultimate in training realism, they are expensive to operate and maintain, and safety considerations restrict the tasks that can be practiced. Augmenting instructor training in a real airplane with computer simulation promises to reduce training costs and allow instruction in emergencies in a forgiving environment

Today, low end flight training devices (FTDs) are sometimes employed as self paced instruction for specific tasks. For the GA marketplace these devices are classified as part task trainers (navigation, reading basic flight instruments, etc.). They do not have the aircraft system comprehensiveness, accuracy, nor the simulated reality of the full flight simulators used by major airlines and the military. On the other hand, FTDs don't have the high development and operation cost of full flight simulators. With modern advances in computer graphics, microcomputer processing power, and electro-mechanical control loading, it is possible to build FTDs that approach the capabilities of full flight simulators at a significantly reduced price.

For this CASI research project, we investigated augmenting FTD simulation capabilities with adaptive techniques that model the abilities of each student pilot. Ultimately the ability to predict each student's capabilities and to adapt the simulation according to their strengths and weaknesses will reduce the number of human instructor hours required and allow for better training over a shorter period of time. Our first thrust at this broad problem was to develop a system that builds dynamic models of a student's current flying ability and predicts their trajectory in FTD simulations.

2 Objective

If large numbers of new pilots are to be trained for the general aviation (GA) market as envisioned by the NASA/FAA AGATE project, then the training process for new pilots must be made cost-effective. This implies that as much of a pilot's training as possible should take place on low-cost flight-training devices (FTDs), thereby reducing the need for expensive in-flight supervised instruction. Our long-term goal of an intelligent training system for general aviation is to maximize the training benefits of FTD simulations.

The immediate goal of this project was to lay the foundation for a future intelligent training system by developing techniques to model the skills and performance of individual students. These models can be used to develop intelligent training systems that 1) select training scenarios that challenge but not overwhelm the student, 2) selectively exercise skills in which a student is deficient, while only occasionally reinforcing previously mastered material, and 3) actually exert forces on control surfaces in situations where the student consistently makes mistakes, like a golf instructor helping a novice player with their swing. Such intelligent training systems, however, will rely on techniques such as those developed in this project to 1) develop accurate models of individual pilots and 2) verify that the learning process of pilots differ sufficiently to warrant individualized training regimens.

This project developed techniques for modeling the performance of individual novice pilots. Whereas most intelligent tutoring systems model students as decision trees or through sets of production rules, we believe these techniques may not be effective in the real-time, continuous-process world of general aviation. We therefore developed control-model techniques which learned a mapping from the current state of the plane (including its recent history) to the pilot's predicted action[1]. One challenge in developing these control models is to select a state space, since both the plane's current position/orientation/speed and the recent history of these variables may determine a pilot's action, particularly given varying reaction times and the tendency of many novice pilots to overcompensate for small errors. Another challenge is to learn a non-stationary process model for the student, whose performance improves with each trial. A significant result of this work is that we verified that novice pilots vary significantly from each other in what and how they learn.

3 Approach

We built a system that develops predictive control models of individual student pilots, and compared these models across students to measure how similar or dissimilar they are at different points during a training regimen. We believe that we succeeded in modeling a pilot's behavior largely because of the "highway in the sky" concept underlying the AGATE project. Part of the motivation for AGATE is that general aviation pilots will enter a flight plan into the plane's computer before take off. The computer system will then use this information to project a virtual 3D highway onto the plane's windshield, complete with road signs for auxiliary information. In this way, flying a plane becomes much more like driving a car; pilots simply keep the plane slightly above the (virtual) highway, and slow down or speed up when (virtual) road signs tell them to. A mock-up of the "highway in the sky" is shown in Figure 1.

The virtual skyway reduces flying to a reactive task. Pilots react to the projected highway in front of them, and the act of flying becomes the hand-eye coordination task of keeping the plane over the skyway. This does not imply that flying becomes trivial; pilots must still control a plane in 3D to keep it on course relative to the pre-planned highway, a difficult task in some many weather conditions or if the flight plan calls for relatively tight curves (for example, on take-off or landing). Nonetheless, it does make flying a primarily reactive skill in which pilots respond to the information immediately before them, rather than an abstract navigation task in which pilots respond to mental estimates of their current and desired position. Therefore, we developed a gated network of neural network "expert" controllers, each automatically tuned to a different aspect of flying, that models a pilot's behavior in terms of a small set of immediately available variables, summarizing the position, velocity and acceleration of the plane and the local position and curvature of the path.

3.1 Related Work

It should be noted that researchers have sought to develop intelligent tutoring systems since at least the early 1980's (e.g. [4]; see [5] for a survey of early work), with a great deal of recent work focusing on medical training applications (e.g. [6]). What separates this project from most of the literature in intelligent tutoring is that we are not trying to model students as they learn abstract concepts. Instead, we are trying to teach a hand-eye coordination task, almost at the level of a motor skill. As a result, we do not need to model a student's concepts and ideas about flying, but rather their reaction time, their tendency to overcompensate of undercompensate for positional errors, and how well they control the plane. This is why we use the term "intelligent training" as opposed to "intelligent tutoring" (in so doing we are keeping with the terminology of the NASA Conference on Intelligent Computer-Aided Training [7]).

Although most intelligent tutoring systems have modeled their students using decision trees (or sometimes production rules), we do not believe that such hierarchical, discrete models are appropriate for hand-eye coordination tasks. Instead, we believe that the domain shift from abstract ideas to motor

[1]Note that we are learning to predict what the pilot *will* do, not what they *should* do. The former is learning a model of the pilot, whereas the second would be learning to fly the plane.

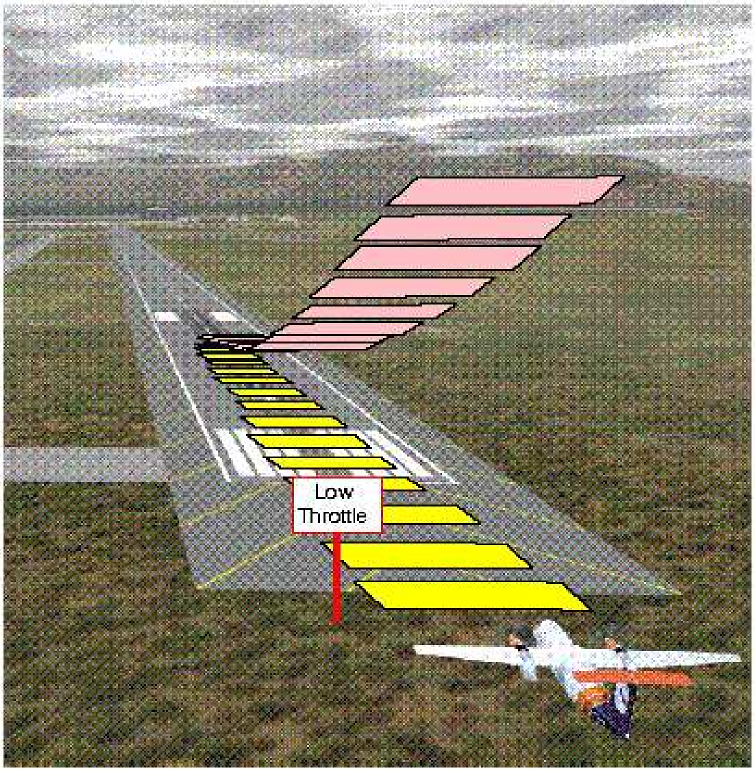

Figure 1: Highway in the sky.

skills should be matched by a corresponding shift from hierarchical decision tree models (or sometimes production rule models) of student behavior to subsymbolic models. This is why we created a gated network of neural net "experts" to accurately model the performance of novice pilots.

3.2 Modular Neural Networks

Learning a model of a complex, dynamic process is closely related to the problem of learning to control such a process. In this section, we describe the modular network approach to control as it was first defined. Then we describe how we extended the approach to the problem of modeling a student pilot.

For the control of complex, dynamic systems, it is often impossible to define a single feedback controller that works well over all possible operating conditions. Instead, multiple controllers can be defined and activated for different parts of the system's state space. One way to do this is by *gain scheduling*, where multiple controllers are defined by specifying sets of controller parameters, or gains, and assigning their use to different regions of state space [3]. The designer must partition the state space into regions for which the system can be approximated by linear time-invariant models. Individual controllers can then be designed for each region.

Partitioning the state space appropriately requires knowledge of the dynamics of the system to be controlled. The design effort would be drastically reduced if the regions and controllers could be learned from samples of interactions between the controller and the system. Jordan and Jacobs [2] describe such an approach. Their method consists of multiple neural networks, called *expert networks*, each learning different, relatively simple, control laws. An additional network learns when to activate the expert networks, possibly mixing, or *gating*, the responses of the individual networks to obtain an overall response from the collection of networks. Figure 2 show the configuration of a modular neural network consisting of n expert networks and the gating network.

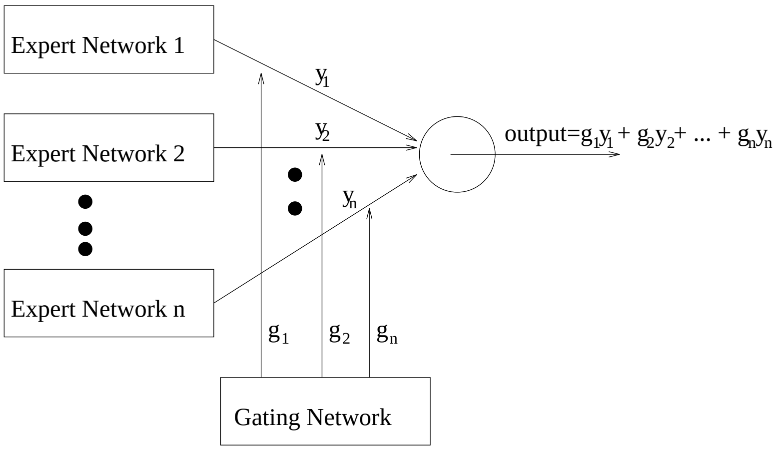

Figure 2: A modular neural network composed of one gating network and n expert networks. The expert networks learn unique, simple models. The gating network mixes the predictions of each expert to obtain an overall prediction.

When complex neural networks are trained to model a set of data, they are very likely to *overfit* the data, much as a high-order polynomial will overfit a small number of data points. The modular network approach is one way to limit the amount of overfitting. Each expert network is restricted to the learning of a model of low complexity. The complexity of each network is limited by the number of computational units contained in the network, their interconnectivity, and their transfer functions. The gating network combines the output of the experts, but again the combination is very simple, usually just a weighted, linear combination of the outputs of the expert networks. The expert and gating networks and the combination function are parameterized by numerical weights, which can be adjusted by any optimization technique. The most common is an extension of the error backpropagation algorithm, which is simply a gradient descent in the overall network's error function.

It is rather surprising that all of the weights in the modular network can be adjusted simultaneously. In fact, by learning all the expert networks and the gating network simultaneously, the problem is automatically decomposed into pieces whose complexity is limited by the complexity of the expert networks. Jacobs and Jordan [2] show that for a problem involving the control of a robot arm whose gripper can hold a variety of payloads, the expert networks learn simple control laws that pertain to particular payload masses. The appropriate expert network is selected by the gating network as a function of the payload identity, which does not indicate the actual mass of the payload. After their modular network was trained for a number of different payloads, they tested the network's ability to control a novel payload. They found that the gating network used a weighted combination of previously-learned expert networks to successfully control the new payload.

Another example is from the work of one of the PI's of this project. Anderson and Hong [1] developed a reinforcement-learning variant of the modular network architecture. They trained their network to keep a simulated inverted pendulum balanced on a cart and away from the bounds of a horizontal track. Analysis of the resulting modular network showed that the expert networks did automatically decompose the task. In most runs, the expert networks became tuned to either positive or negative velocity of the cart. Other decompositions were also found.

For our project, we learned models of student pilots using a modular neural network. In this case, the modular network learned to predict the actions of the students, given as input the current variables that the students can observe. The ability to automatically decompose a problem is very useful in this context. In addition to controlling for overfitting, the resulting decomposition forms a categorization of student behaviors. For example, one expert network might learn to predict student actions on take-offs, while another might learn to predict actions for level flight. This result would be very useful to SymSystems for future intelligent training systems. Fundamental behaviors that are identified would form the basis of the adaptive formation of training regimes for each student.

We expect particular behaviors to be modeled by the expert networks if the networks' complexity is maintained at a fairly low level. If we allow the expert networks to be somewhat more complex, an exciting possibility arises. Simple behaviors would no longer be learned; instead, whole repertoires of behaviors would be modeled by each expert network. Each expert network would become specialized to the nuances in behavior exhibited by classes of students. Perhaps one expert network would become tuned to predicting the actions of skilled pilots, while another would predict actions of novices. If certain students have more trouble with left and right control while others have trouble with up and down control, then we would expect separate expert networks to become tuned to these two classes of students. This highlights the research thrust of this proposal. The modular network architecture will ultimately provide answers to questions related to the learning of flying skills.

Our study of modular networks was constrained to simple feedforward neural networks for the expert and gating networks. One hidden layer was used in each. The number of hidden units in each network can be varied to study the effects of varying the complexity of the networks. The number of expert networks can also be varied.

We began by modeling the behavior of a single student. The accuracy of our model was evaluated by testing on data that was not used to train the modular network. The resulting decomposition was analyzed. A new modular network was then trained and tested on data from two students. The analysis of the resulting decomposition was summarized for SymSystems as a basis for the design of intelligent training systems.

3.3 Data Acquisition and Refinement

A majority of the effort on this project was consumed by the recording of flight data and experimentation with ways of representing the data as input to the neural networks. This section of the report summarizes the data that we collected, the representation problems we faced, and our solutions to these problems. Directories and file names all refer to the organization of the data and code that was transferred to SymSystems.

3.3.1 Data

Data is stored in the `data` directory. There is a subdirectory for each of the six pilots for whom we have collected data. Each flight consists of three files: `Attitude`, `PhaseEtc`, and `XYZ`. Flights are distinguished by sequential "flight numbers", 1, 2, etc., which are used as file extensions. For instance, the `XYZ` data file for Trevor's second flight is `data/trevor/XYZ.2`. Currently available data sets are shown in Table 1.

pilot	flight #s	flight date
dave.p	1, 2	January 15, 1998
chuck	1, 2	March 22, 1998
david	1, 2	March 22, 1998
joe	1, 2	March 22, 1998
trevor	1, 2	March 22, 1998
chuck	3–6	April 8, 1998
bruce	1–6	April 8, 1998

Table 1: Available data sets.

All of the data files are ASCII files. Each file has two header lines; the top header line gives field names. Field values are comma (and space) separated. Here are the first three lines of a data file (an Attitude file, after comma delimiters have been stripped):

```
  Common_time   Msn_elap_msec          Pitch           Roll            Yaw       PitchVel        RollVel         YawVel
999999999999999999999999999999999999.9999999999999.9999999999999.9999999999999.9999999999999.9999999999999.99999
      53681056            1056        0.00090        0.00000      -90.02499        0.05467        0.00000        0.00000
```

The data recorded from the flight simulator is organized into three data structures: states, actions, and pavers. Each is a two-dimensional matrix, of sizes T x 16, T x 4, and P x 3, respectively, where T is the number of time steps in the flight and P is the number of pavers in the flight plan. The details of the matrices' fields are explained in Table 2. Additional details of some of the fields follows.

data struct	field #	field name	field symbol	field description	ref frame	range	units
states	1	phase		phase of flight		0–10	
	2:4	PosX, PosY, PosZ	(x, y, z)	plane position	world		meters
	5:7	VelX, VelY, VelZ	$(\dot{x}, \dot{y}, \dot{z})$	plane velocity	plane		feet/sec
	8:10	AccelX, AccelY, AccelZ	$(\ddot{x}, \ddot{y}, \ddot{z})$	plane acceleration	plane		feet/sec^2
	11:13	Pitch, Roll, Yaw	$(\theta_{\text{pitch}}, \theta_{\text{roll}}, \theta_{\text{yaw}})$	plane orientation	world		degrees
	14:16	PitchVel, RollVel, YawVel	$(\dot{\theta}_{\text{pitch}}, \dot{\theta}_{\text{roll}}, \dot{\theta}_{\text{yaw}})$	plane rotation	world		deg/sec
actions	1	SLPC		throttle		0–255	
	2	stickX		surface deflection: aileron		-18–18	degrees
	3	stickY		surface deflection: elevator		-15–24	degrees
	4	elevTab		surface deflection: elev trim		-24.5–19	degrees
pavers	1:3	pavers	(px, py, pz)	pavers position	world		meters
common_time	1	common_time		sampling time stamps			msec

Table 2: Data fields as recorded from flight simulator.

phase - Determined by waypoint "intersections". Waypoints are x, y, z points in world coordinates. Intersections are defined by 500 meter-diameter columns. Values correspond to waypoint signs, as listed below. Phases designated as 5 ("none") should be replaced by the last non-5 waypoints. For example, if the sequence of phases is [3 5 5], this is equivalent to [3 3 3].

number	phase
0	final approach
1	initial approach
2	climb
3	cruise
4	descent
5	<none>
6	takeoff-decision
7	takeoff-rotate
8	taxi
9	taxi and hold
10	touchdown

Table 3: Flight phases.

PosX, PosY - The convention used is a left-handed coordinate system, in world reference frame.

PosZ - Negative PosZ values indicate that the altitude is with respect to ground, positive PosZ values are with respect to sea level. All of our data sets use PosZ with respect to ground.

VelX, VelY, VelZ, AccelX, AccelY, AccelZ - Reference frame designation as "plane" is a slight misnomer; think of starting with a coordinate system based on the plane, but stationary. VelX is velocity in the direction of the plane. Think of AccelZ as against gravity. Thus, since VelZ is the integral of AccelZ, VelZ as is almost always positive; only serious drops make it negative.

Pitch, Roll, Yaw - The convention used is a left-handed coordinate system, such that

- positive X is out the nose of the plane
- positive Y is out the right wing of the plane
- positive Z is up out of the plane

SLPC - Single Lever Power Control. Value ranges correspond to types of flight and to the SLPC label on the head-up display, as in Table 4. We can extend these ranges so that they are contiguous,

range	flight type
0-10	idle
11-29	descent
71-79	approach
167-179	cruise
207-215	climb
246-255	takeoff

Table 4: SLPC ranges and the corresponding flight types.

such as declaring climb to be 207-245.

stickX - Negative is left, positive is right.

stickY - Negative is up (i.e. pull backward), positive is down (i.e. push forward).

elevTab - Negative is down, positive is up.

To gain a more intuitive understanding of the data, all values recorded during one flight can be plotted with respect to sample number. The position, velocity, and acceleration for one flight are shown in Figures 3, 4, and 5, respectively. The orientation and change in orientation are shown in Figures 6

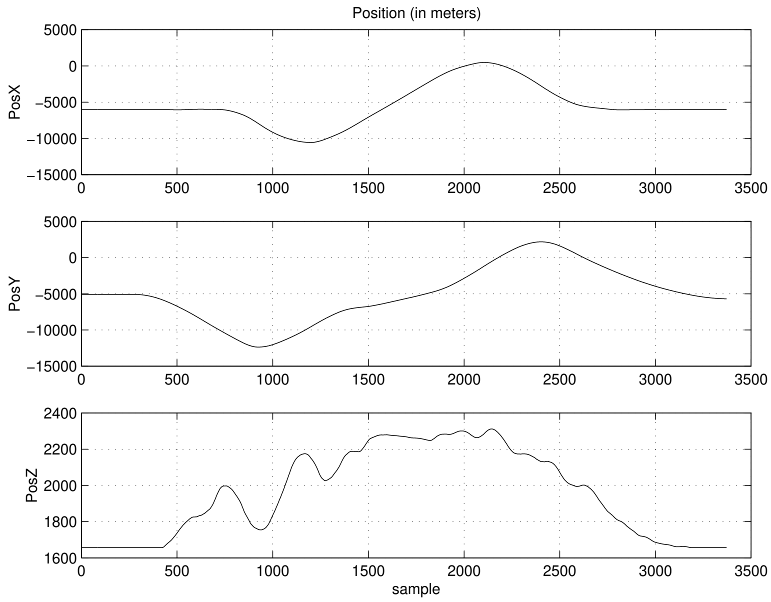

Figure 3: Position in x, y, and z versus sample number.

and 7. The actions taken by the pilot during this flight are plotted in Figures 8, 9, 10, and 11. The plane's position and orientation during a flight is much easier to visualize by superimposing a 3D picture of the plane at multiple points during the flight. Including the pavers allows a quick visual check to see how accurately the pilot followed the pavers. Figure 12 shows this for one flight.

3.3.2 Data Problems

The sampling rate of the simulation system varied. Sample intervals were either 66, 99, 132, or 165 ms. Note that they all differ by exactly 33 ms. Curiously, the 132 ms interval (which would roughly correspond to the 7.5 Hz sampling rate we specified) was the least common. The mean of the sampling interval is about 107 ms. There appears to be a distinct pattern in the time intervals as a function of sample step. However, the pattern is slightly inconsistent: the 132-ms intervals do not occur in a regular pattern. Figure 13 shows how the sampling interval varied during one flight. The figure also shows a frequency histogram of the sampling interval.

This problem was addressed by simply assuming that the data was actually recorded at a constant sampling rate. Checks of the recorded data did not find any examples of jumps in data values corresponding to variations in the sampling interval. In fact, data is much smoother when plotted with respect to a constant sampling interval in comparison to plots with respect to the recorded sampling interval.

However, we did observe that for the data we recorded on January 15th, the data values remain constant for two samples every fifth or sixth sample. Curiously, these "flat" intervals usually occur during the second consecutive 99 ms sampling interval, as shown in Figure 15. This problem was addressed by deleting the second of such samples from all data fields.

A related problem was discovered in data recorded after January 15th. In this data, there are time steps when all of the action fields are frozen, but not all of the state fields. It is possible that the action

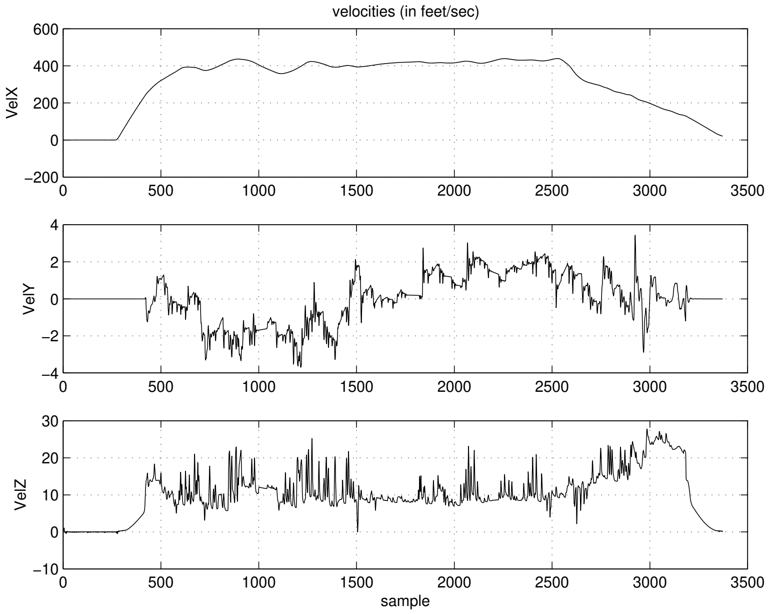

Figure 4: Velocity in x, y, and z versus sample number.

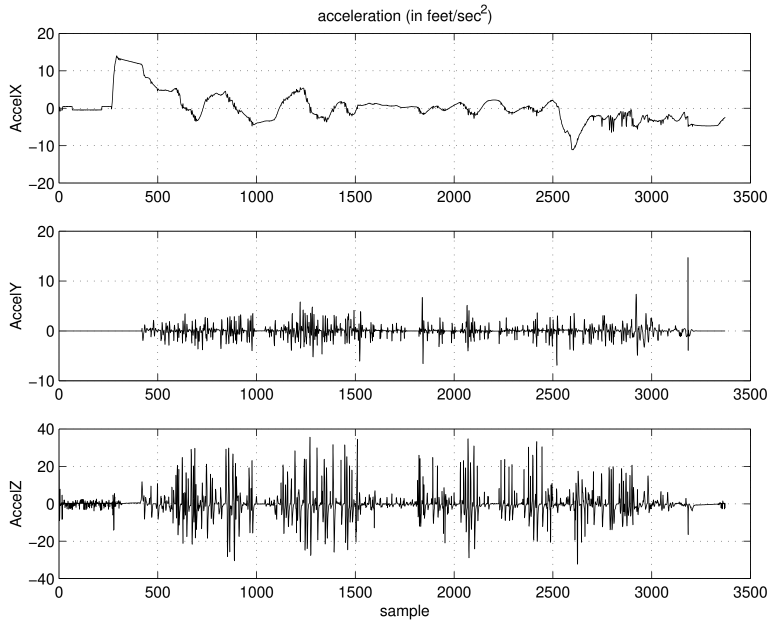

Figure 5: Acceleration in x, y, and z versus sample number.

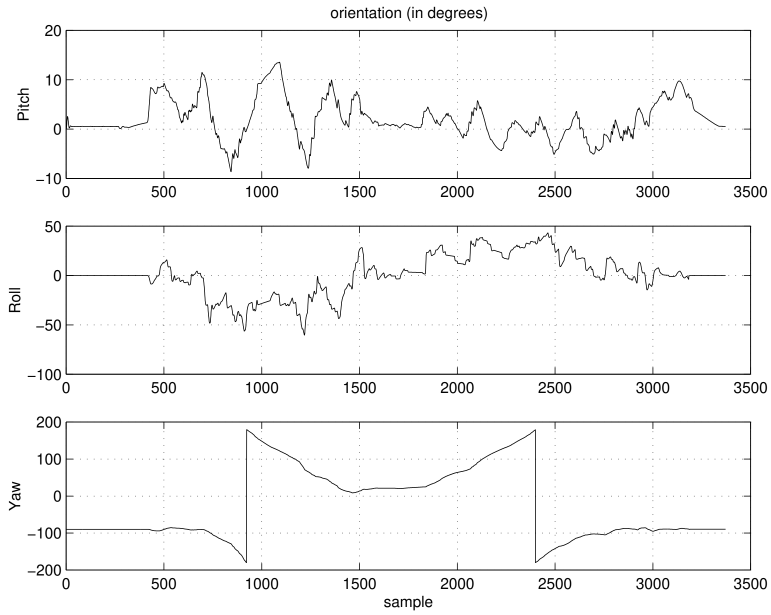

Figure 6: Pitch, roll, and yaw versus sample number.

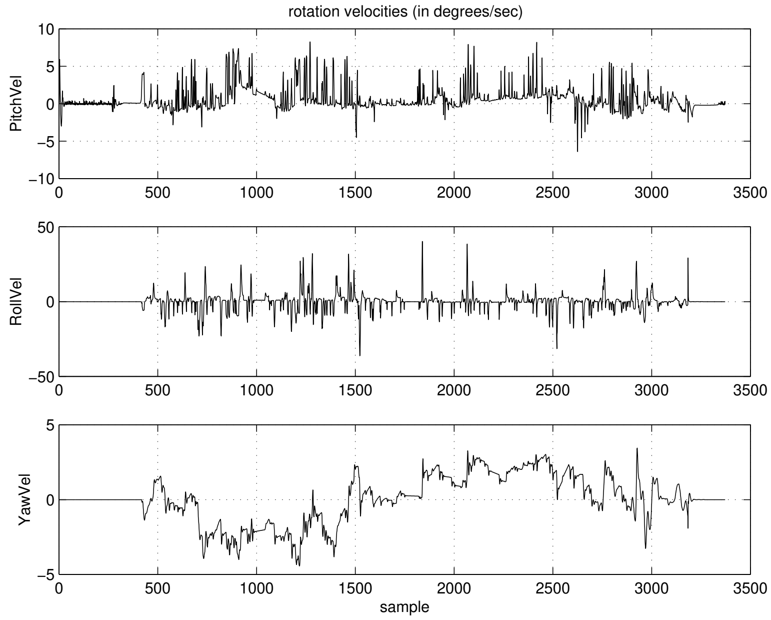

Figure 7: Change in pitch, roll, and yaw versus sample number.

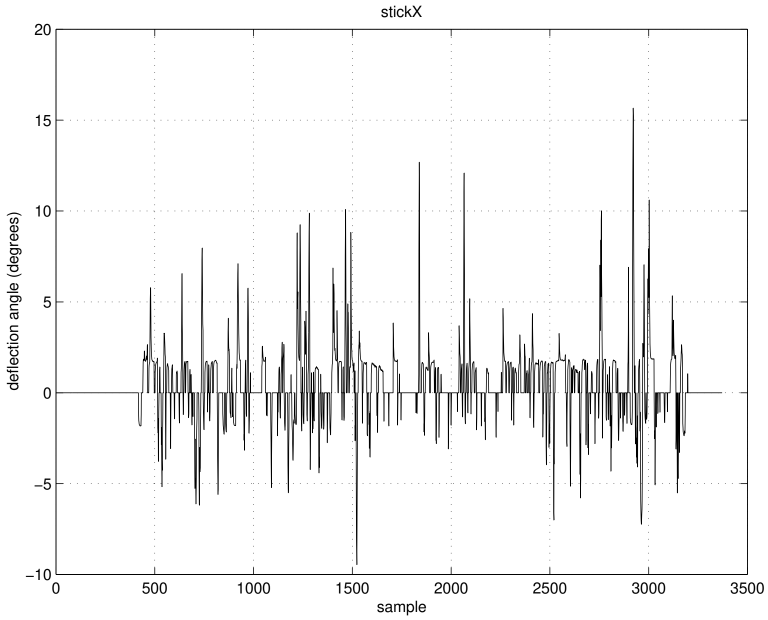

Figure 8: StickX versus sample number.

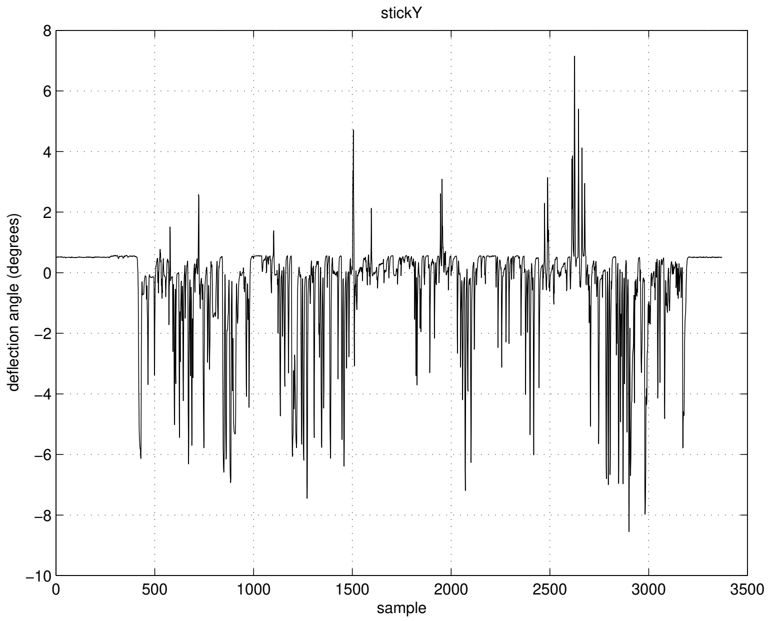

Figure 9: StickY versus sample number.

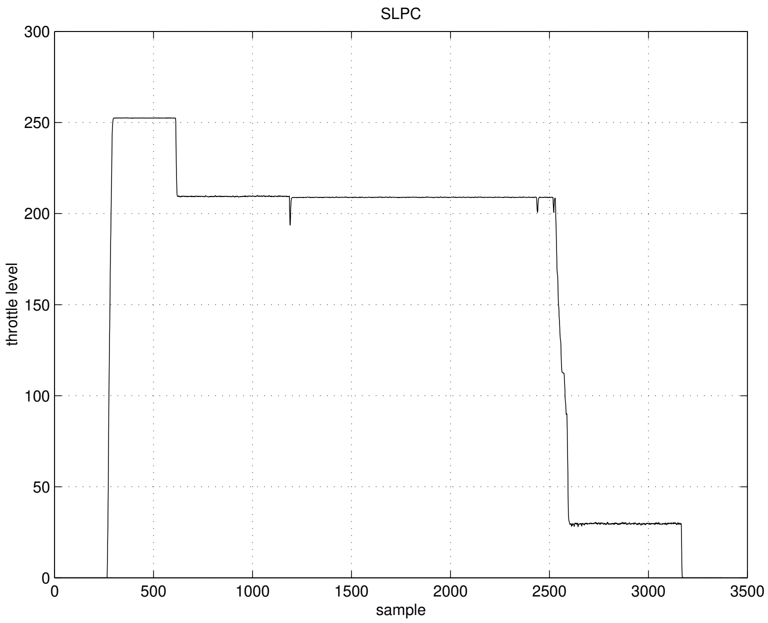

Figure 10: SLPC (throttle) versus sample number.

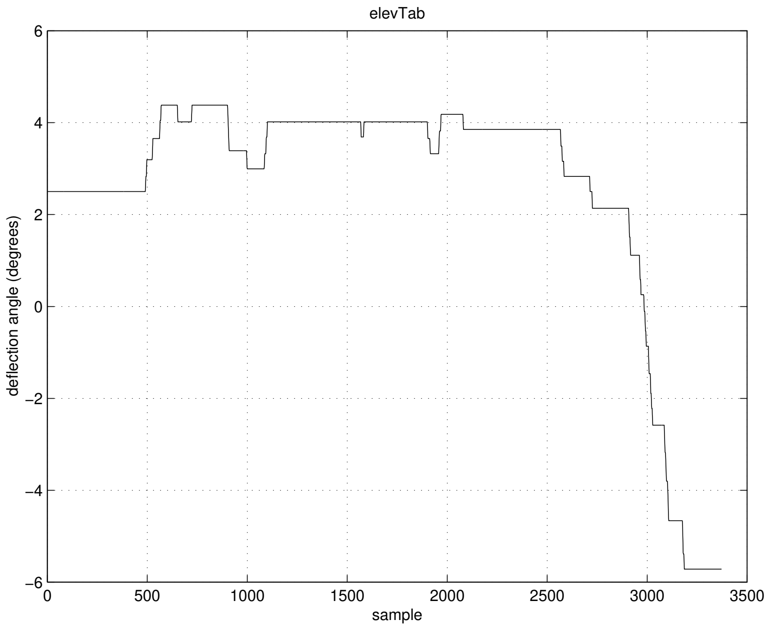

Figure 11: Elevator tab versus sample number.

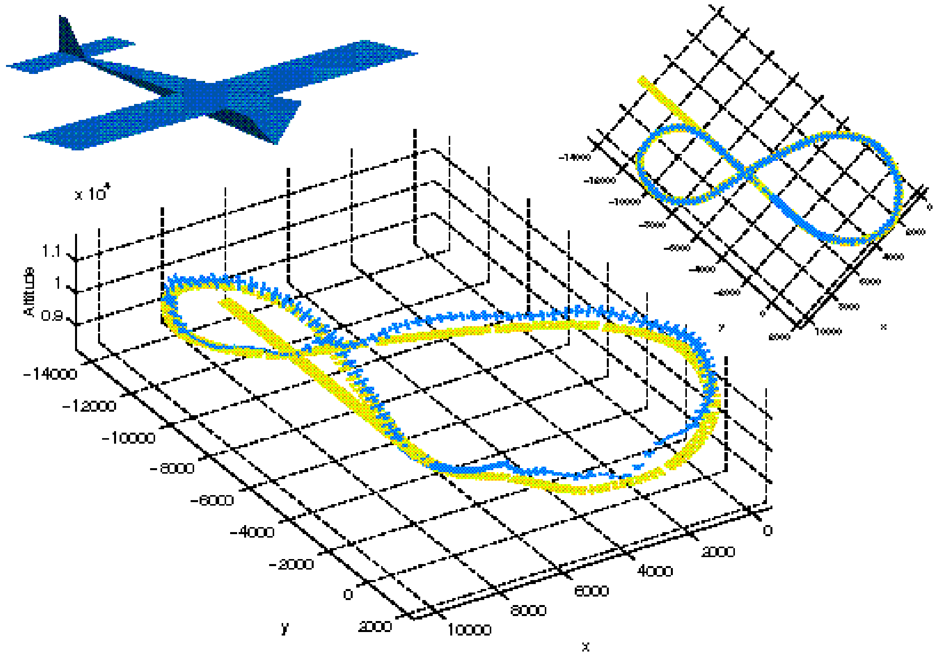

Figure 12: One flight shown as plane's position and orientation at a subset of the sample points. Pavers are also shown.

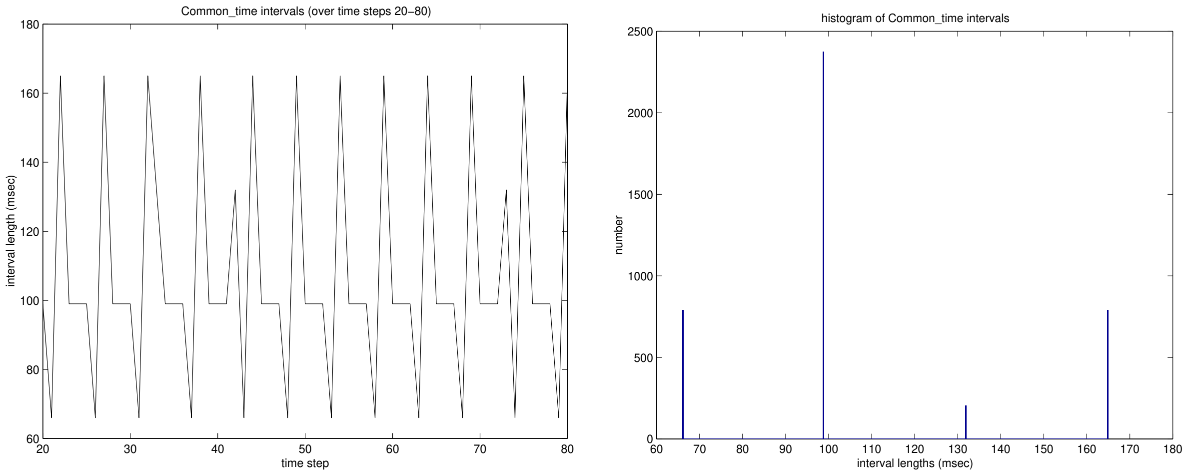

Figure 13: Sampling intervals from one flight. One the left is the sampling interval versus the sample number for the first 80 samples. On the right is a frequency histogram of all sampling intervals for one flight.

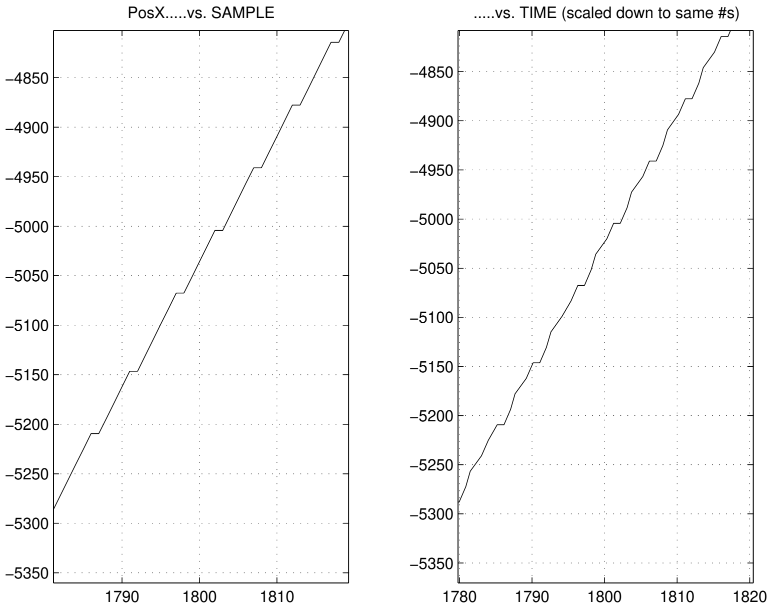

Figure 14: PosX plotted with respect to sample number and with respect to recorded time.

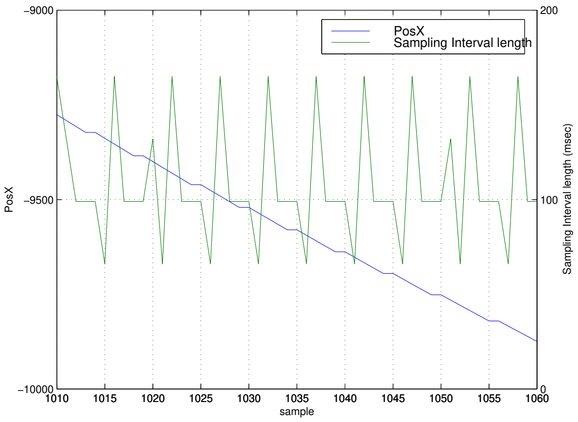

Figure 15: Constant value of PosX correspond to 99 ms sampling intervals.

fields would remain constant for two consecutive samples. Therefore, for these flights, we only deleted those time steps when *all* of *both* the states and actions fields are constant.

Another possible problem was found with the SLPC and stickY fields. They exhibit what appears to be some kind of artifactual noise during relatively constant periods, as shown in Figure 16. Although the noise magnitude is very small, it appears to be occurring, especially in the case of SLPC, during periods when the value should be absolutely constant. Through consultation with SymSystems, this was found

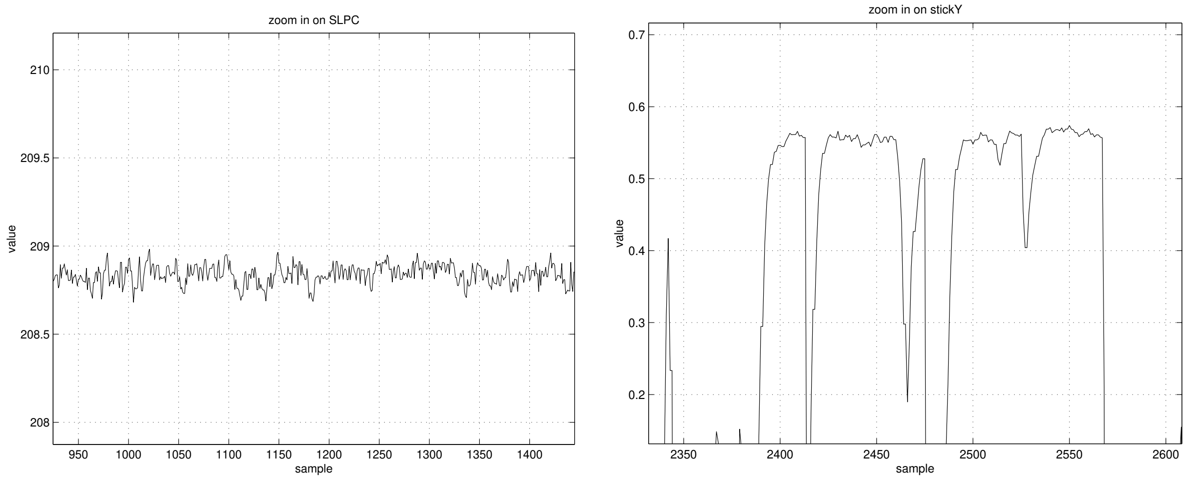

Figure 16: Zoomed in view of SLPC and StickY showing noise in the sampled values.

to be due to noise from the analog input devices. We decided that the noise is of such low magnitude that the averaging properties of neural networks will handle this.

Several fields, including VelZ, AccelX, AccelZ, Pitch, and PitchVel, exhibit some strange artifacts prior to takeoff. In the case of PitchVel, the computed derivative is equal to the measured value. In the case of AccelX and AccelZ, the computed derivative analogues do not show the artifacts, i.e., the computed derivative does not equal the measured value. A representative example is shown for AccelX in Figure 17. SymSystems personnel informed us that the algorithms used when on the ground at

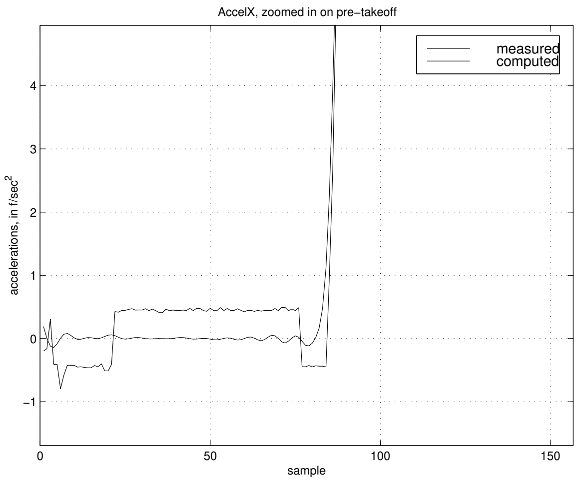

Figure 17: Data variations prior to takeoff.

low velocities are different than when at higher velocities and in the air. This is due to many factors, primarily ground effects modeling. The variations are of such low magnitude, they were ignored.

3.4 Predicting Actions Given Current State and Next Pavers

To model a student pilot, a modular neural network must be provided with inputs that are similar to what the pilot experiences as they make their decisions about how to move the stick, throttle, and elevator. Pilots look ahead to the pavers in the "highway" to decide which direction and how quickly they must turn the plane. There are many ways to model this perception; we chose a simple method that works quite well. The locations of the 5th, 10th, and the 20th next pavers are input to the neural network, in terms of position relative to the plane. A diagram of these pavers is shown in Figure 18.

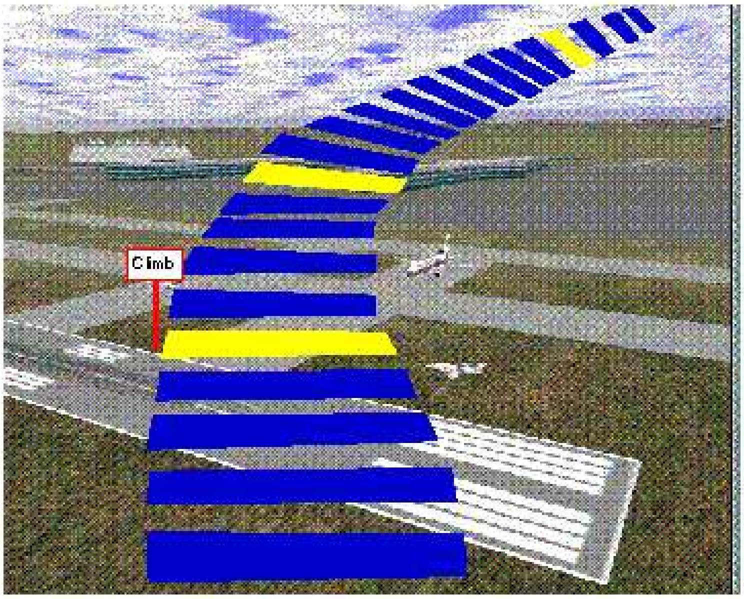

Figure 18: Pavers included as input to neural network are shown in yellow.

These paver positions, along with the plane's position, velocity, acceleration, orientation, and rotational velocity, are provided as input to the network. The output of the network is the prediction of the action values of stick x and y, throttle, and elevator tab positions. But how many steps ahead should we train the network to predict these actions? To explore this question, we trained networks to predict actions at a number of time steps ahead. The results are shown in Figure 19. There is a large increase in error as the number of steps ahead is increased from 1 to 10, beyond which there is much less of an increase. Therefore, we chose to use 10 steps ahead. Figure 20 summarizes the inputs and outputs of the neural network.

4 Results

After much preliminary experimentation, we arrived at the following results. We trained a modular neural network on Bruce's flights 1, 3, and 6 and Chuck's 1, 3, and 6. The modular network had five expert nets, each having two hidden units. The gating network had five hidden units. The input to all networks was (PosX, PosY, PosZ, VelX, VelY, VelZ, AccelX, AccelY, AccelZ, Pitch, Roll, Yaw, PitchVel, RollVel, YawVel) and the x, y, z coordinates, relative to the plane, of the 5th, 10th, and 20th pavers in

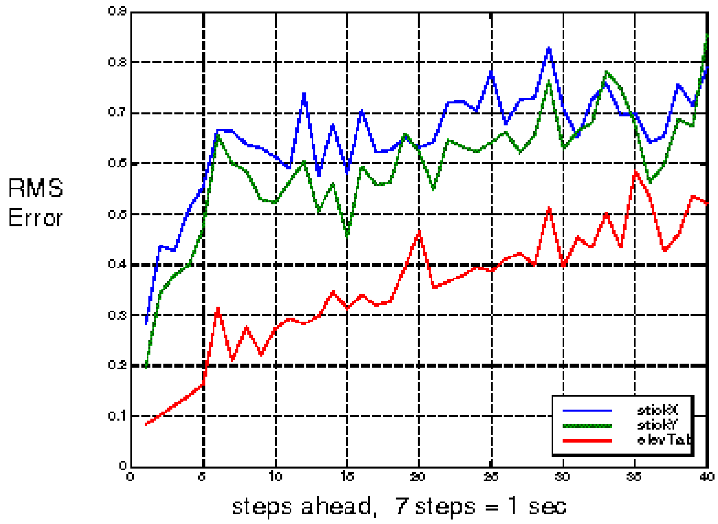

Figure 19: RMS error in predicted actions for different number of steps ahead.

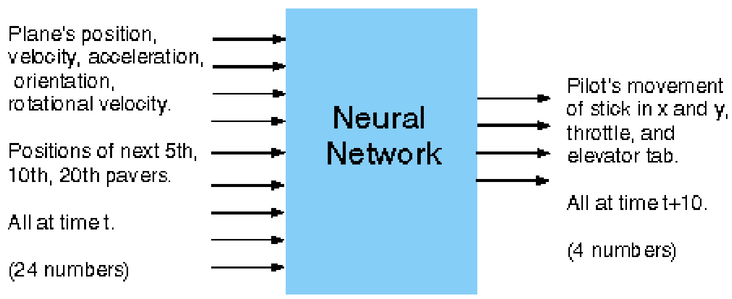

Figure 20: Neural network inputs and outputs.

front of the plane. The modular network was trained to match the pilot's stick x, y, and elevator tab responses in 10 steps, given the current input values. Samples are acquired at 7.5 Hz, so the network is predicting the pilot's response approximately 1.5 seconds ahead. The learning rate for hidden units is 1, and for output units is 0.1. No momentum was used. Initial weights ranged between -0.1 and 0.1.

Figures 21 and 22 show the actual and predicted actions for Bruce's flights 1 and 6. Note that the general trends of the actions are fairly well predicted, but the large, fast changes are not matched very well. This may be due either to a deficiency in the network's input or in the network itself. The input is deficient if the input variables do not adequately represent all of the information known by the pilot. Our prediction of actions may not be based on the whole context available to the pilot. The neural network would be deficient if too few expert networks are used, or if each expert network contains too few hidden units.

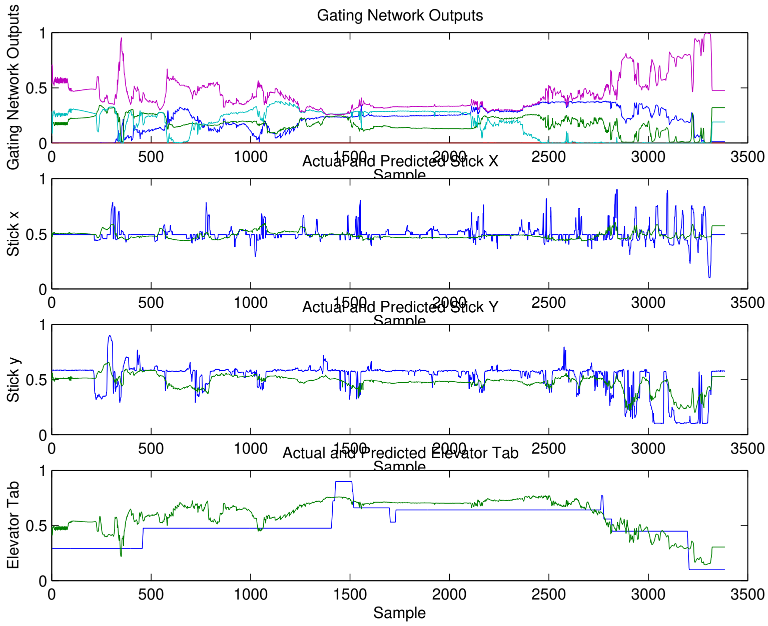

Figure 21: Bruce Flight 1, all samples.

The first graphs in Figures 21 and 22 show the outputs of the gating network. We can see that one output, the purple one, is largest for most of the samples. This shows that a single expert network was able to model most of the decisions of the student pilot, particularly for the take-off and landing phases of the flight. All but one expert, the red one, are having an effect on the predictions made by the modular network.

Figures 23 and 24 show the results for Chuck's flights 1 and 6. These graphs show that Chuck's actions are somewhat easier to predict. To aid comparisons between different flights and different subjects, results for flights 1, 3 and 6 for Bruce and Chuck are plotted consecutively in Figure 25.

It is difficult to judge from these graphs which expert net is best at predicting actions for any sample. A much easier way is to again draw a 3D view of the plane's position and orientation at various samples. However, this time we draw the plane using a color corresponding to which gating network output is highest, and thus which expert network is best at predicting the actions. This view is shown for two of Bruce's flights in Figure 26. One of Chuck's flights is shown in Figure 27. The colors do not correspond with the colors in the previous graphs. We see that the blue expert provides the best prediction for most samples. However, there are situations where other experts are best. For this flight, these situations seem to be ones where the pilot must dive back to the pavers, or turn right or left to return to the pavers.

Most samples for all of our pilots can be predicted by a single expert network, as shown in the

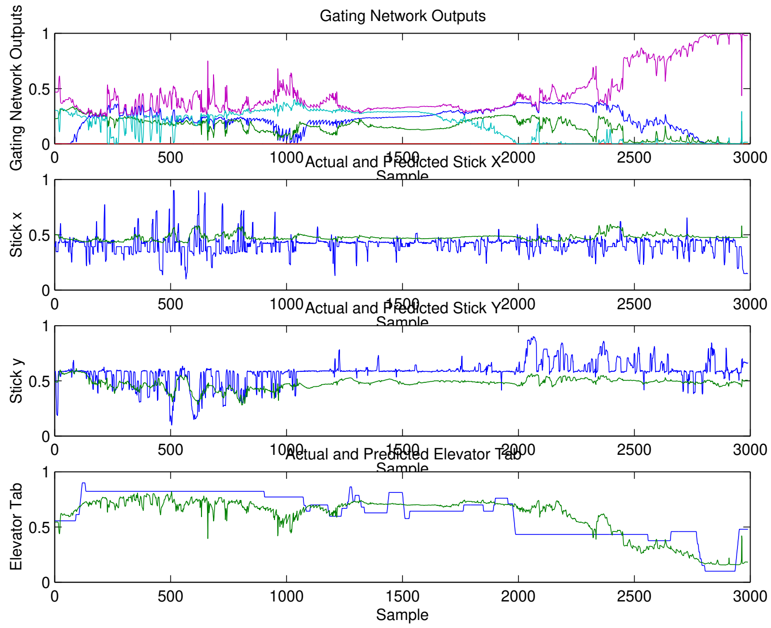

Figure 22: Bruce Flight 6, all samples.

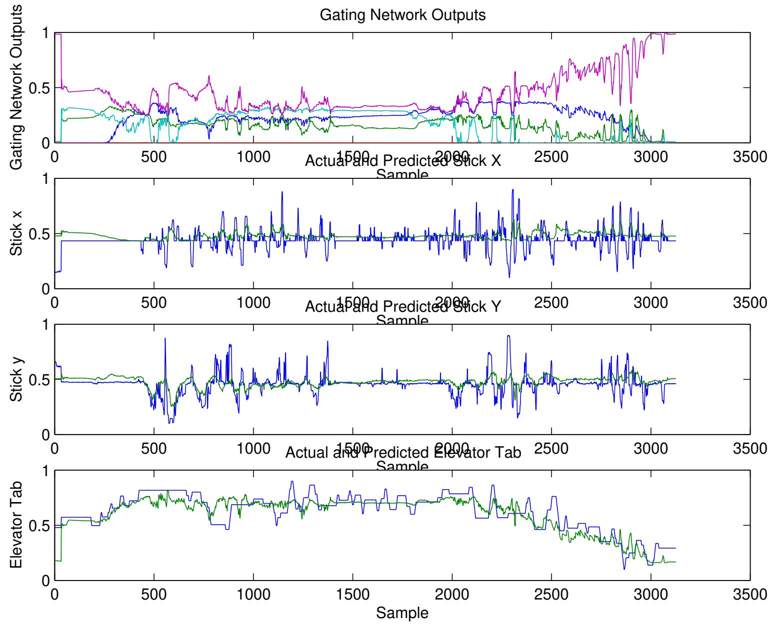

Figure 23: Chuck Flight 1, all samples.

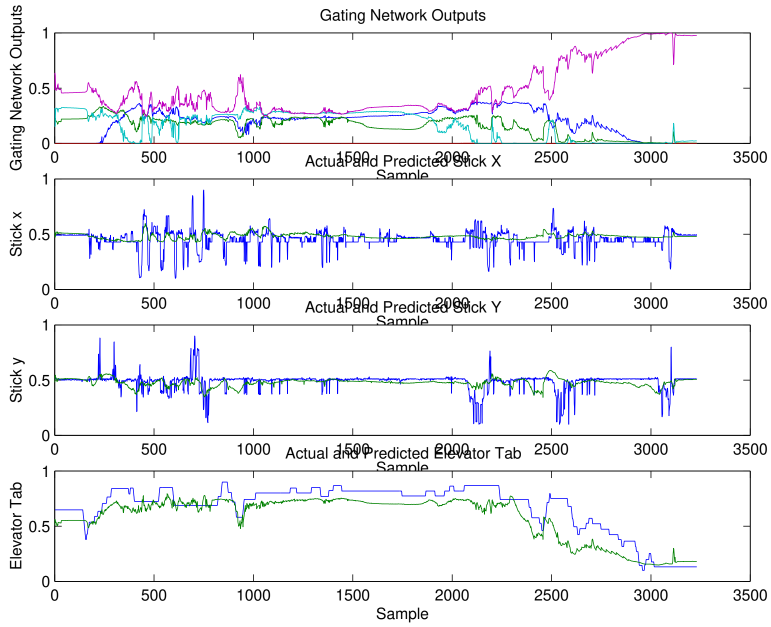

Figure 24: Chuck Flight 6, all samples.

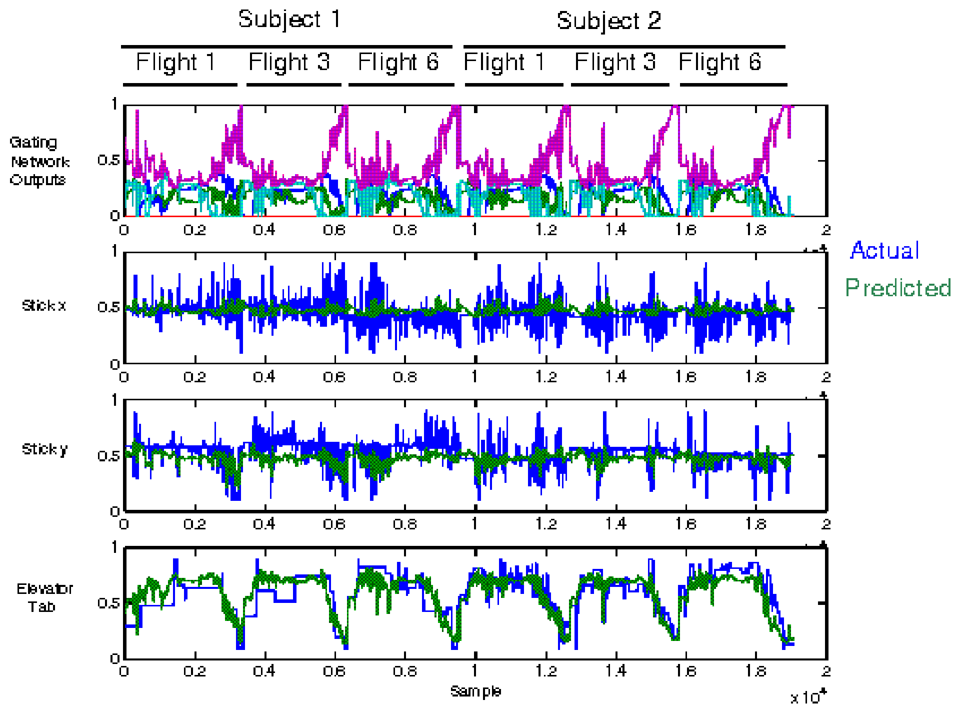

Figure 25: Actual and predicted pilot actions, and output of gating network for two subjects, three flights from each.

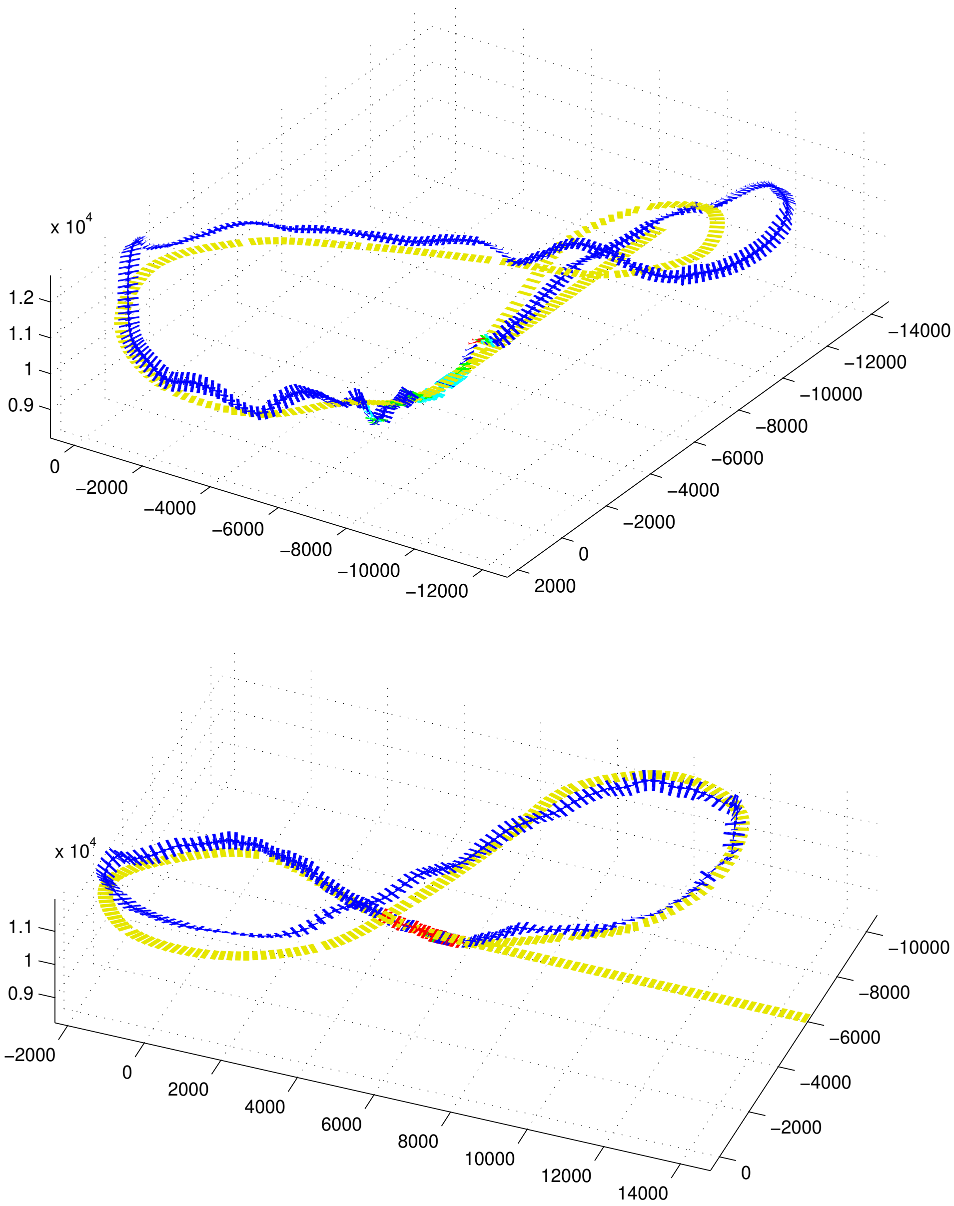

Figure 26: Bruce's first and sixth flights. Color of plane shows which expert network most accurately predicts actions.

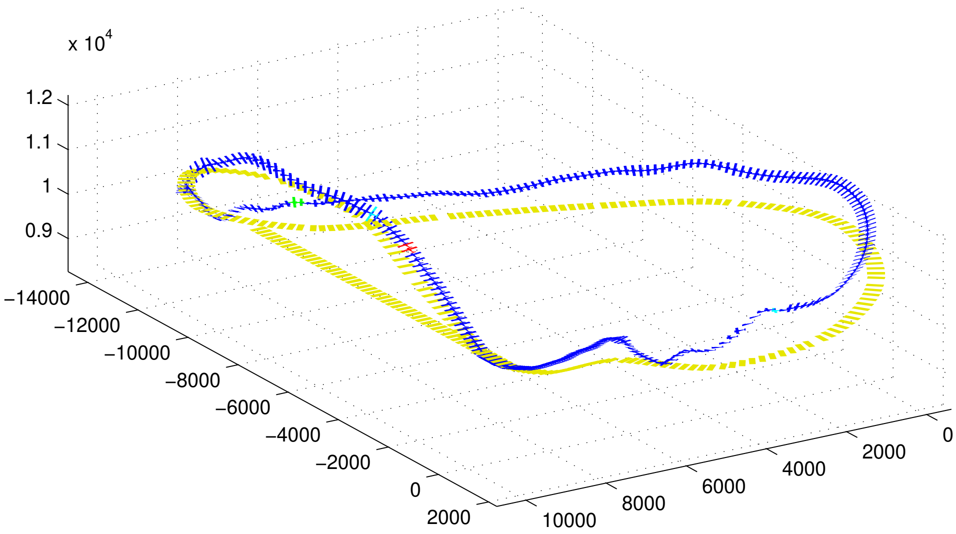

Figure 27: This is for one of Chuck's flights.

frequency histogram in Figure 28. This corresponds to samples where flying is relatively easy. The pilot just makes minor adjustments to follow the pavers. Occasionally the pilot will drift too far from the pavers and take drastic action to return. It is these skills that students must develop to become better pilots. It is also these skills that the students seldom practice. Therefore, we focused our analysis on the samples that are not best predicted by the usual expert.

By ignoring the samples best predicted by expert 5, we could look for differences in the distributions of best experts between subjects while each subject learns. These results are shown in Figure 29. This figure shows clearly that different students learn differently. Expert 1 appears to best predict situations that only occur in flight 2 for Bruce, and flight 3 for Chuck. Expert 2 predicts a skill that Chuck seems to have from the first flight, but that Bruce develops only after his third flight.

These initial results prompt more questions than answers. Data for more subjects and more flights are clearly needed in order to draw valid conclusions. However, the results we have obtained so far suggest that our approach may have a tremendous payoff for intelligent training. A system that adapts to the learner's acquired skills could greatly reduce training time by focusing on experiences that are predicted to be troublesome for the student.

5 Evaluation

Evaluation of our system was in two forms. First, the error between the predicted pilot's actions and the actual actions was calculated. Our neural network approach was able to predict the pilot's actions quite well.

The second form of evaluation is more important for the success of this project. Our objective was to determine whether or not the modular network approach would distinguish different skills learned by student pilots. The number of subjects in our study was limited by the amount of time available on the simulator. In spite of this, we were able to find distinct differences in strategies learned by different students.

Thus, the project is deemed a success. Results show that this approach to intelligent training is definitely an approach that should be explored.

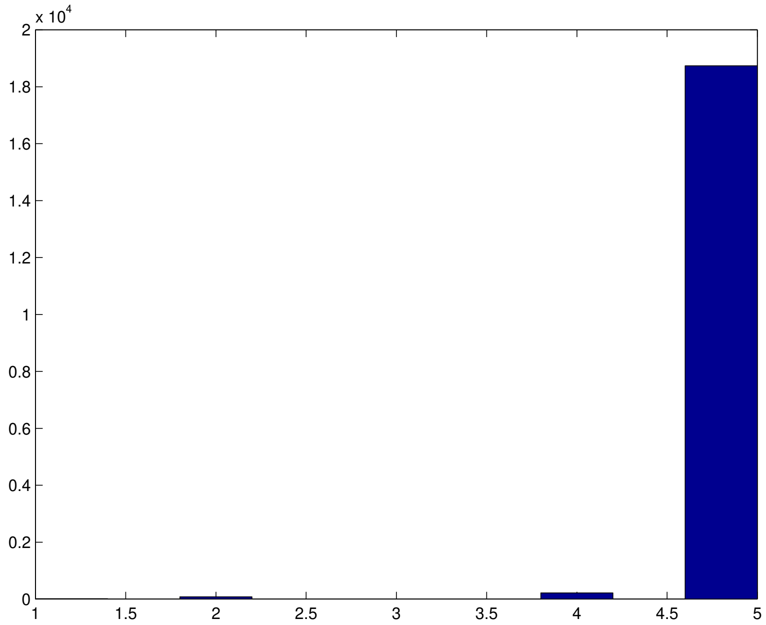

Figure 28: Histogram showing counts of samples in which expert networks 1 through 5 best model the pilots. Expert 5 is best for almost 20,000 samples, much more than any other expert. Expert 4 is best for about 500 samples.

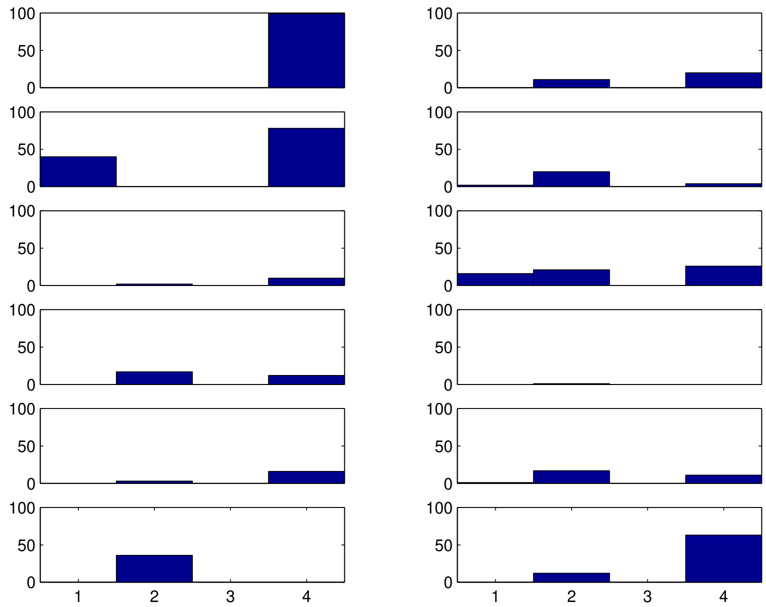

Figure 29: Histograms for flights 1 through 6 for Bruce are shown on the left. Flights 1 through 6 for Chuck are shown on the right.

6 Technology Transfer

The most critical resource for this project was data about student performance in FTDs. This data was supplied by SymSystems, which used its Mission Simulator System (MSS) to collect data from novice pilots. MSS is a real-time, reconfigurable, human-in-the-loop simulator for commercial and military aircraft. Developed in 1988, MSS is in use today as the core component of realistic flight training devices and as a prototyping tool used to explore aircraft, cockpit, C3I, and avionics system designs. For this project, SymSystems used MSS on the SymSystems AGATE concept demonstrator cockpit. The demonstrator consists of a fiberglass aircraft cockpit with flight control stick and single lever power control. It uses two SGI workstations and a 21-inch monitor to display the out the window view and head up displays.

Data was collected using the MSS data collection software, which collects simulation data and asynchronous events, triggers events, performs mission replay, and post processes the simulation data. The collected data was transferred to CSU for off-line analysis to compare the models predicted with measured behavior. In addition to this project, the data was also used by students in C. Anderson's (one of the co-PIs) graduate class in artificial intelligence. One student in that class used common feedforward networks to model a subset of the data.

Transfer to the company consisted of several meetings at the SymSystems site to discuss our modular neural network approach, and the reporting of our results to SymSystems staff. All code and data are available to SymSystems to continue with this direction of research.

7 Networking

C. Anderson contacted C. Sammut at the University of New South Wales, in Sydney, Australia, to discuss Sammut's work in modeling student pilots. Sammut's work involves the construction of symbolic models, while our work involves numerical, or sub-symbolic, models. Sammut is very interested in our approach. As a result, Anderson has applied for a sabbatical at the University of New South Wales to collaborate with Sammut and continue the work begun on this CASI project.

No conferences were attended where this work was presented.

8 Publications

No journal publications have been submitted on this project. A web site is being developed that will include our CASI final report, slides from our CASI talk, and details of our approach, including some of the code that was used.

9 Follow-on Funding

No follow-on funding has yet been obtained. A proposal is in preparation that will be submitted to the National Science Foundation, either to the Learning and Intelligent Systems section of the Knowledge and Distributed Intelligence Program, or to the Knowledge and Cognitive Systems program within CISE. During Anderson's sabbatical in Australia, joint funding with C. Sammut will be pursued.

References

[1] Anderson, C.W., and Hong, Z. "Reinforcement Learning with Modular Neural Networks for Control", *Proceedings of NNACIP'94, the IEEE International Workshop on Neural Networks Applied to Control and Image Processing*, 1994, pp. 90–93.

[2] R.A. Jacobs and M.I. Jordan, "Learning Piecewise Control Strategies in a Modular Neural Network Architecture", *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2), pp. 337–345, March/April, 1993.

[3] J.S. Shamma and M. Athans, "Analysis of Gain Scheduled Control for Nonlinear Plants", *IEEE Transactions on Automatic Control*, 35, pp. 898–907, 1990.

[4] D. Sleeman and J.S. Brown (eds), *Intelligent Tutoring Systems,* Academic Press, London, 1982.

[5] M. Yazdani. "Intelligent Tutoring Systems Survey", *Artificial Intelligence Review,* Vol 1, page 43, 1986.

[6] B. Woolf. "Intelligent Multimedia Computer Tutors", *Communications of the ACM,* 34(4), April 1996, 30-31.

[7] Conference on Intelligent Computer-Aided Training. Houston, TX: Johnson Space Center. 1991.

A Intellectual Property Disclosure Form