Aircraft Position Prediction Using Neural Networks

by

Anuja Doshi

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

January 28, 2005 [February ...]

Author _____
Department of Electrical Engineering and Computer Science
January 28, 2005

Certified by _____
Rafael Palacios
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Aircraft Position Prediction Using Neural Networks

by

Anuja Doshi

Submitted to the
Department of Electrical Engineering and Computer Science

January 28, 2005

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The Federal Aviation Administration (FAA) has been investigating early warning accident prevention systems in an effort to prevent runway collisions. One system in place is the Airport Movement Area Safety System (AMASS), developed under contract with the FAA. AMASS uses a linear prediction system to predict the position of an aircraft 5 to 30 seconds in the future. The system sounds an alarm to warn air traffic controllers if it foresees a potential accident. However, research done at MIT and Volpe National Transportation Systems Center has shown that neural networks more accurately predict the future position of aircraft. Neural networks are self-learning, and the time required for the optimization of safety logic will be minimized using neural networks. More accurate predictions of aircraft position will deliver earlier warnings to air traffic controllers while reducing the number of nuisance alerts. There are many factors to consider in designing an aircraft position prediction neural network, including history length, types of inputs and outputs, and applicable training data. This document chronicles the design, training, performance, and analysis of a position prediction neural network, and the presents the resulting optimal neural network for the AMASS System. Additionally, the neural network prediction model is then compared other prediction models, including a constant speed, linear regression, and an auto regression model. In this analysis, neural networks present themselves as a superior model for aircraft position prediction.

Thesis Supervisor: Rafael Palacios

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Rafael Palacios. From start to finish, he has provided me with guidance, encouragement, and support. His motivations are always noble, and his unrelenting pursuit of the best prediction system has brought our project to where it is today. Although he is working overseas at the Universidad Pontificia Comillas, Dr. Palacios has graciously agreed to supervise my thesis at MIT. His technical knowledge is an inspiration to me, and he is always ready to answer my questions, however trivial they might be. So, thank you, Dr. Palacios. I couldn't have done it without you.

In addition, I want to thank the other students I had the pleasure of working alongside: Yoshi Nakanishi, Geoff Cooney, and Gap Thirathon. And for the confidence and input of Brent Midwood, and Vince Orlando, our sponsors from The Volpe National Transportation Systems Center. I would also like to thank Dr. Amar Gupta, the project supervisor, who gave me the opportunity to work on this project. Lastly, I would like to thank my parents who provided me with education and drive to get to where I am today.

1. INTRODUCTION

According to the Federal Aviation Administration (FAA), from 1999 through 2002, there have been 1,480 runway incursions in major US airports [10]. That is approximately six runway incursions for every one million operations. To increase airport safety, the FAA has been investigating early warning accident prevention systems. These systems are designed to provide air traffic controllers, who are responsible for traffic management on runways and taxiways, with enough early warning to intervene prior to a potential or impending accident. One system in place is the Airport Movement Area Safety System (AMASS), developed under contract with the FAA. AMASS utilizes prediction models to predict separation distances between aircraft, and sounds an alarm if it foresees a potential accident. Currently, AMASS is being used in the nation's 34 busiest airports. Although the AMASS system performs relatively well, research has shown that incorporating neural networks into AMASS will provide better airplane position predictions, thereby providing earlier alerts and more accurate alarms. The neural network enhanced system began development under a collaboration of the Volpe National Transportation Systems Center and PROFIT Initiative, headed by Dr. Amar Gupta of the MIT Sloan School of Management.

The evolution of the neural network prediction system has occurred in two main phases. In the first phase, chronicled in Chapters 3 and 4, a single neural network was developed to predict future airplane coordinates. In creating a neural network, one is faced with many design decisions. Instead of speculating the best choices for the new network, I simulated many neural network variations in MATLAB, and used the results to construct the most effective position prediction neural network. The neural network was optimized by investigating three variables: history length, types of inputs and outputs, and training data proportions. The history length is the number of seconds of previous airplane position used as an input to the neural network. The inputs and outputs to the neural network can be plane coordinate positions, velocity vectors, or some combination of the two. The training data is extremely large, and it can be broken up into the following sets: high speed data, taxi data, and stop data. In order to minimize training time and maximize performance, different proportions of these datasets can be used to train the neural net.

Unfortunately, the "optimal" neural network did not perform as well as the team expected it to. The network was very sensitive to imperfections in input data. After much investigation, it became apparent that two neural networks were needed. One for high speed predictions, and another for low speed predictions. Furthermore, we needed to more closely examine how AMASS computes incursion distance. This work is chronicled in Phase II, Chapters 5 and 6.

Chapter 7 compares the accuracy of the neural network prediction system to other prediction models. We compared the neural network model to the AMASS prediction system, a constant speed model, and an auto-regressive model. In addition to predictive analysis, we looked closely at the alarms generated by the AMASS system, and the neural network enhanced system. The neural network predictive model produced fewer false alarms, and earlier warning in real hazardous situations.

2. BACKGROUND

Background information about AMASS and neural networks is presented in this section. This information should help with the understanding of our research and the challenges we faced in using neural networks in order to enhance the AMASS system.

2.1 AMASS

AMASS consists of the Terminal Automation Interface Unit (TAIU), which is built by Dimensions International, and the AMASS subsystem, developed by Norden Systems. TAIU suggests a runway for arriving aircraft after receiving information from the control center radar. TAIU passes runway and airborne aircraft information to the AMASS subsystem. The AMASS subsystem also receives data on ground aircraft from ASDE-3 (Airport Surface Detection Equipment Radar). AMASS analyzes the position and velocity of aircraft in the air and on the ground, as well as vehicles on the runway. If there is a possible conflict, the system alerts air traffic controllers.

The software, written in C++, is composed of three modules.

- The radar acquisition module, the core of TAIU, processes information on the velocity, altitude and position of incoming aircraft.
- The runway prediction module scans the airport and suggests a runway on which the aircraft can land.
- The AMASS subsystem processes the information through safety algorithms and issues alerts if necessary.

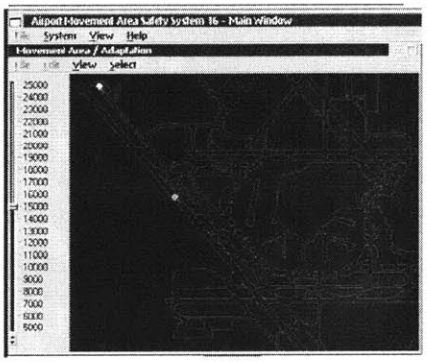

Figure 1. AMASS Screen Shot

Air traffic controllers monitor the Operator Display Unit (Figure 1) and awaits alarms. The controller is notified by text alerts and aural alerts broadcast in the tower. If alerted, the controller takes immediate action to notify the aircraft in danger.

2.2 Neural Networks

Scientists believe that the humans store learned information in a highly interconnected neural network in the brain. When given a set of inputs and outputs, our neurons "learn" the relationship between the inputs and outputs through weights assigned to each input and net of synapse firing thresholds.

Artificial neural networks are modeled after biological nervous systems. In practice, neural networks are especially useful for classification, pattern recognition, and function mapping problems. Neural networks are tolerant of some imprecision and they are most useful in problems where the learning functions are too complicated for algorithmic modeling. Neural networks are also self-learning, so they can be improved anytime new data becomes available. Neural networks are currently used in signal processing, speech recognition, vision learning, and even have applications in financial and medical industries.

The core unit of the neural network is the artificial neuron, also referred to as a perceptron (Figure 2). The perceptron weights its inputs and sums the weighted inputs together. The sum is fed through a transfer function and output to the rest of the network. Thus, if the inputs are x_1, x_2, x_3, and the weights are w_1, w_2, w_3, the output is

$= f(x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3)$, where $f(x)$ is usually the step function or sigmoid function.

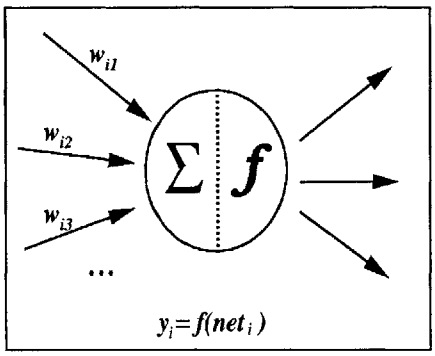

Figure 2. The Perceptron.

Artificial neural networks are a connected network of perceptrons, organized in layers (Figure 3). A neural network can be thought of as a black box; it receives inputs from an outside source, a outputs data to the external environment. Inside the box, there are one or more hidden layers, composed of perceptrons (Figure 2). The network learns by assigning weights to all the inputs contained in the network. The net adjusts the weights when new material is learned.

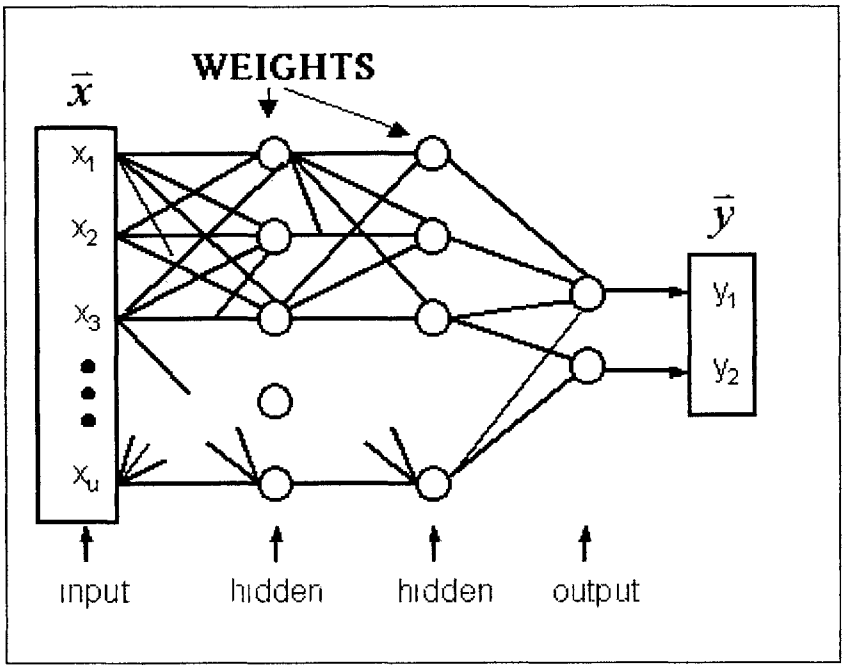

Figure 3. An Artificial Neural Network, composed of perceptrons.

Practical applications of NNs use supervised learning. For supervised learning, the neural network is trained with data that includes both the input and the desired result. After successful training, presenting the input data alone to the NN will result in output values that approximate the desired result. For training to be successful, a lot of training data and computation is necessary. Multilayer networks can approximate any smooth function as long as there are a sufficient amount of hidden nodes. Unfortunately, too many hidden layers can cause the network to learn the noise in the data, which results in an overtrained neural net. Using a validation set while training prevents overtraining the network. As the weights are reassigned, the neural network checks to ensure that the error in the validation set predictions is decreasing. As soon as the error begins to increase, the neural network stops learning to protect against overtraining.

3. DESIGN PHASE I: SINGLE NEURAL NETWORK APPROACH

This chapter chronicles the design decisions that were faced in determining the optimal neural network for aircraft position prediction. Sections 3.1 through 3.3 present the justifications for design decisions made before I joined the project. Sections 3.4 through 3.6 show the factors I had to consider in molding the neural network.

3.1 Separation Distance vs. Position Prediction

In 2003, I focused on developing a position prediction neural network (PPNN), using the separation distance neural network (SDNN) as a guide. This section explains the design decision to switch to a position prediction neural network.

3.1.1 Separation Distance Neural Network

As of January 2003, a neural network had been developed and integrated into AMASS to predict the separation distance between two planes. The neural network used 5 seconds of history and its prediction results are for 10 seconds in the future:

	Neural Network enhanced AMASS	Original Version of AMASS	% Improvement
Mean Absolute Error	64.41 ft	235.66 ft	72.67%
Mean Square Error	130.48 ft	419.32 ft	68.88%

Table 1. Separation Prediction Neural Network trained with one week of ORD data.

Clearly, the NN enhanced AMASS performed much better than the original version if evaluated with all available data. Unfortunately, the neural network enhanced system did not trigger an alarm in real hazardous situations because it was unable to predict very small separation distances since it had rarely been trained data from hazardous situations. That was the motivating factor to switch to a position prediction neural network. Other weaknesses of the separation neural network surfaced when we investigated different NN prediction models.

Weaknesses of the separation distance prediction neural network:

- **Alarm Analysis**:

  When the predicted separation distance between two planes falls below a certain value, AMASS generates an alarm. However, the SDNN network rarely predicts a separation distance in this range. This is because the data used to train the neural network has so few cases of small separation distances. In the real world, aircraft avoid these situations, so they do not show up in the training data. Our research shows that SDNNs trained with more data generates fewer alarms, and it does not sound an alarm on real alarm situations, including ORD 08/21/2001, 02:31:15. Since the motivation of our research is to produce earlier and more accurate alarms, this setback was our basis for switching neural network models.

- **Limited length of each training event**:

  Since the SDNN only monitored planes after AMASS starts monitoring them, it didn't have long periods of separation distance data. AMASS only monitors planes when it gets in the range of the runway. We have approximately 30 seconds for each event. For example, for a 30 second event, using 5 seconds of history means that we only 25 seconds of data can be used in our Output training set (Figure 4). Thus, there isn't much data for the SDNN to learn how to predict separation distance 30 seconds. In addition, we don't have the option of using a longer history input because that would result in even less data to train the future predictions.

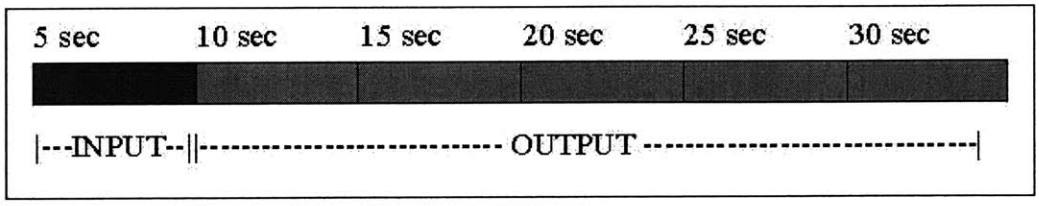

Figure 4. Breakdown of 30 seconds of training data.

3.1.2 Position Prediction Neural Network

The team decided that predicting individual airplane positions and then computing SD based on the current position plus predictions would be more beneficial to our goal of

earlier and more accurate alarms. The advantages and challenges of this design are profiled in this section.

Strengths of the position prediction neural network (PPNN):

- **Alarm Analysis:**

  Since the PPNN learns only the flight pattern of a single plane, it is irrelevant that there are so few alarm situations in real life. Later, the incursion distance can be calculate from the aircraft coordinates. The coordinate information will be taken from the position predictions neural network. If the calculated incursion distance falls below a certain level, or if it is negative, an alarm will be generated. Even though negative separation distances don't exist in real life, if one is calculated, it's a warning of an impending collision.

- **Large number of training events:**

  The PPNN uses training data from all tracks in the raw FAA logs, not just the planes that AMASS monitors. Since there is data for all planes in the logs, the number of training events is huge. All events are monitored: departure, landing, taxi, and stopped planes.

- **Long individual training events:**

  The logs monitor each plane for an extended period of time. Some events last up to 10 minutes, providing more data for each event. Since the events are longer, there is more data to teach the PPNN how to predict aircraft position 30 seconds in the future. And since the length of the event is not a factor, we can choose a longer input history size without reducing the available output training data.

Challenges of the position prediction neural network:

- **Too much data:**

  Since there is so much data, the training data needs to be filtered before the PPNN can be trained. If the training data is not filtered, the training takes too long (days) to finish. A lot of the longer events are stopped planes, and the neural network might not need 10 minutes of stopped plane data to predict that a stopped plane does not move. There are many ways to filter the training data, and care needs to be taken to filter the data in way such that the neural network learns enough about each aircraft state.

3.2 Number of Neural Networks

In the SDNN, the inputs were the separation distance between the planes for the past 5 seconds. Thus, there were 5 inputs. In the PPNN, both the x and y coordinates are inputs and outputs of the neural network. We needed to determine whether to have two distinct neural networks, one to calculate the x coordinate output, and one to calculate the y coordinate output. Or whether to use one neural networks with both x and y inputs and both x and y outputs.

2.3.1 Two Neural Networks

Two neural networks are ideal if the x-coordinate prediction is independent from the y-coordinate prediction. However, the system must separate x and y information and make two calls to the neural network module. Using two neural networks ensures a much shorter training time.

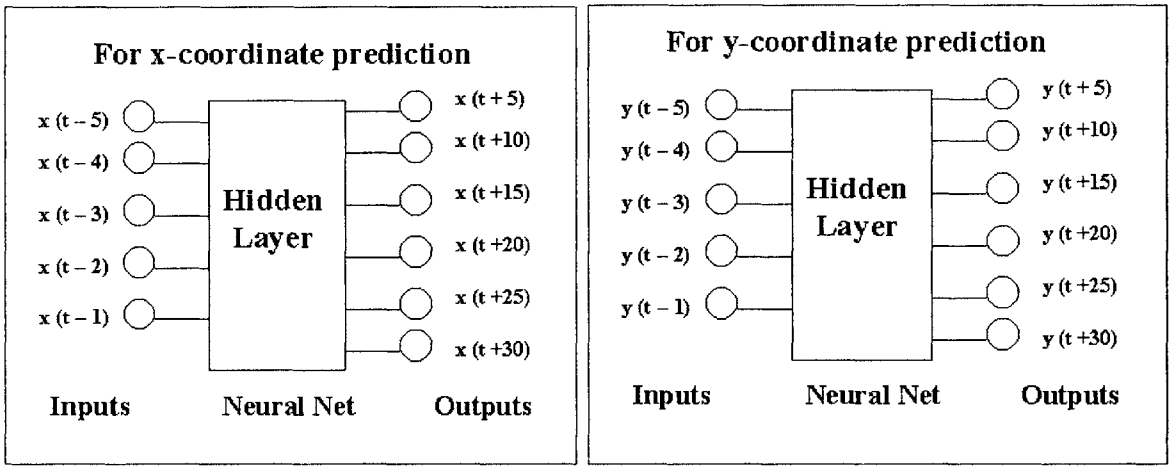

Figure 5. Two independent neural networks to predict x-position and y-position.

3.2.2 One Combined Neural Network

Using one combined neural network is another option. If the x and y coordinates are not independent, the neural network will adjust the weights in such a way that x inputs have incidence on y estimation. It is practical to use a combined neural network if objects are restricted to move in a known path. Thus it would be useful in predicting train movement, but not airplane movement. In addition, using one combined neural network has the disadvantage of a longer training time.

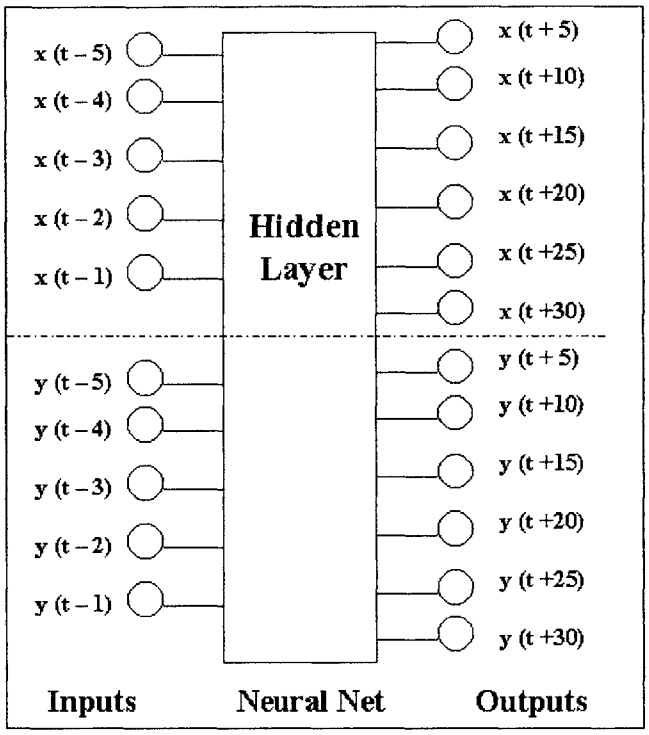

Figure 6. One Neural Network to predict both x and y coordinates.

3.3 Training Algorithm and Neural Network Type

The team had to decide which training algorithm to use to train the new neural networks. We considered both resilient back propagation (trainrp), and the Levenberg-Marquardt algorithm (trainlm). Resilient back propagation is a simple batch mode training algorithm with fast convergence and minimal storage requirements. The Levenberg-Marquardt algorithm is the fastest training algorithm for networks of moderate size. Since our training data set is extremely large, we decided to use resilient back propagation, which was the training algorithm used to train the separation distance neural network. Comparative studies of different types of NN and different training algorithms were performed in order to determine NN type. The group determined that a single layer neural network with 10 hidden nodes was the most appropriate network for our needs [20].

3.4 Length of History

The length of event history affects predictions in the following way. A long history is good for reducing noise while a short history allows for earlier alerts After training the neural network with a longer history, we can look at the weights assigned to the earliest data points. If the weights are very small on the earliest history, it is unnecessary to include

them. To determine how much history to use, we decided to train PPNN with both 5 seconds of history and 10 seconds of history. The performance of both NNs were analyzed to determine which network to move ahead with.

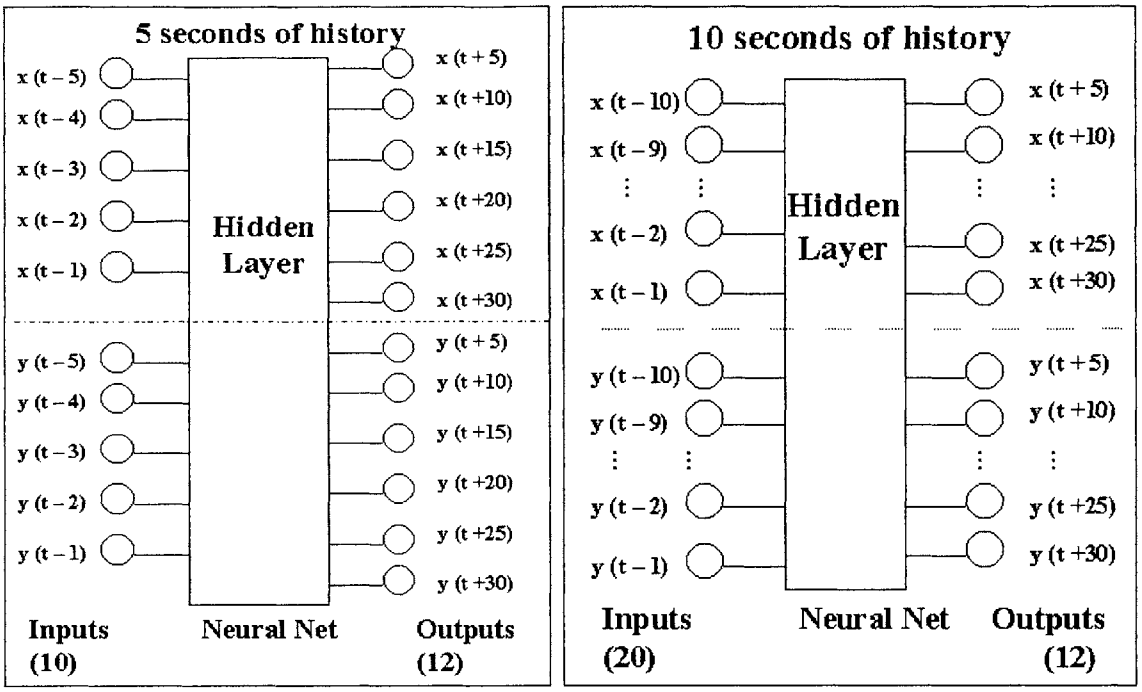

Figure 7. Visual comparison of neural networks with 5 seconds and 10 seconds of input.

3.5 Inputs and Outputs

The team considered various types of inputs and outputs to the neural network. We focused on three different options. All three options were carefully analyzed to determine which approach was optimal.

Input (x, y) → *Output (x, y)*

Having x and y coordinate inputs and outputs will teach the neural network how planes behave based on the coordinates of the airport. For example, the PPNN might learn that a small section of the airport is a holding place for stopped planes and another path is for taxiing planes. An advantage to this approach is that the neural network is being trained with more information. A disadvantage is that the network might be overtrained. For example, if an airplane lands on a runway in the opposite direction than usual, the PPNN might not be able to predict it's movement if it has only been trained with planes landing in the opposite direction.

Input (vx, vy, x, y) → Output (x, y)

This option uses four inputs for each second of history, and the output remains as the x-coordinate and y-coordinate. *vx* stands for the x-velocity vector, and *vy* stands for the y-velocity vector. These inputs come from the raw log file, but the team is unsure how these velocities are measured / calculated. Theoretically, the neural network should be able to determine the airplane velocities from the plane position. However, using these numbers as explicit inputs might increase the knowledge of the position prediction neural network. Note that this neural network also depends on absolute plane position.

Input (dx / dy) → Output (dx / dy)

This option uses only calculated velocities to determine future airplane velocities. The input velocities are calculated from absolute coordinates from the airplane history. For example:

$$dx(t-3) = \frac{\text{distance}}{\text{time}} = \frac{x(t-3) - x(t-5)}{2 \text{ seconds}}$$

This neural network has only 5 inputs [dx(t – 4) … dx(t)], and 6 outputs [dx(t + 5)…dx(t + 30)]. The input can be either dx values or dy values depending on the desired output. The advantage of this neural network is that the same neural network can be used to predict dx or dy. Another advantage of this neural network is the absolute orientation and position of the plane are not factored into the predictions, hence avoiding overtraining and enhancing generalization power. If an airplane lands on a runway opposite the customary direction, the PPNN will be able to predict its future positions just as accurately as if the airplane lands in the standard way. If given the initial position of an aircraft, calculating the absolute future position predictions is straightforward using the outputs of this neural network.

3.6 Training Data Filtering

With such an extensive training dataset, filtering the data is a necessity. In addition, filtering the training dataset is necessary to avoid overtraining. The data can be filtered according to the type of event. The following events are classified by velocity.

Types of events:

Event	Threshold (ft/sec)
Stop	0 to 2
Taxi	2 to 80
High Speed inl. Departure / Landing	80 +

Table 2. Classification of aircraft Events.

The three options we are investigating are:

- **Case 1**: 100% of Stop data, 100% of Taxi data, and 100% of High Speed data.
- **Case 2**: 0% of Stop data, 50% of Taxi data, and 100% of High Speed data
- **Case 3**: Training dataset is evenly balanced with High Speed and Taxi cases.
  (Training data: 50% Taxi and 50% High Speed)

Note the subtlety in between Case 2 and Case 3.

It is important to give the neural network training examples of different cases which it will predict. However, reducing the training dataset might not adversely affect the PPNN performance if the neural network is not learning from additional examples of the same event. One goal of the filtering is to make the training dataset as small as possible without reducing the performance of the NN. And more importantly, we wish to obtain a good balance of taxi , high speed, and low speed events to teach the network information about landing and departures on all runways.

4. TRAINING AND TESTING THE SYSTEM

This chapter presents the findings of the investigation of the following neural network parameters: history length, neural network inputs, and training data. Section 4.4 presents the optimal neural network resulting from the analysis. Section 4.5 compares the predictive capability of the "optimal" neural network with the current prediction system implemented in AMASS.

4.1 Testing Protocol

The performance of each neural network was tested in two ways.

- Each available training set was broken down into 50% Training Data, 25% Validation Data, and 25% Testing Data. After each neural network was created, it was tested with its Testing Data to evaluate it's performance.
- To objectively compare different neural networks, the neural networks were all tested with the same testing dataset. The data used to test the NNs was extracted from the log file of a different day (ORD 08/20/02) than the training data (ORD 08/21/02). This section presents the results of this objective comparison because it was this comparison that determined which neural network to move forward with.

The following neural network was used as the control variable to compare the results of the different networks.

- 5 seconds of input history
- Input (x, y) → Output (x, y)
- 0% of Stop / 50% of Taxi / 100% of High Speed Training Data available

The following table shows the performance of the Control neural network when tested with the test dataset. Two error indicators had been computed: the mean absolute error (MAE), and normalized mean square error (MSE). The former is computed as the average of the absolute value of the difference between real position and estimated position. The later is computed as the square root of the average of the squares of the differences. The square root is computed in order to obtain rational units, so that both indicators are expressed in feet.

Table 3. The AE and MSE for the control NN for each prediction.

NN #1	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	38.5123	214.4988
	10	61.1302	214.5783
	15	118.3901	345.6276
	20	177.801	477.9115
	25	238.2453	601.4243
	30	326.6187	798.0976
Y coordinate predictions	5	17.0812	78.1279
	10	52.6802	156.4426
	15	75.7189	245.5349
	20	120.0383	370.8078
	25	166.9513	485.7636
	30	222.4006	588.8976

4.2 Length of History

The following table presents the results of NN #2, which takes an input of 10 seconds of past history for both the x and y coordinates. These results will be compared to the control neural network, which takes an input of 5 seconds of past history. Both were tested with the same data set, ORD 8/20/02.

Table 4. Performance of NN with 10 seconds of input data.

NN #2	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	20.3658	130.9486
	10	51.9801	171.0901
	15	104.4055	272.278
	20	150.686	387.0589
	25	217.1947	507.6121
	30	290.2496	670.8289
Y coordinate predictions	5	22.6604	98.2866
	10	45.7165	143.7931
	15	76.3565	223.3593
	20	118.3267	308.7598
	25	169.9173	419.2577
	30	232.5621	543.7197

The neural network with 10 seconds of history input performed better than the neural network with 5 seconds of history input. The average difference in absolute error is approximately 10 feet, and the average difference of MSE is 50 feet. These numbers, although small, make an important difference in a safety critical system such as AMASS.

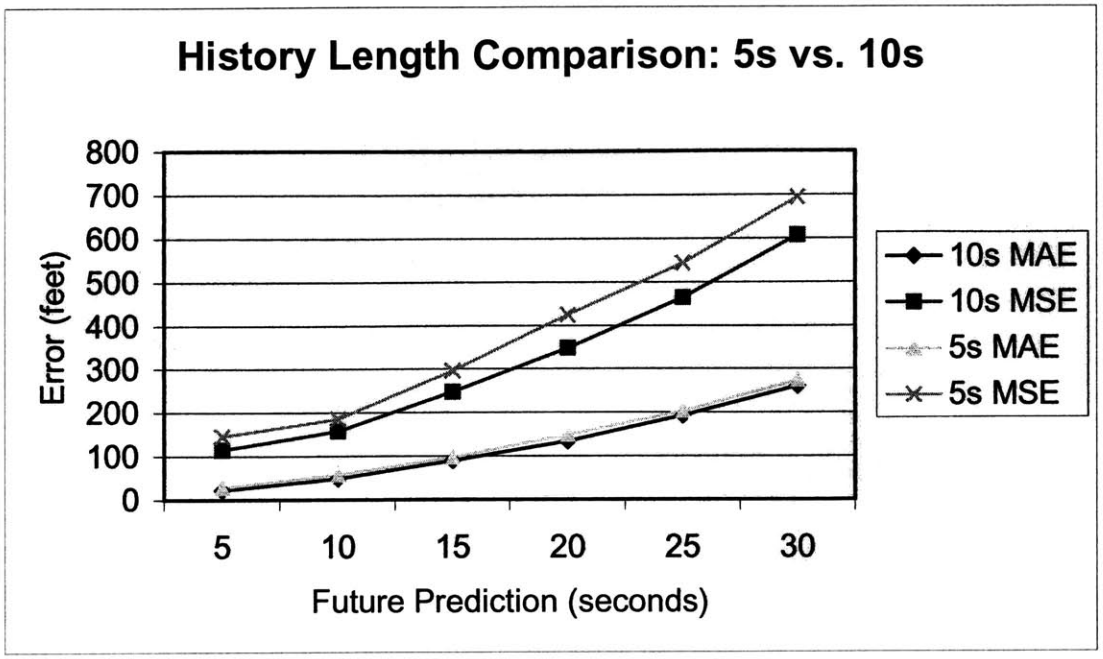

Figure 8. A comparison of the NNs with 5 seconds and 10 seconds of history input.

This experiment shows that future airplane position is dependent on more than the previous 5 seconds of position. Although training a neural network with 10 seconds of past history is more time and computationally intensive, the trade–off in performance is worth the extra time in training.

4.3 Inputs and Outputs

This section presents the results of neural networks with different types of inputs and outputs. Potential inputs include position coordinates, velocity vectors, or coordinates along with velocity vectors. The following table is the performance of a neural network which takes an input of (x, y, vx, vy) for each second of history, and outputs of (x, y) for each prediction interval.

Table 5. Performance of NN with x, y, vx, and vy as input data.

NN #3	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	28.6249	94.7369
	10	60.1002	126.6303
	15	105.9637	195.2329
	20	165.1381	288.5431
	25	234.8033	394.6505
	30	314.1055	533.6045
Y coordinate predictions	5	25.0464	80.5171
	10	54.6794	105.5863
	15	96.2944	192.5369
	20	146.7566	237.5608
	25	203.5444	331.9708
	30	275.1589	450.6343

Table 6, below, presents the results of NN #4, which takes an input of either dx or dy, and outputs dx or dy for each prediction interval. Note that this neural network has half the number of inputs as the control neural network (Table 3), and a fourth of the inputs of NN #3 (Table 5).

Table 6. Performance of NN with dx or dy as input and output data.

NN #4	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	23.0143	69.8669
	10	56.3569	119.4121
	15	103.3652	188.5133
	20	161.7185	277.5678
	25	231.1155	392.2792
	30	310.8419	530.2315
Y coordinate predictions	5	20.9678	54.9081
	10	50.9114	97.0091
	15	92.2279	156.4053
	20	143.6874	235.8143
	25	203.6226	334.5951
	30	271.3923	451.4881

The neural network with an input of dx or dy and an output of dx or dy performed best. This neural network is advantageous because it does not need two coordinates of input for

each second of history. The same neural network predicts both the x and y coordinates accurately. Although this neural network outputs velocity values as its predictions, it is easy to convert the velocities into positional predictions. Its position predictions were compared to the output of the test data to determine its performance.

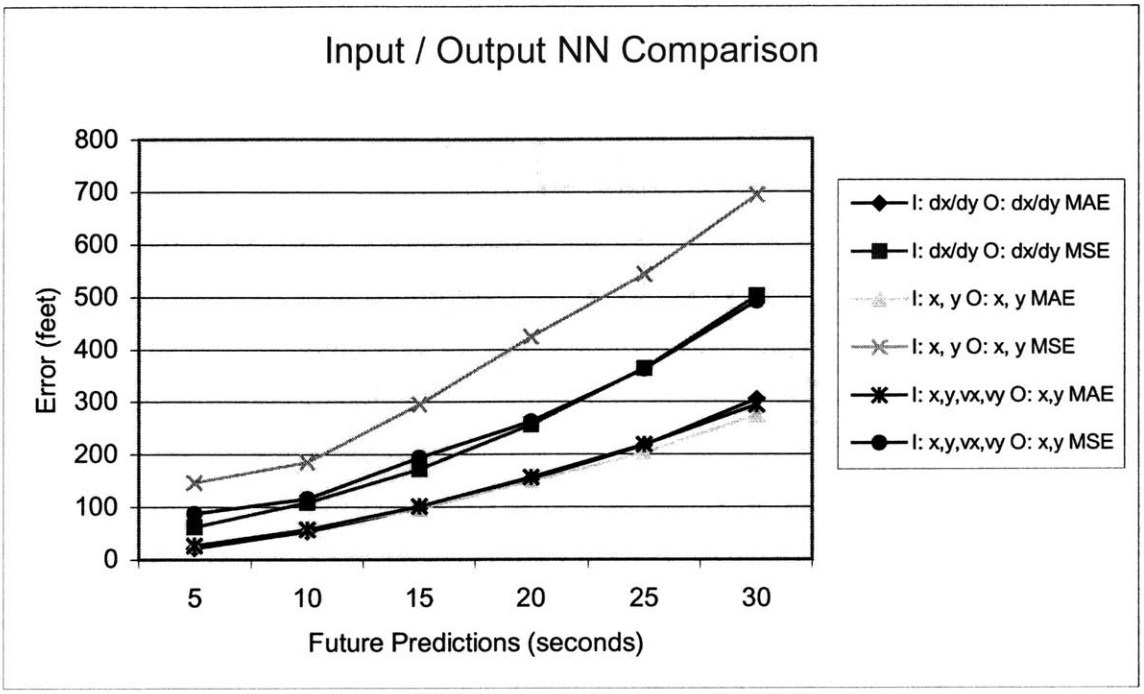

Figure 9. A comparison of the NNs with different inputs and outputs.

Input velocities, compared to input coordinates, result in more accurate predictions because the training set becomes more generalized when coordinate positions are converted into velocities. Thus, there are many more cases to teach the neural network about general flight patterns. The velocity-input NN learns general airplane movement instead of movement based on absolute position. A NN which learns movement based on actual position is very sensitive to its training set selection algorithm and often results in overfitting. Using coordinate inputs results in poor predictions when an airplane is landing on a runway in an atypical orientation. For example, if the network is trained with the majority of airplanes taking off eastward on a given runway, then a plane taking off in the opposite direction would produce poor predictions using a coordinate based neural network model.

4.4 Training Data

This section presents the performance of neural networks trained with various subsets of training data. The results of the two neural networks in this section should be compared to

each other, as well as the control neural network (NN #1), which was trained with 0% Stop, 50% of Taxi, and 100% of High Speed training data available. The following table is the performance of a neural network trained with all of the available training data.

Table 7. Performance of NN trained with all available training data.

NN #5	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	74.1291	151.4362
	10	90.8942	206.9486
	15	150.6029	331.8271
	20	180.7897	410.8898
	25	277.706	593.7297
	30	337.6999	719.9308
Y coordinate predictions	5	98.9886	218.2521
	10	97.732	193.4412
	15	115.8326	240.1898
	20	171.9372	354.6423
	25	240.785	478.0171
	30	307.1436	612.7633

NN #6, whose results are presented below, was trained with 100% of the available high speed data. The same absolute quantity of Taxi data was also used to train the NN, so the resulting data training set was balanced with 50% High Speed and 50% Taxi data.

Table 8. Performance of NN trained with a balanced set of training data.

NN #6	time in the future (s)	Absolute Error	Mean Square Error
X coordinate predictions	5	39.8228	164.5383
	10	78.2687	221.8536
	15	144.9835	349.9594
	20	214.513	506.7347
	25	286.3815	635.2542
	30	377.6536	801.9031
Y coordinate predictions	5	42.493	85.0201
	10	98.3929	170.6203
	15	167.7627	273.7703
	20	266.1394	419.2434
	25	354.864	539.8046
	30	496.871	711.7728

NN#4, (table 3) trained with 0% of the Stop data, 50% of the Taxi data, and 100% of the high speed data performed the best. It performed significantly better than the other two neural networks.

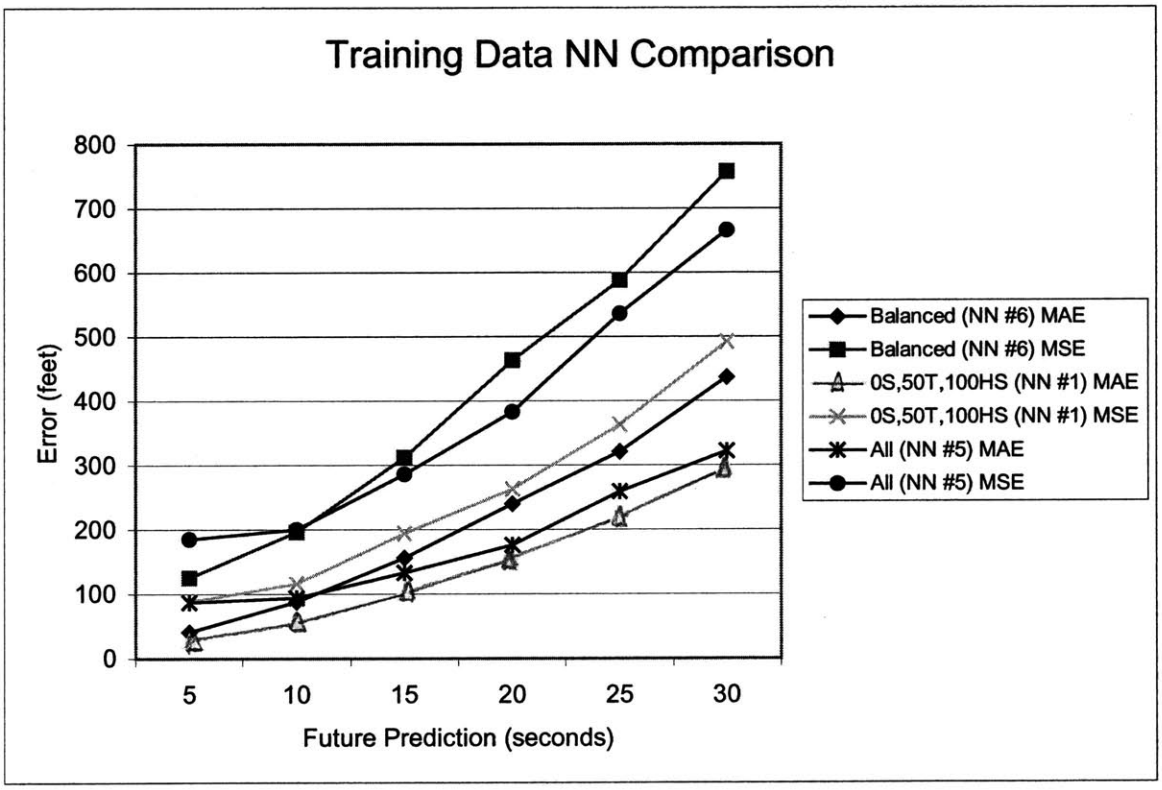

Figure 10. A comparison of the NNs trained with different amounts of training data.

The results of this experiment are somewhat surprising. The NN trained with all of the data performed much worse than the winner, which was trained with a subset of all the data. One reason is that the stop data biased the neural network to predict output values too close to the input values. The NN trained with a balanced dataset did not perform well potentially because there was not enough data. Since it performed worst in this experiment, it is best to move forward with the neural network trained with 0% of the Stop data, 50% of the Taxi data, and 100% of the High Speed data.

4.5 Optimal Neural Network

Through the results of the various experiments, an optimal neural network presents itself. The neural network uses dx or dy as its input and output. It takes in 10 seconds of past history as an input. The neural network should be trained with 0% of the Stop data, 50% of the Taxi data, and 100% of the High Speed data.

In conclusion, the new optimal neural network yields accurate position predictions and lends itself easily to predict alarm situations. The advances discovered in this research bring us closer to our goal of earlier and more accurate alarms, which will greatly enhance aircraft safety. Note that the same neural network will be used to predict movement in the x and y independently.

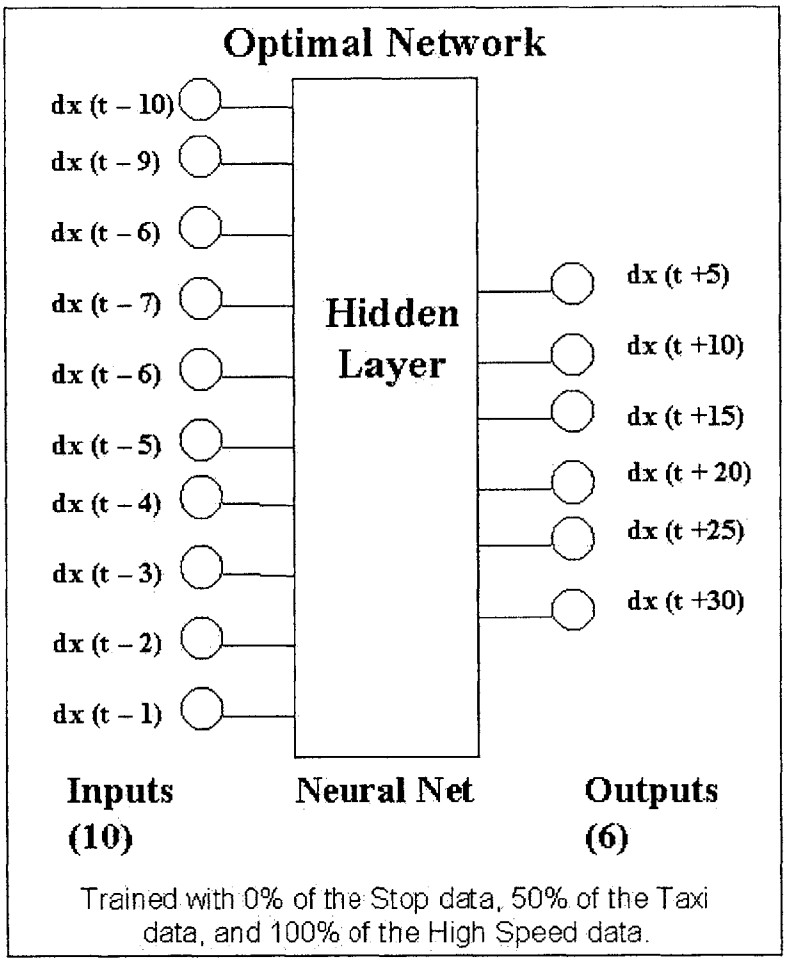

Figure 11. Diagram of the optimal neural network for aircraft position prediction.

4.6 Optimal Neural Network Analysis

The optimal neural network was integrated into the AMASS by modifying the source code provided by Volpe. Further performance analysis was conducted by running AMASS in simulation mode to evaluate its performance in the control tower. The performance comparison conducted so far has only compared the prediction error of the change in X or Y coordinates. Since the neural network is integrated in to the NN-AMASS at this stage, it is able to output the predicted separation distance between two airplanes. Thus we can compare the prediction capabilities of the AMASS prediction model and Neural Network prediction model. The neural network's performances were evaluated using ORD0821 Log file, and all the events tested were situations when the AMASS monitors and predicts airplane separation distances. The test sets were performed on four separate situations; All, HS-HS, HS-LS, and LS-LS.

- All scenarios include all the events except for the Stop-Stop, where both airplanes are in Stop mode. This will provide an overall accuracy of the neural network system compared to the AMASS system.
- HS-HS events include situations where both airplanes are in high-speed mode. The events considered as high speed are Arrival, Landing, Takeoff, and Departure.
- LS-LS events are situations where both airplanes are in either Taxi or Stop mode.
- HS-LS events are situations where one aircraft is in high-speed mode while the other is in low speed mode.

The number of events that match each of these categories in the log file ORD821 are in the following table:

Table 9. Distribution of Situations (data points)

Time (sec)	Stp-Stp	HS-HS	HS-LS	LS-LS
5	365	203	499	3817
10	254	103	169	2734
15	198	41	48	1972
20	157	4	17	1407
25	130	0	7	987
30	97	0	2	727

As shown in Figure 12, the initial testing shows that the neural network is slightly better at predictions 25 and 30 seconds in the future compared to the AMASS; however, for seconds 5 through 20 in the future, the AMASS seem to be predicting more accurately and consistently. The latter observation leads one to more closely investigate this particular neural network model. Additionally, a look at the distributions of the test situations shows that most situations tested were LS-LS, which are much easier to predict and less likely to trigger an alarm.

Results Filtered (no Stp-Stp)				
	NN Prediction		AMASS Prediction	
	MAE	MSE	MAE	MSE
5	172.4	1747.9	71.4	549.8
10	391.1	5163.4	103.4	451.7
15	426	6557.2	138.1	210.5
20	220.3	365	195.4	287.6
25	277.5	409.4	289.2	435.5
30	370.8	546.5	386.8	594

Results HS-HS				
	NN Prediction		AMASS Prediction	
	MAE	MSE	MAE	MSE
5	770	4137	184	386
10	2482	13673	213	325
15	2700	3115	322	431
20	3149	3312	373	398
25	N/A	N/A	N/A	N/A
30	N/A	N/A	N/A	N/A

Results HS-LS				
	NN Prediction		AMASS Prediction	
	MAE	MSE	MAE	MSE
5	95	454.7	30.2	162.9
10	405.3	1897.4	40.9	182.2
15	1008.8	4285	22.5	29.2
20	74.7	95.2	23.2	30.2
25	90	96.6	22.6	23.9
30	159	166.6	19.2	19.2

Results LS-LS				
	NN Prediction		AMASS Prediction	
	MAE	MSE	MAE	MSE
5	38.8639	57.9987	35.2082	56.0249
10	86.0352	128.019	80.3272	123.2421
15	143.5018	209.9712	132.2069	200.9412
20	205.6121	305.1151	194.4219	287.072
25	273.1189	402.6833	289.6508	436.5538
30	367.4307	540.2458	387.3256	594.7503

Figure 12: Results of Position Prediction Neural Networks

For the HS-HS events, future 25 and 30 second prediction comparison could not be performed because there were no data available. Additionally, some results are misleading, such as future 30 seconds predictions for HS-LS events, because there is not enough data.

Some data was available in predicting future 5 to 20 seconds for the HS-HS and HS-LS situations, however the neural network forecast of the future separation distance is inferior than the AMASS forecast. This is concerning as the most dangerous situations result when both airplanes are at high speed (HS-HS) or at least one is in high speed (HS-LS).

The above comparison illustrates the mediocre accuracy performance provided by this particular neural network system, especially in the events that include high-speed situations. To investigate why the neural network is doing so poorly in the high speed situations, detailed analysis was conducted of an event that took place on Aug 21 02:35:16 when an airplane was landing while another one was on the designated runway.

The coordinates of the landing airplane were not taken uniformly, hence producing large gaps or narrow gaps, as shown in Figure 14. T1 is the airplane landing on the runway, and T2 is the airplane taxing on the runway. The non-uniform sampling is a product of the ASR radar system which is less accurate than ASDE.

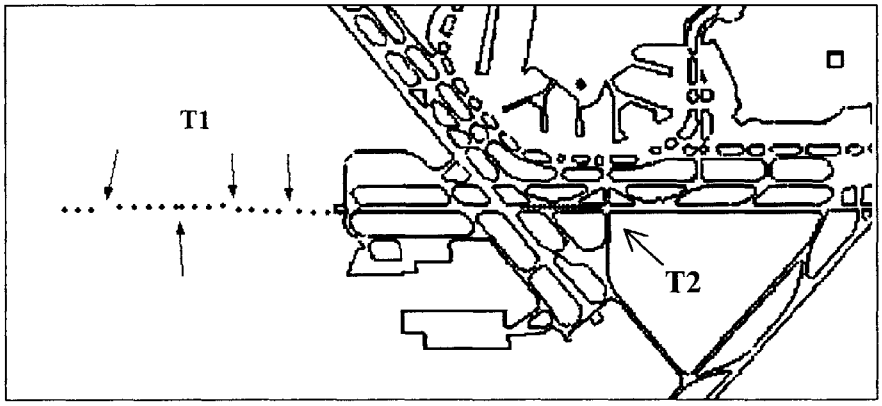

Figure 13: Simulation of Situation on Aug 21 02:35:16. Note the inaccuracy of the radar sampling rate.

As a consequence, the speed of the airplane appears as non-uniform while in average is 288 ft/s. In Figure 15, the velocity of both aircraft are graphed, and it is clear that the speed of T1 was sampled non-uniformly, while the slow moving target T2 shows a clean signal.

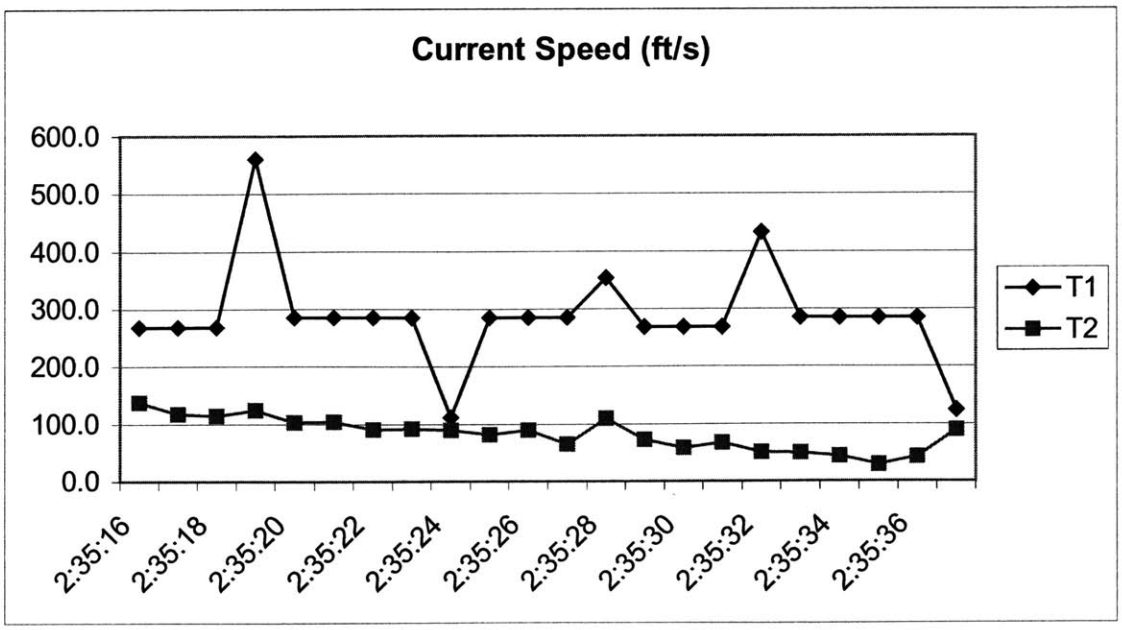

Figure 14: Speed of T1 and T2 estimated as difference between current position and previous position

Therefore, as shown in Figure 16, the estimation for displacement made by the neural network is not uniform and shows values too large (at 2:35:19) or too small (at 2:35:24).

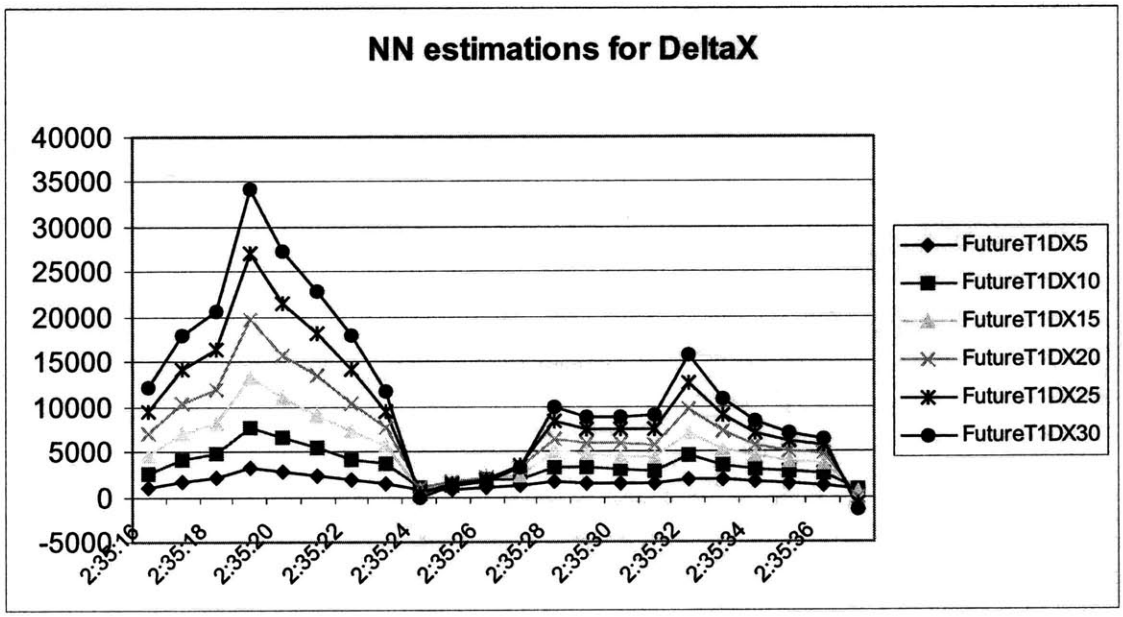

Figure 15: Estimations of displacement in X direction for T1

The neural network should have taken into account the effect of bad sample spacing to make better estimations, especially because it is using ten samples of the past data. However, it appears that the neural network is not working with the expected level of

accuracy. Figures 17 and 18 show the values of the separation distances for different estimation times: 15 and 20 seconds. Results for 25 and 30 seconds are not shown because most of the real values are not available.

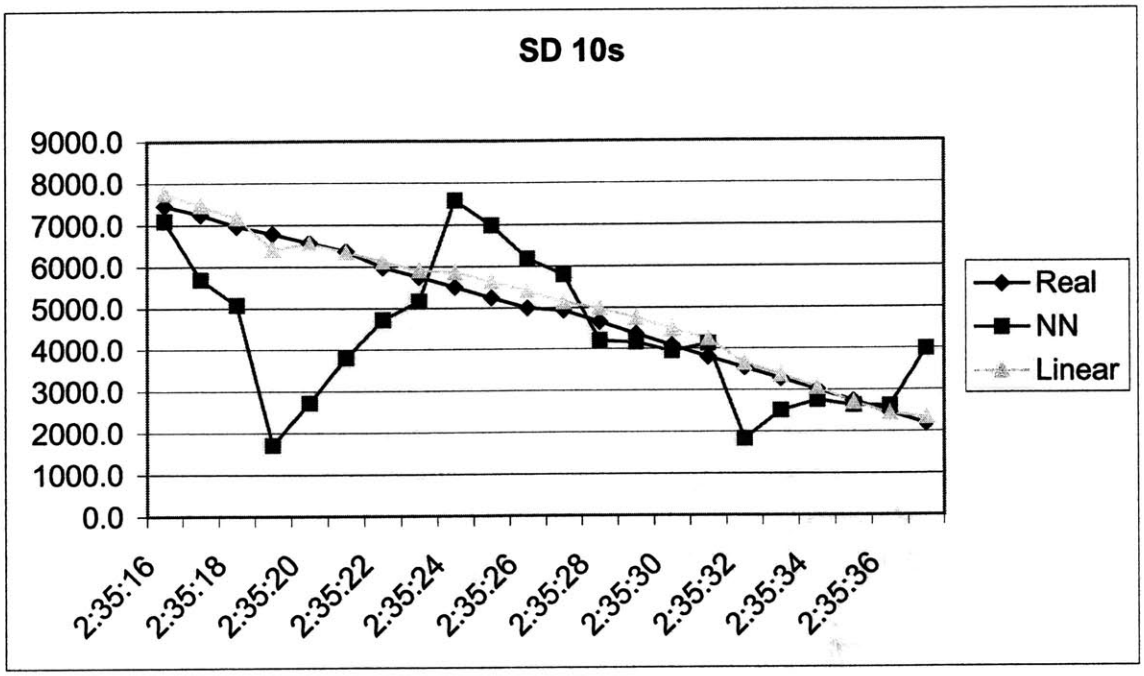

Figure 16: Separation Distance at 10 seconds in Future

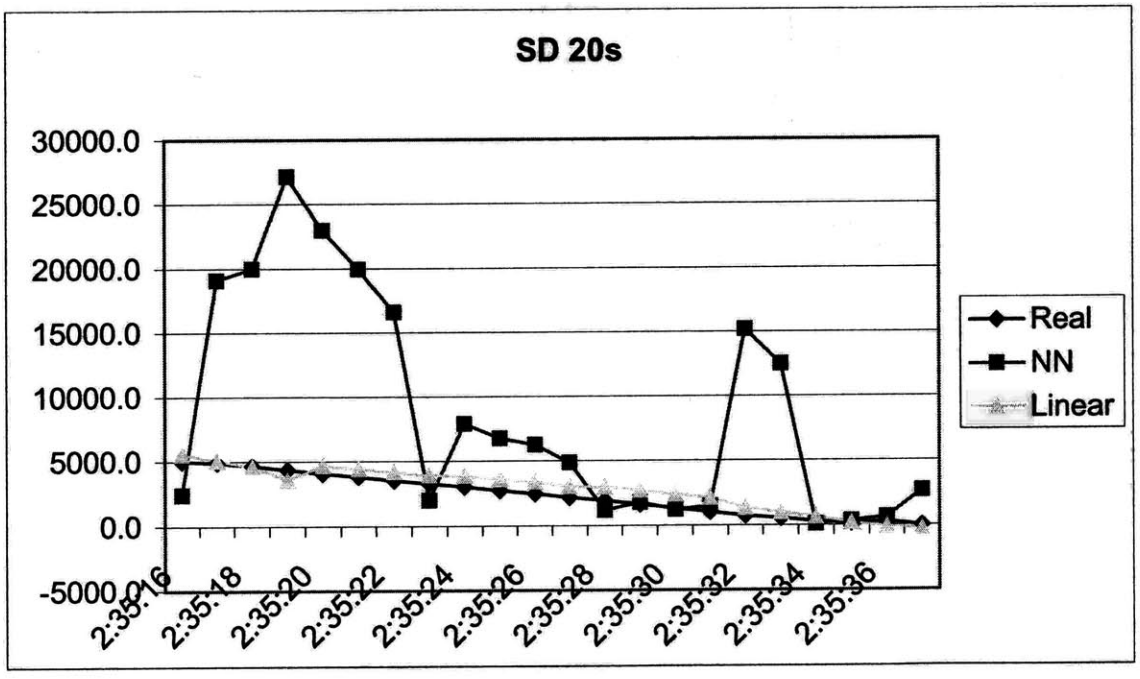

Figure 17: Separation Distance at 20 seconds in Future

In order to check if the neural network was correctly implemented in AMASS, graphs were generated to compare the estimation of delta values in X direction with the actual delta values, which are similar to the data used to train the neural network. The graphs in Figure 19 and 20 show very large errors for neural network predictions. In fact, the mean absolute errors were: 531.3 ft (t=5s), 1330.1 ft (t=10s), 2356.1 ft (t=15s), and 3529.4 ft (t=20s), which are not consistent with the accuracy analysis shown previously in this chapter.

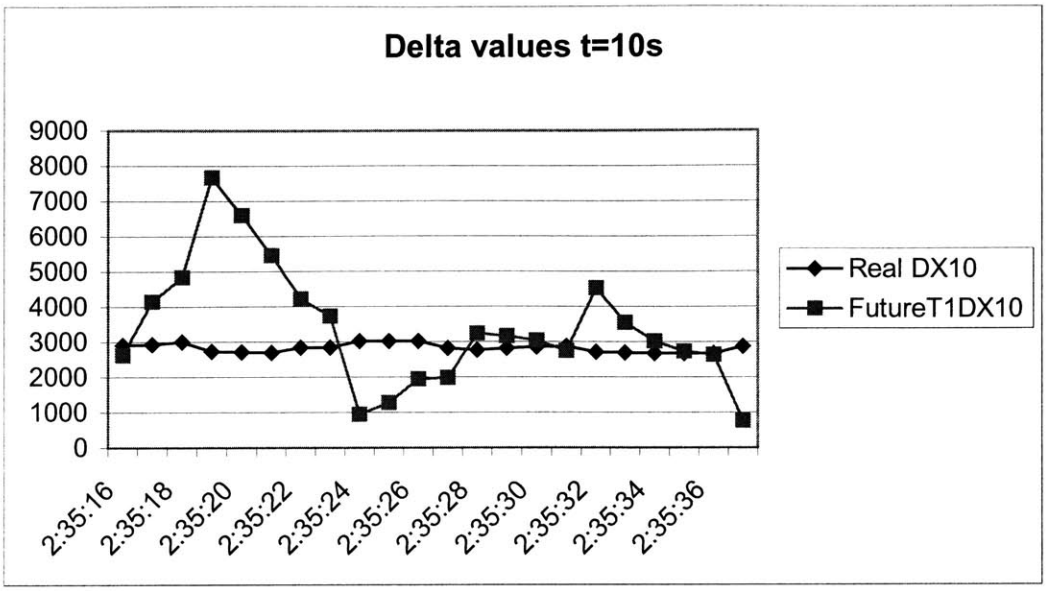

Figure 18: Delta X at 10 seconds in Future

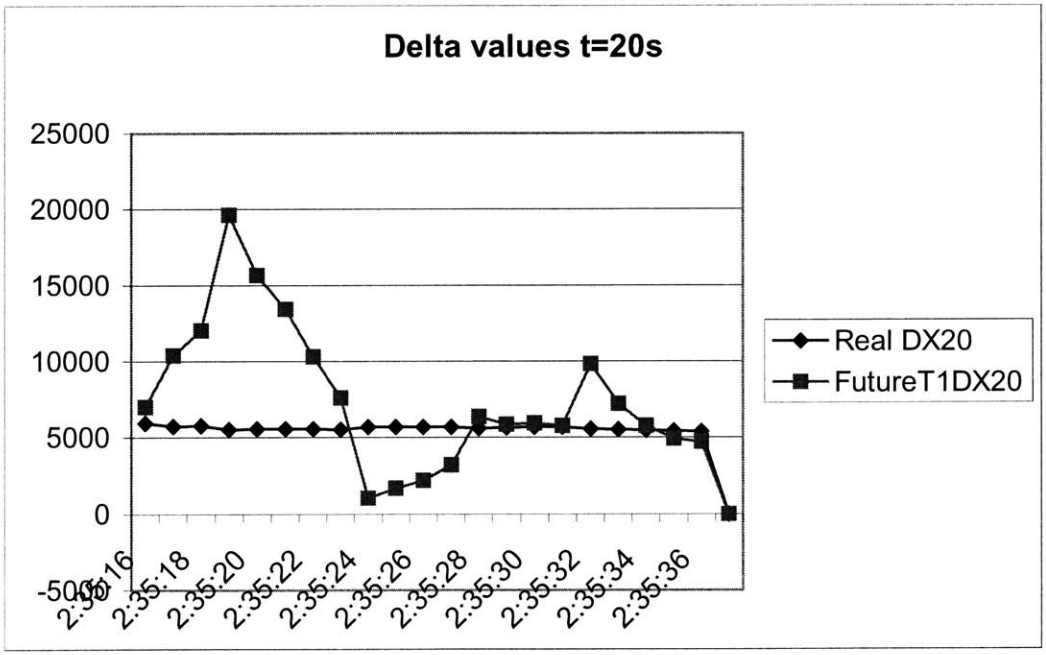

Figure 19: Delta X at 20 seconds in Future

From the investigation above, it was concluded that re-training of the neural network was necessary. Neural networks are known for their ability to make accurate predictions despite imperfections in the input data. However, our neural network did not robustly handle the flawed input data. Since it could not handle poor sample spacing, a new method of data extraction was investigated and implemented. As explained in Chapter 5, the new extraction method obtains all Landing and Takeoff events in ORD runways, and trains the neural network with this data.

5. DESIGN PHASE II: A NEW APPROACH

This section discusses the motivation for creating a new approach that involves using two neural networks and a new method for computing separation distance. It also presents the algorithm used for determining the incursion distance between two planes, that is based on trigonometric analysis and yields SD values useful for danger detection.

5.1 ASR and ASDE Neural Networks

Previously, the neural network was trained with coordinate data from ORD log file, with attention given to the balance of stop, high speed, and low speed events. There are two radar systems used at the Chicago O'Hare airport: ASR and ASDE. ASR tracks airplanes in the sky and it is imprecise in its measurements. ASDE tracks airplanes on the airport surface. The new extraction method involves extracting each landing and takeoff event individually. This method is able to handle the switch of the radars as airplane coordinates from both ASDE and ASR radar are appended to make a complete takeoff or landing. Thus, each event is longer, and there will now be enough data to train the neural network to make predictions up to 30 seconds in the future. Note that the previous NN was only trained with two HS-LS events and no HS-HS events for predictions 30 seconds in the future. The obtained coordinates are transformed into neural network input/output pairs (delta input/output) to be used as training sets.

Additionally, the training sets were separated into two sets depending on which radar is tracking the airplane at the time. By dividing the training sets by the radars two neural networks, ASDEnet and ASRnet were trained. The advantage of having two neural network systems is that one neural network (ASRnet) can handle the noise in the input when the airplane is being tracked by ASR radar, while the other neural network (ASDEnet) can be trained for inputs without any noise when the ASDE radar is tracking the airplane. Furthermore, because ASR radar observes airplanes outside of the runways and ASDE radar monitors airplanes on the runways, it can be said that ASRnet predicts the characteristics for high-speed airplanes and ASDEnet predicts the characteristics for low speed airplanes. Furthermore, important work was done to extract landing and departure

events in every of the six runways at ORD, in order to train the NN for all possible operations [19].

5.3 Incursion Distance Algorithm

The AMASS system sounds an alarm when a negative separation distance is predicted. The first implementation of the neural network into NN-AMASS employed the Euclidean distance formula to obtain the future separation distance between two aircrafts. This distance, Separation Distance, is useful to observe the accuracy of the Neural Network Model, however is disadvantaged in alert triggering as the model is unable to produce negative future distances. The AMASS prediction of the future distance between two airplanes is calculated depending on the positions and the directions two airplanes are travelling. Through this computation method the AMASS is able to attain negative future distances, which will trigger alerts in the safety cell. Therefore, addition of the future distances parameter for alert triggering, called Incursion Distance, is to be included to further enhance the neural network model. This algorithm was primarily designed by Dr. Rafael Palacios.

Using the AMASS code as a reference, the two airplanes will be categorized by the relative position and direction that each airplane is travelling. There are three categories:

- **Head-On**: Two planes are travelling toward each other. (Figure 21)
- **Chase**: One plane is travelling similar direction as other airplane. For example a Lander behind Lander. (Figure 22)
- **Apart**: One plane is travelling opposite direction as other airplane. (Figure 23)

Since single-surface events are only considered, all three categories are related to a pair of airplanes moving in the same runway. Therefore we are interested in distances attained along the main runway direction in order to compute current separation distances (CurrentSD) and predicted separation distances (FutureSD).

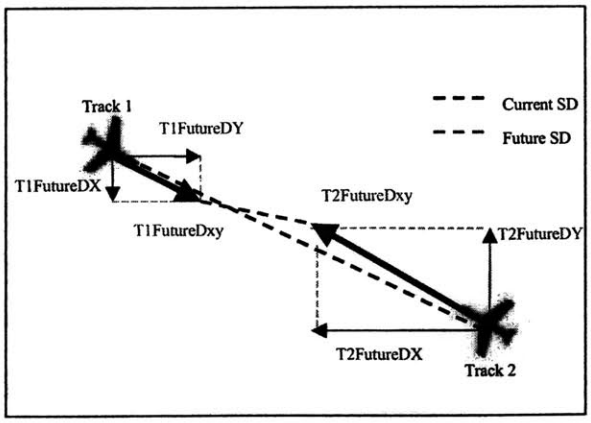

Figure 20. Head-on event.

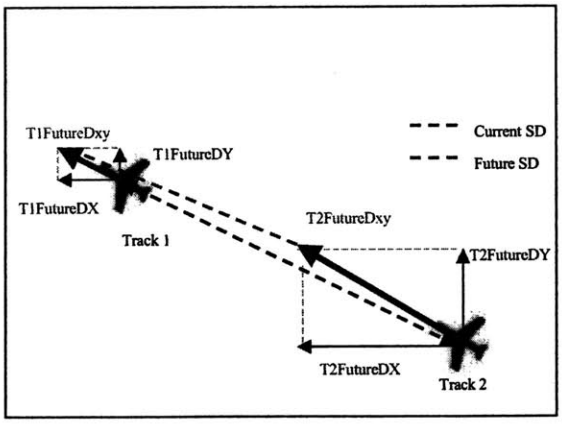

Figure 21. Chase Event.

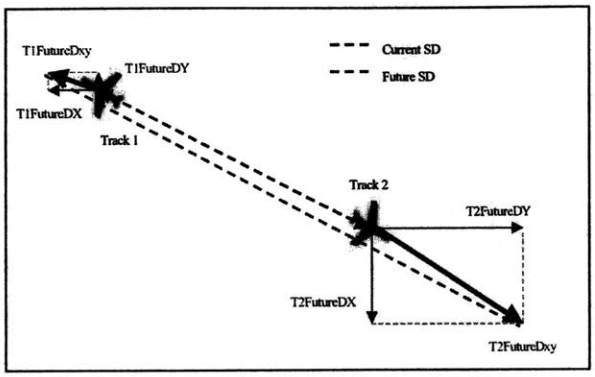

Figure 22. Apart Event.

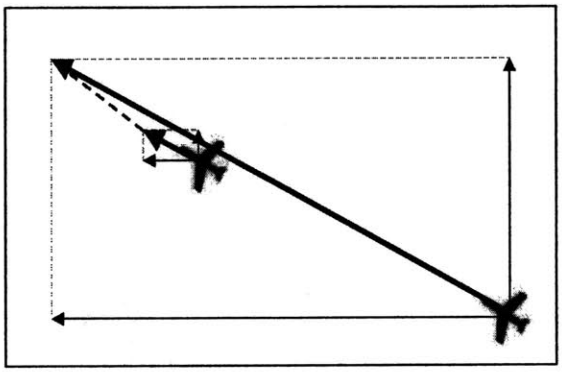

Figure 23. Chase Event, Overtake.

The neural network prediction is used independently for X direction and Y direction, providing values of the predicted distance that the airplane will move in each direction. For example for predictions 20 seconds into the future T1FutureDX20 and T1FutureDY20 are obtained, that represent the distance that target 1 will travel during the next 20 seconds in X and Y direction respectively. The neural network developed is able to simultaneously compute predictions for 5, 10, 15, 20, 25, and 30 seconds into the future. Nevertheless for the following examples, only the variable for 20s in future will be shown.

The estimation of airplanes future positions and the estimation of future separation distance can be obtained using the following equations:

$$T1Future20X = T1X + T1FutureD20X$$
$$T1Future20Y = T1Y + T1FutureD20Y$$
$$T2Future20X = T2X + T2FutureD20X$$
$$T2Future20Y = T2Y + T2FutureD20Y$$
$$CurrentSD = \sqrt{(T1X - T2X)^2 + (T1Y - T2Y)^2}$$
$$FutureSD20 = \sqrt{(T1Future20X - T2Future20X)^2 + (T1Future20Y - T2Future20Y)^2}$$

Quite often vector T1Future20xy or vector T2Future20xy are not exactly parallel to the runway direction, as shown in previous figures. This is a consequence of small instantaneous changes in bearing, noise in radar signals or error in the estimation of FutureDX or FutureDY. As a consequence CurrentSD and FutureSD get values slightly larger that the real values.

Another problem is that computing FutureSD by using the basic trigonometric functions of distance always yields positive values. Therefore in a situation like the one depicted in figure 24, where target 2 overtakes target 1, FutureSD computed with the previous equations is a large positive value that will not generate an alarm. In this situation it makes more sense to change the sign of FutureSD and return a negative prediction.

Accordingly, there was a need to define another way to compute predicted separation distance able to produce negative values and able to minimize the impact of noise. That distance is called Incursion Distance.

The best way to minimize the impact of noise in this environment is to identify the runway direction and project every vector over that direction. Moreover, working with projected distances the computation of Incursion Distance is simplified because it is converted just in additions or subtractions.

Figure 25 shows an event similar to the one shown in figure 24 and the projected vectors and distances. Future incursion distance is computed after the values of projected magnitudes (noted with "p").

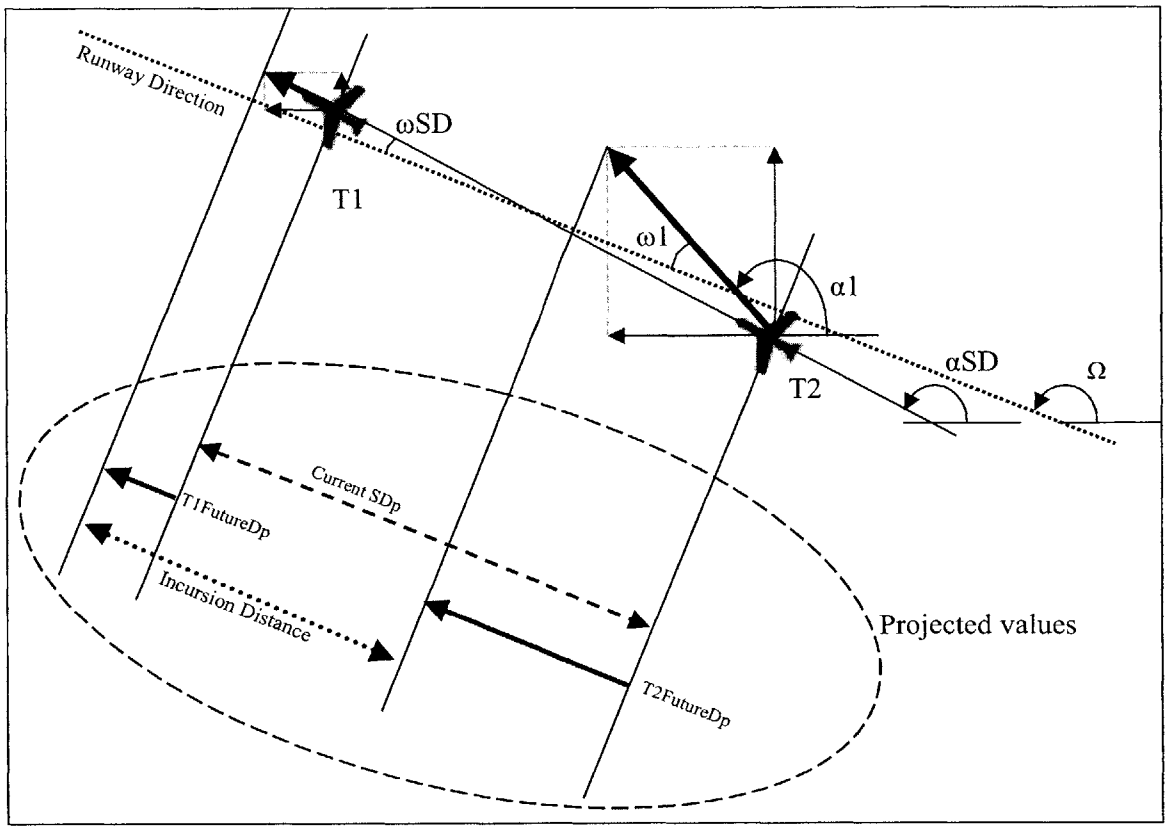

Figure 24: Calculation of Incursion Distance in a Chase Event

Using projected magnitudes, computing incursion distance is as simple as the following equation:

$$IncDist = CurrentSDp + T1FutureDp - T2FutureDp$$

According to the angles show in Figure, projected magnitudes are calculated as:

$$CurrentSDp = CurrentSD \cdot \cos(\omega SD)$$
$$T1FutureDp = T1FutureDxy \cdot \cos(\omega 1)$$
$$T2FutureDp = T2FutureDxy \cdot \cos(\omega 2)$$

The main runway angle (Ω) is obtained by rounding the direction (bearing) of the fastest moving airplane, since runway directions are always multiples of 10 degrees. Runway direction should be similar to T2T1 vector. Once Ω is known, ω1, ω2, and ωSD are computed as difference with the basic angles. The basic anbles α1, α2, and αSD are easily obtained using atan2 function (in C or Matlab).

$$\alpha 1 = \text{atan2}(T1FutureD20Y, T1FutureD20X)$$
$$\alpha 2 = \text{atan2}(T2FutureD20Y, T2FutureD20X)$$
$$\alpha SD = \text{atan2}(T1Y - T2Y, T1X - T2X)$$

Figure 26 shows the same category of event in a situation where target 2 is moving at a much higher speed, so that it is able to overtake target 1. In this case incursion distance should be negative, as it is automatically obtained applying the equations.

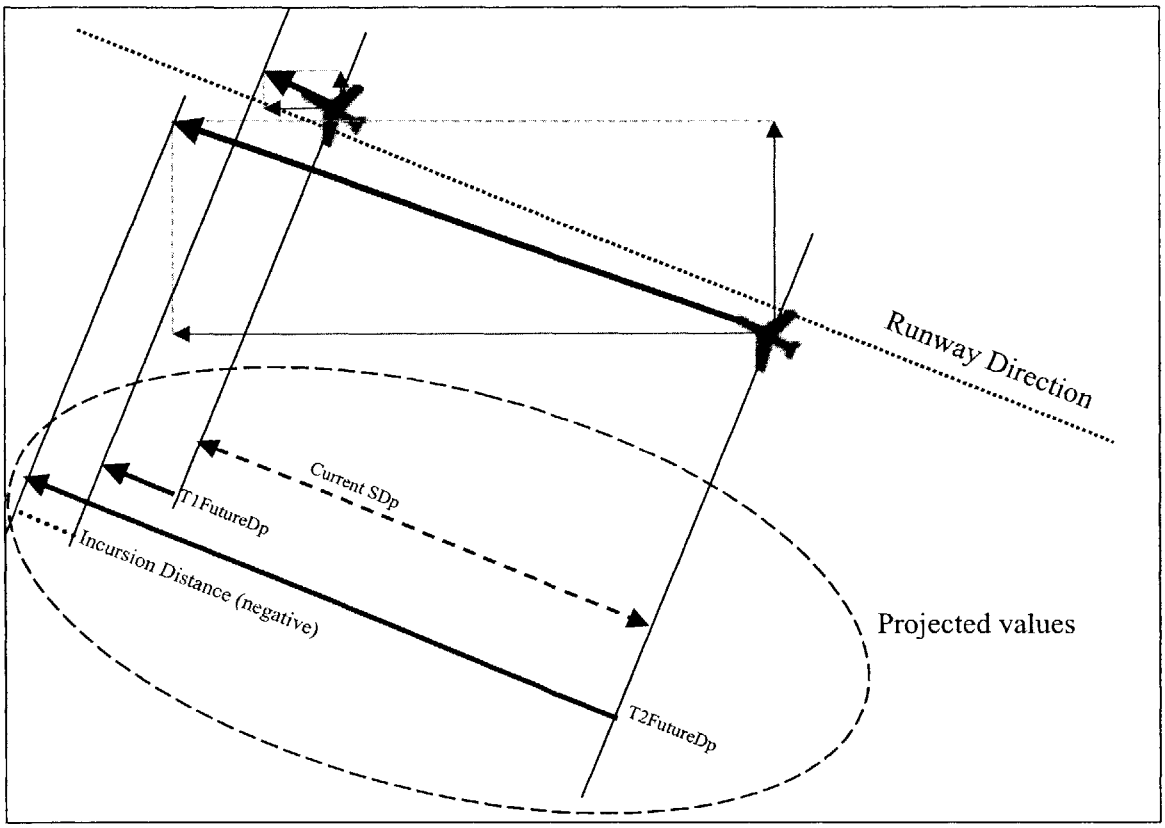

Figure 25: Calculation of a Negative Incursion Distance

The only difference with other categories is the way to compute incursion distance. The equations are:

$$Head-on: IncDist = CurrentSDp - T1FutureDp - T2FutureDp$$
$$Chase: IncDist = CurrentSDp + T1FutureDp - T2FutureDp$$
$$Apart: IncDist = CurrentSDp + T1FutureDp + T2FutureDp$$

Nevertheless it is not necessary to find which particular category is being computed, because being careful with angle definitions, projected vector will automatically become negative magnitudes. Hence only the first equation was used for implementing calcIncDist function.

6. PHASE II FINDINGS

Section 6.1 presents the findings of the dual neural network model. Section 6.2 shows the capability of the neural network model after the incursion distance algorithm was implemented.

6.1 Dual Neural Network Results

After improving the extraction tools and training new neural networks for ASDE and ASR data separately, similar analysis with the 02:35:16 event on August 21 was conducted. The initial result shows that new neural networks are more accurate than previous. Using the current approach, the estimation of delta values is now very accurate, as shown in Figure 27 and 28.

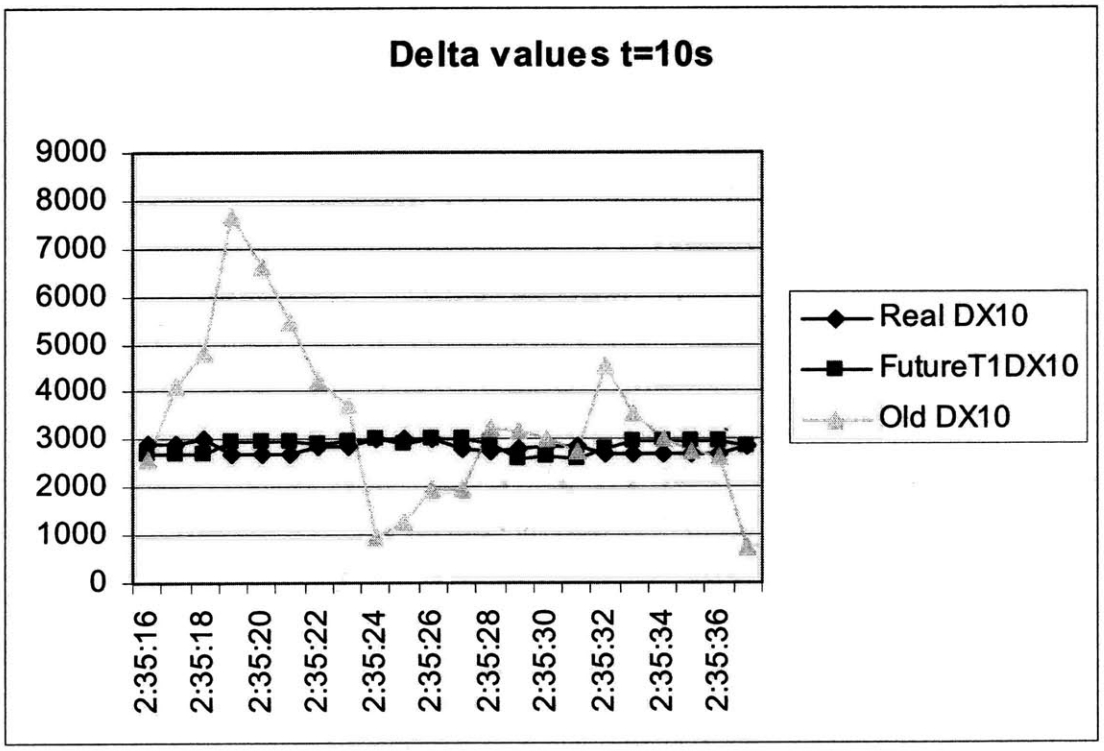

Figure 26: Delta X values for 10 seconds in Future

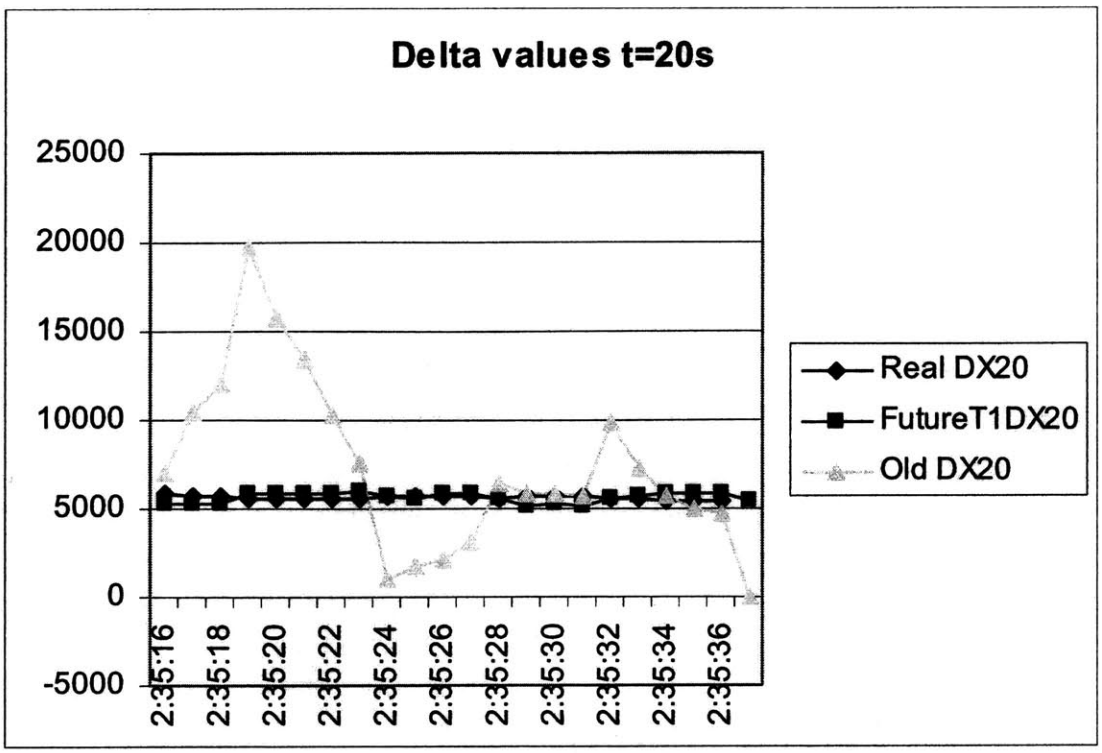

Figure 27: Delta X values for 20 seconds in Future

The graphs above show that new neural network is now able to handle the noise in the input. While the old neural network struggled with the bad sampling of the ASR radar, as there were huge ups and downs in the delta values in the X direction, the new neural network handles the bad sampling very well. To ensure that the neural network has improved at all levels, the separation distance between two airplanes for the event was also compared. The graph below clearly illustrate that the neural networks exhibit significant improvements over the last set of neural networks.

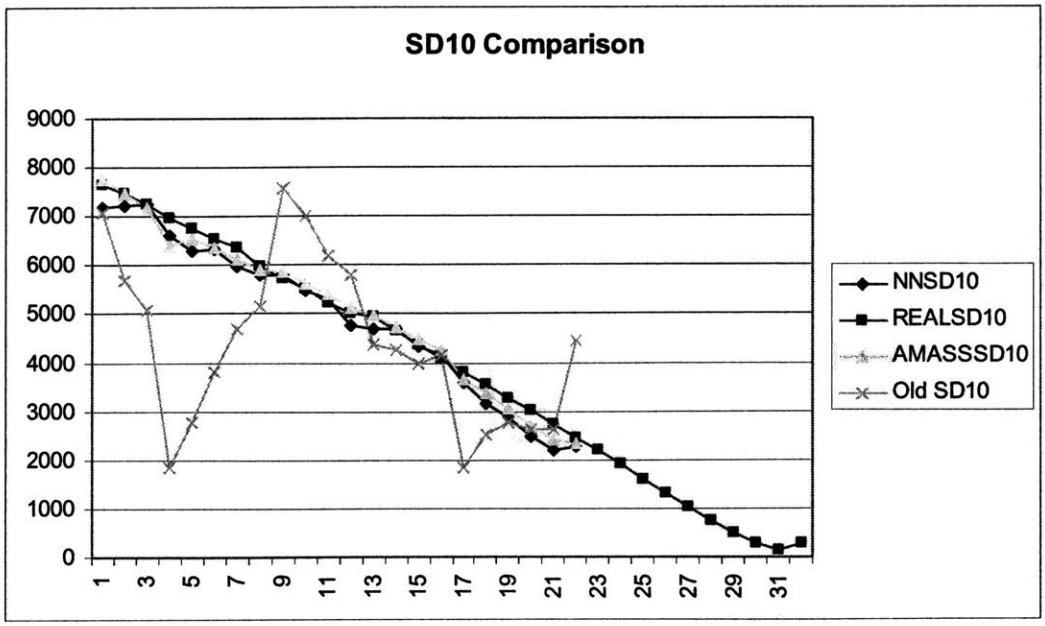

Figure 28: Separation Distance at 10 seconds in Future

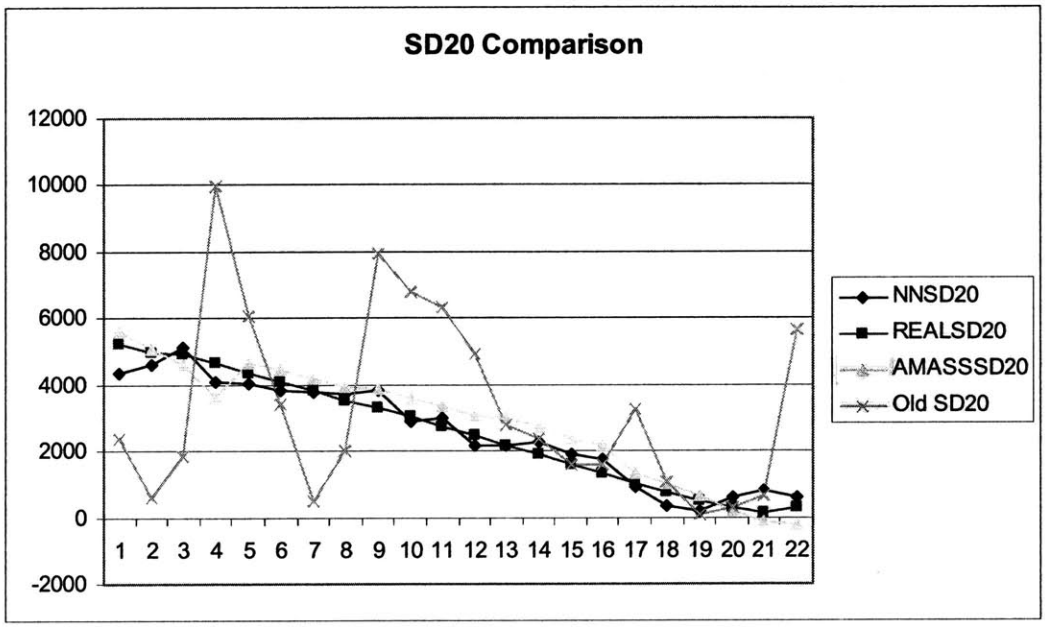

Figure 29: Separation Distance at 20 seconds in Future

The separation distance also improved dramatically from the old neural network to new neural network. Although the neural network was only partially trained with small set of data, in the figure above, the new approach appears to be performing better than AMASS predictions.

Additionally, similar test was conducted using the ORD0821 Log file to evaluate the neural network's performance. The result is as follows:

	NN	AMASS
	MAE	MAE
5 Seconds	56.85	95.52
10 Seconds	102.82	116.09
15 Seconds	145.13	160.97
20 Seconds	193.72	230.52
25 Seconds	237.75	294.72
30 Seconds	267.46	401.39

Figure 30: ORD0821 Results (all predicted data)

The test confirms that the new neural network is performing better than the AMASS linear algorithm.

6.2 Incursion Distance Results

In order to examine the incursion distance performance, an alarm-triggering event was analyzed carefully. The ORD0821 02:35:16 event was simulated, and following are the distance comparison of neural network predicted separation distance, AMASS predicted separation distance, actual separation distance, and neural network predicted incursion distance for future 20 and 30 seconds.

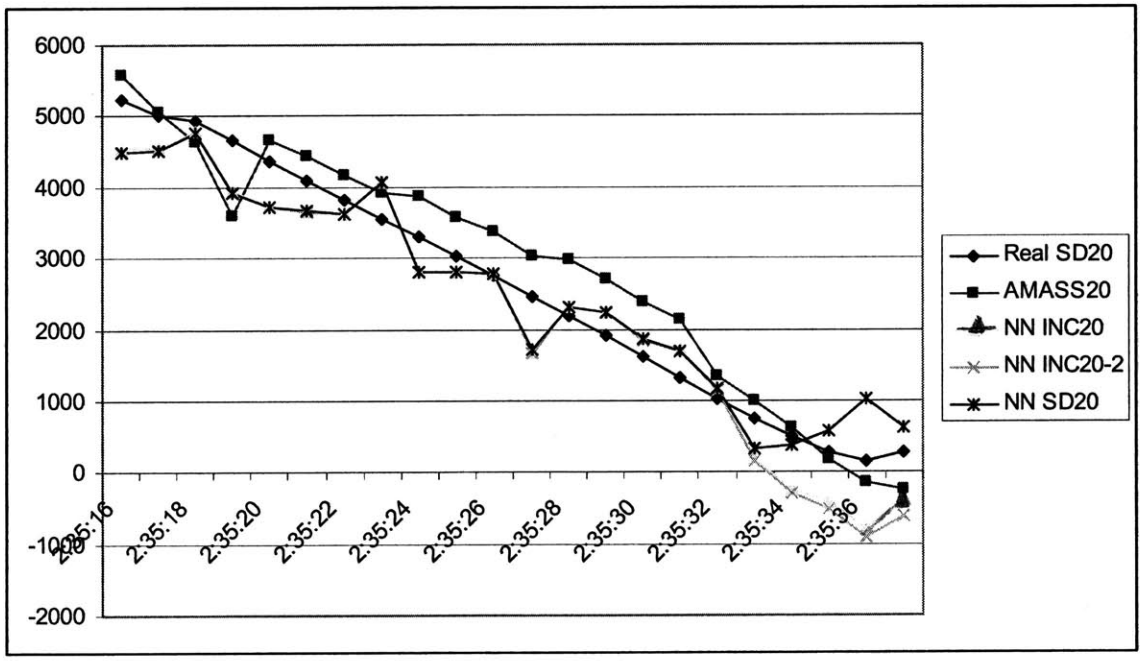

Figure 31: Incursion Distance Results for ORD0821 02:35:16 Event (20 sec)

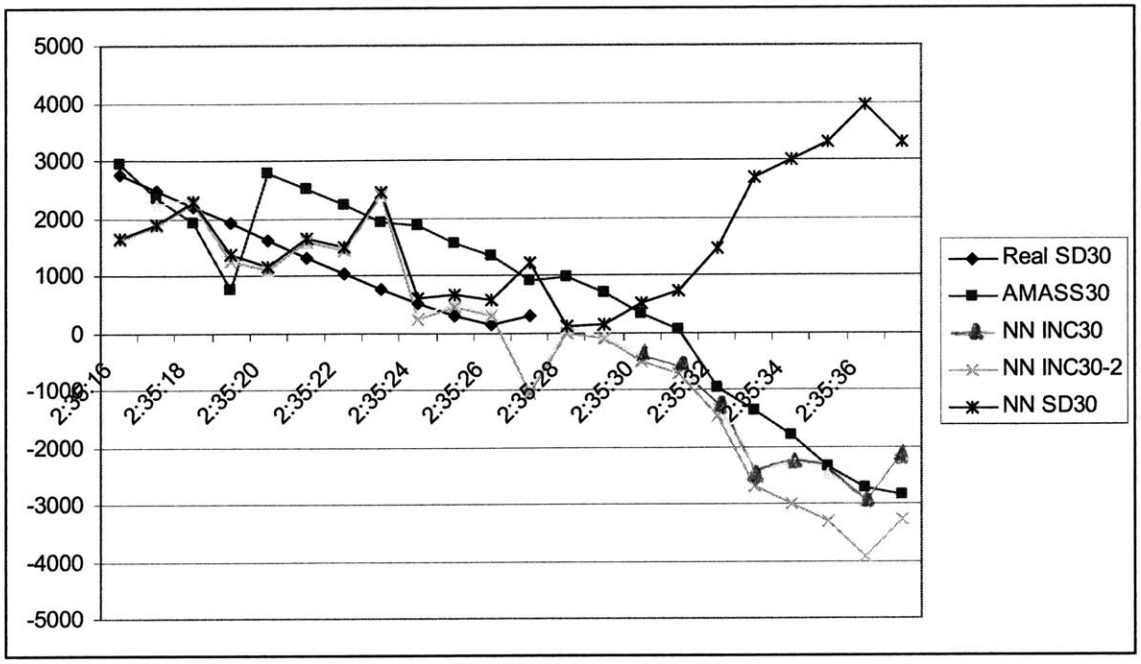

Figure 32: Incursion Distance Results for ORD0821 02:35:16 Event (30 sec)

The graphs above display that neural network incursion distance predicts the possible collision two seconds ahead of the AMASS for future 20 seconds prediction. The neural network incursion distance dips below the zero negative distance at the 19th second, triggering an alarm for the neural network, while AMASS prediction becomes negative at the 21st second. Additionally, for the future 30 seconds prediction, the neural network incursion distance triggers an alarm at the 12th second, which is five seconds ahead of equivalent scenario with AMASS.

It is difficult to evaluate accuracy in real events because pilots take actions to avoid dangerous situations. On August 21st, 2001, at 2:35:27, the system predicted that in 30 seconds, an airplane landing at ORD runway 09R was going to reach another airplane moving slowly on the same runway. Fortunately, the pilot was alerted and the airplane aborted the landing by flying over the airport, so it is difficult to tell if the estimation was correct. Nevertheless, it can be seen in the graphs that the evasive maneuver was performed at constant speed and that the minimum distance between both planes was attained at 2:35:56, which matches the NN prediction within 1 second.

7. ANALYSIS

This chapter presents a concrete comparison of our neural network model with other prediction methodologies. The first three sections introduce the models that the neural network will be compared to, and the fourth section presents the actual comparison errors. All models for position prediction were trained and tested using real data extracted from log files recorded at ORD during the summer of year 2001. The events extracted comprise examples of landing, take off and taxi using all possible runways. The extracted data was randomized into a training and validation set, used for adjusting the models, and a testing set used to generate statistics.

In addition, another analysis was performed based on the number of alarms

7.1 Constant Speed Model

Airplanes and ground vehicles are subject to inertia so sudden changes in position or bearing are not possible. Therefore one method of forecasting aircraft future position is to estimate current speed vector $\hat{v} = (\hat{v}_x, \hat{v}_y)$ and compute future position assuming constant speed. Computation is done independently for x direction and y direction.

$$\left.\begin{aligned} \hat{x}(t+\delta) &= x(t) + v_x \cdot \delta \\ \hat{y}(t+\delta) &= y(t) + v_y \cdot \delta \end{aligned}\right\} \quad \text{where} \begin{cases} \delta = \{5,10,15,20,25,30\} \\ (v_x, v_y) \text{ is the speed vector, which in practice is also an estimation} \end{cases} \tag{1}$$

Instead of using past coordinates as inputs to compute the speed vector, it was found more convenient to compute differences of coordinates which will be referred to as delta values from here onwards. Delta values order 1, noted $dx1$ and $dy1$, are the differences between target position at time t and target position at time $t+1$, therefore we have:

$$\begin{aligned} d\hat{x}1(t) &= \hat{x}(t+1) - x(t) \qquad \text{(estimated)} \\ dx1(t-1) &= x(t) - x(t-1) \\ dx1(t-2) &= x(t-1) - x(t-2) \\ &\vdots \end{aligned} \tag{2}$$

In the same way we define delta values order n, noted dxn, as the difference between position at time t and position at time $t+n$. This notation will be used to describe the

distance that the airplane will make during the following 5, 10, 15, 20, 25 and 30 seconds, as predicted by the model:

$$\begin{cases} d\hat{x}5(t) = \hat{x}(t+5) - x(t) \\ d\hat{x}10(t) = \hat{x}(t+10) - x(t) \\ d\hat{x}15(t) = \hat{x}(t+15) - x(t) \\ d\hat{x}20(t) = \hat{x}(t+20) - x(t) \\ d\hat{x}25(t) = \hat{x}(t+25) - x(t) \\ d\hat{x}30(t) = \hat{x}(t+30) - x(t) \end{cases} \tag{3}$$

Since radar sampling frequency is equal to 1 Hz, past delta values order 1 expressed in feet are equivalent to past speed in feet per second. Hence the estimation of speed at time t can be easily computed as de average speed during the last 10 seconds.

$$v_x = \tfrac{1}{10}\sum_{i=1}^{10} dx1(t-i) \quad \text{and} \quad v_y = \tfrac{1}{10}\sum_{i=1}^{10} dy1(t-i) \tag{4}$$

Finally the model obtains the estimation for future positions according to equation 1.

It is important to note that this physically-based model is completely general and it does not have any parameters, hence no adjustment (parameter estimation) is necessary. Although not very accurate, it is valid for all runways, all airports and all target types.

7.2 AMASS Prediction Model

A more sophisticated model can be developed in a similar way by estimating the acceleration, which requires using a simple linear regression model. AMASS uses this type of model to make its predictions. Linear prediction modeling assumes that each output sample of a signal, x(k), is a linear combination of the past n outputs, that is, it can be "linearly predicted" from these outputs, and that the coefficients are constant from sample to sample:

$$x(k) = -a(2)x(k-1) - a(3)x(k-2) - \ldots - a(n+1)x(k-n)$$

This model is better during landings and departures where acceleration values are significant and therefore assuming constant speed yields too-short predictions during take off and too-long predictions during landings. On the other hand the model based on

velocity and acceleration is worst for constant speed fly, taxi movement or stop targets, since measurement errors could be misinterpreted as large acceleration values by the model. Our test demonstrated that estimating velocity and acceleration together attains worse overall results than estimating velocity alone.

Nevertheless, both the linear regression model and constant speed model are characterized by a delay, since changes in velocity or acceleration have a small incidence in the forecast until the model is using a significant amount of date related to the new status. For our model a typical delay in a sudden change of speed is 5 to 10 seconds until estimations become accurate again. But this dynamic response can be improved by estimating current moving conditions using a weighted average of delta values in equation 3.4, so that most recent values are given more confidence. An easy way to implement this approach is using adaptive models (or self-tuning models) with forgetting factor [13] where the weights associated to past values follow an exponential profile. As it will be shown latter, using a weighted average of delta values is equivalent to an autoregressive model of speed described in the following section.

7.3 Autoregressive Prediction Model

Stochastic process theory provides the mathematical support needed for time series analysis. Since target position observations are available at discrete, equispaced intervals of time, the forecast made at origin t for values that will occur at future time $t+l$ is based on the latest available values:

$$\hat{z}(t+l) = f(z(t), z(t-1)\ldots \tag{5}$$

Since the behavior of the airplane frequently changes according to pilots actions, it is only necessary to make use of a small set of the most recent values to compute current forecast. In this sense, the system can be considered an autoregressive (AR) process of order p, where future values are obtained as a finite linear aggregate of previous values of the process and a shock a_t [12].

$$z(t) = \phi_1 z(t-1) + \phi_2 z(t-2) + \ldots + \phi_p z(t-p) + a_t \tag{6}$$

The name autoregressive comes from the fact that it has the same structure as a linear regression model where independent variables are previous values of the dependent (said to

be "regressed") variable. In this case it is equivalent to say that the velocity of a target can be expressed as a linear combination of previous values of velocity of that target.

Using training sets extracted from log files recorded at ORD on August 2001, the parameters for an AR model order 10 ($\phi_1 \ldots \phi_{10}$) were obtained using the Least Squares adjusting method. Hence the following models were obtained:

$$\begin{cases} d\hat{x}5(t) = \phi_{1,5} dx1(t-1) + \phi_{2,5} dx1(t-2) + \ldots + \phi_{10,5} dx1(t-10) \\ d\hat{x}10(t) = \phi_{1,10} dx1(t-1) + \phi_{2,10} dx1(t-2) + \ldots + \phi_{10,10} dx1(t-10) \\ d\hat{x}15(t) = \phi_{1,15} dx1(t-1) + \phi_{2,15} dx1(t-2) + \ldots + \phi_{10,15} dx1(t-10) \\ d\hat{x}20(t) = \phi_{1,20} dx1(t-1) + \phi_{2,20} dx1(t-2) + \ldots + \phi_{10,20} dx1(t-10) \\ d\hat{x}25(t) = \phi_{1,25} dx1(t-1) + \phi_{2,25} dx1(t-2) + \ldots + \phi_{10,25} dx1(t-10) \\ d\hat{x}30(t) = \phi_{1,30} dx1(t-1) + \phi_{2,30} dx1(t-2) + \ldots + \phi_{10,30} dx1(t-10) \end{cases} \quad (7)$$

As it can be seen in each line of equation 7, this model is a weighted sum of previous delta values similar to the weighted average proposed in the previous subsection, with the advantage that these weights are not exponential but automatically obtained during the adjustment process to best represent the dynamic behavior of real airplanes.

7.4 Prediction Model Comparison

During this research we have been using a version of AMASS that only supports single surface analysis. This means that the system was only able to analyze events in which both targets are located in the same runway. The most common single surface event is lander behind lander where an airplane has just landed while another one is approaching to land in the same runway. ORD, like most major airports with parallel runways, usually operates using one runway for landings and the other for departures. The interval between two consecutive take offs is always under control because the second airplane is held until the first airplane is flying at adequate distance. In fact it is mandatory to wait a minimum amount of time to avoid turbulence tracks left behind the first airplane. On the other hand, landings are less controllable because several aircraft usually approach to the airport in line over several miles so small differences in velocity may yield reduced separation distances. If one airplane that has just landed, for any reason takes some additional time to leave the runway, the airplane coming behind will no be able to hold so it will result in a dangerous situation.

In order to analyze single surface events, the best measurement that can be used to determine the level of hazard is the separation distance. In fact the version of AMASS used was design to compute all the safety logic based on current separation distance and also integrates a linear model to predict future values of separation distance. As described in Chapter 5, we use the incursion distance algorithm to predict the distance between two airplanes. Several log files were run within AMASS to compare the behavior of the models during the precise moments in which the system needs future prediction. All data was recorded at ORD on year 2001: August 10^{th}, 11^{th}, 14^{th}, 17^{th}, 19^{th}, and 21^{st}..

Incursion Distance is obtained for the three position prediction models and compared with the separation distance predicted by AMASS linear models and also with the actual separation distance. The former value is extracted from the log files reading the coordinates of both airplanes ahead in time. The results are shown in Table 10.

In order to make a more impartial comparison between models, non-single surface events were eliminated from the analysis using position prediction approach. The improvement in linear model errors is very significant; nevertheless neural network model is generally also accurate.

Table 10. Error comparison of prediction models.

time	NN MAE	NN MSE	AMASS MAE	AMASS MSE	CS MAE	CS MSE	AR MAE	AR MSE
5	156.56	234.42	158.96	603.42	190.66	263.92	186.35	281.47
10	287.46	469.45	264.18	675.41	422.34	570.43	388.27	606.59
15	448.24	741.26	412.40	709.26	716.45	954.10	679.93	1059.04
20	577.77	975.11	588.44	845.10	1008.88	1324.45	1003.15	1570.06
25	724.37	1186.32	760.28	982.04	1289.87	1676.30	1367.54	2123.54
30	863.46	1368.75	931.46	1159.96	1528.37	1965.40	1753.31	2680.00

In theory, prediction of separation distance based on previous values of separation distance should be more precise than computing separation distance after two predicted positions, because in the latter case prediction errors are added. As a consequence, the linear model that estimates separation distance directly (AMASS) usually beats other models. Nevertheless, the research team determined that real hazardous situations found in the log files usually correspond to one airplane approaching and another one on ground moving slowly. For this particular configuration the neural network model is very accurate and even better that the separation distance model.

Although the neural network performed well, we assumed that the predictions of the first 10 second of every event are contributing significantly to the error. At the beginning of each event, when the airplane has just appeared within radar range, there are not enough data points to compute an accurate prediction. Thus, it was assumed that the airplane is flying at constant speed, estimated after the first two values of position data, and repeat the current delta value to fill the input vector. However, often, the first two values are inaccurate (see example on figure 33), and this skews the error to be much worse than it should be. The following table shows the actual errors of each prediction model without the artificial generation of first 10 samples of each event.

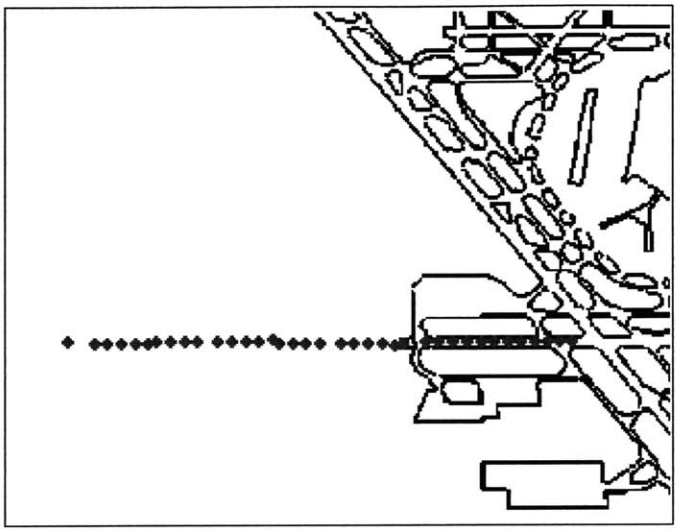

Figure 33. The first two samples, which are innaccurate make the system think that the airplane is flying faster than its real speed.

Table 11. Error comparison of prediction models excluding first 10 seconds of each event.

time	NN MAE	NN MSE	AMASS MAE	AMASS MSE	CS MAE	CS MSE	AR MAE	AR MSE
5	165.39	271.42	283.01	909.06	189.56	297.65	197.39	318.27
10	339.77	618.66	354.65	904.50	424.69	678.50	395.44	677.54
15	543.99	980.08	463.59	983.85	694.30	1087.53	643.03	1094.34
20	629.14	1146.73	537.81	1045.95	870.37	1333.35	826.79	1410.94
25	669.15	1177.53	607.86	1105.32	940.83	1423.43	1049.86	1745.28
30	760.82	1174.22	661.24	1170.82	933.14	1368.63	1254.39	2116.43

The results of all the prediction models improved significantly, demonstrating that the errors levels obtained using simulated samples at the beginning of each event, contributed notably to the overall mean error. The neural network and AMASS models attained the best results, though AMASS is getting error levels higher than the rest of the models for short predictions (5 s and 10 s in the future) especially for mean square error. Further

research is being performed in order to determine which particular events or situation are contributing gravely to the mean error level.

7.5 Alarm Analysis

Thus far, this report has detailed the design and development of the optimal neural network for aircraft position prediction. Our network was validated in a MATLAB simulation environment. In the next stage, our team integrated our changes into AMASS for a side by side comparison of the neural network and linear prediction. The goal of our work is for NN-AMASS to generate more accurate alerts than the original AMASS system. The following steps were taken to validate the performance of the new system.

- Compile all alarms generated by available log files
- Categorize alarms as real, nuisance, or multi-path
- Validate NN-AMASS performance in real alarm situations
- Compare number of nuisance alarms generated by each system
- Determine the reliable lead time for alerts in both systems.

Compilation of Alarms

The team had access to 18 log files that chronicle aircraft movement at the Chicago-O'Hare airport from August 10, 2001 to August 27, 2001. Using the original AMASS system, the team ran each log, and optimized the position prediction system to the specific landing pattern at the airport. For example, the alert system is optimized depending on which direction airplanes are landing on a runway. The team cataloged all the alerts generated. Please see the Appendix D for a list of alerts found.

Categorization of Alarms

After collecting all the alerts generated, each alarm situation was replayed to determine what situation caused the alert to be generated. We broke down the alerts into three categories. The alerts were characterized as:

- Multi-Path
- Real
- Nuisance / LAHSO

The AMASS ground surveillance system sends out radar to the area surrounding the airport. When it receives a signal back, it determines the location of a given airplane. However, if two signals for the same airplane are received, it means that one of the signals that was received by the receiver must have taken a non-direct path (i.e. it bounced off of another object.) Thus, the system sometimes creates an additional airplane on the runway from the second signal it receives. Sometimes an alarm sounds even though there is really no plane in a given position. AMASS assumes there is a plane because it received a multi-path signal. Thus, these alerts are characterized as Multi-Path. These alerts are usually created by adverse weather conditions. Operators tend to turn off the AMASS system during inclement weather.

Real alerts are alerts that are generated in dangerous situations, in which the air traffic controller has to take prompt and appropriate action to correct the situation. Nuisance alerts are alerts that sound when a Land and Hold Short Operation (LAHSO) is taking place. During these operations, airplanes come close together, sounding an alarm. However, air traffic controllers do not have to take any action on these alarms because they know that the concerned pilots have adequate control of the situation. LAHSO alerts are "nuisance" alerts which one would like the Neural Networks to minimize.
The following real and nuisance alerts were found. The rest of the alerts, not listed here, were multi-path alerts.

LAHSO (NUISANCE)

Date: 08-17-01, Time: 21:06:06, Op-Config: 27L, Safety Cell: 1.3.4.c, Runway: 22R.
Date: 08-18-01, Time: 0:44:01, Op-Config: 22R, Safety Cell: 1.3.4.c, Runway: 22R.

REAL

Date: 08-21-01, Time: 02:35:41, Op-Config: 9R, Safety Cell: 6.4.3.c, Runway TWY F, Situation: Occupied Runway.
Date: 08-21-01, Time: 02:35:36, Op-Config: 9R, Safety Cell: 1.5.4.c, Runway 9R, Situation: Go-Around.

Validation of NN-AMASS in Real Alarm Situations

The NN-AMASS system performed better than the unmodified AMASS system in the real alert condition on August 21st. On runway 9R, there is an airplane taxiing on the runway, and another airplane proceeds to land on the same runway. Both AMASS systems generate an alarm.

The chart below indicates the lead time before the near-collision. Note that using NN-AMASS, air traffic controllers have additional lead time in which to prevent dangerous situations.

	20s Safety Cell	25s Safety Cell	30s Safety Cell
Linear Prediction AMASS	22 seconds	30 seconds	37 seconds
NN-AMASS	4 seconds	31 seconds	41 seconds
Additional lead time using NN-AMASS v. LP-AMASS	2 seconds	1 seconds	4 seconds

NN-AMASS and LAHSO Alerts

NN-AMASS was able to reduce the number of nuisance alerts in LAHSO situations. Notice that using either the 20s, 25s, or 30s safety cell, the NN-AMASS generates fewer alarms than the Linear Prediction AMASS. Thus, air traffic controllers will not be as distracted by nuisance alerts, and they will have more confidence in real AMASS alerts.

		20s Safety Cell	25s Safety Cell	30s Safety Cell
Linear Prediction AMASS	17-Aug	No Alarm	No Alarm	**Alarm**
	18-Aug	No Alarm	**Alarm**	**Alarm**
NN-AMASS	17-Aug	No Alarm	No Alarm	No Alarm
	18-Aug	No Alarm	No Alarm	**Alarm**

NN-AMASS and Multi-Path Alerts

19 multipath alerts were generated on August 19, due to a inclement weather. Most of the alerts were generated in a very short period of time. In 4 of the 19 alerts, the prediction algorithm was called before sounding an alarm. However, NN-AMASS and LP-AMASS both sounded an alarm in each of these four cases.

Reliable Lead Time with NN-AMASS

Currently, LP-AMASS uses the 20 second safety cell to generate alerts. Since NN-AMASS is more accurate than LP-AMASS, we can potentially use the 30 second look ahead to generate earlier warnings. The amount of lead time in a real alarm situation would increase by at least 10 seconds. Future work would include computing the error of every estimator (NN+5, NN+10 ... NN+30, and Linear+5, Linear+10,...Linear+30) for alarm situations. Once the team determines that using the 30 second estimator does not increase the error, we can recommend changing the safety cell parameter in NN-AMASS.

8. CONCLUSIONS

Air traffic is increasing worldwide every year, so air traffic controllers need to manage airports more efficiently. This is specially true under low visibility conditions when it is required to cancel most of the flights to ensure larger separation distances. New operations such as LAHSO, or allowing a second departure to get ready for take off as soon as the first aircraft begins take off, or allowing landers to flight closer to each other, are measures that may increase airport efficiency drastically. Nevertheless new tools are needed in order to assure high safety levels, and there is a request to improve safety even higher than the current level.

A Surveillance system implementing the strategies and forecasting capabilities described in this paper, is one of the tools that better may increase safety for commercial aviation. Forecasting models have been presented to predict future position of aircraft and ground vehicles around the airport. These models allow for automated monitoring of all movements, ensuring that all operations are performed correctly, including LAHSO. It is also remarkable the ability to detect threatening runway incursions if these modules for position prediction are integrated in a surveillance system. All models were trained and tested using read data acquired at O'Hare International Airport (Chicago, IL) during August 2001. For the forecasting approaches implemented, neural network outperformed the others. The average estimation error for 20 seconds look ahead is about 200 ft, which is the size of a large commercial airplane.

Large arrays of neural networks were trained using different training data, methodology, and architecture, all contributing to the enhancement of the neural network model in an incremental manner. In the end, one neural network model specifically trained for the Chicago O'Hare Airport was optimized and integrated into the AMASS to analyze it's the overall efficacy and performance of the NN enhanced version of AMASS.

The results were very promising as the neural network system yielded improved accuracy of predicting the future positions of individual airplanes, as well as the separation distances between matched airplanes. The accuracy of the predicted values allows two major benefits: first, real alarms can be triggered earlier than is possible with the current version of AMASS; and second, the probability for the generation of nuisance alerts can be

significantly reduced. Furthermore, being able to accurately predict position prediction allows the system to detect dangerous situations involving airplanes on intersecting runways, LAHSO events, and runway incursions. The separation distance approach implemented in AMASS is properly able to detect dangerous lander behind lander situations, but it does not analyze intersecting runways or LAHSO events, and it has strong limitations for incursion detection.

The availability of accurate historical information is a major contributor to the generation of early and accurate projection times with the assistance of the neural networks. In other words, a significant portion of the success of neural networks can be attributed directly to the quality of the data that are used to train the network. These data encompass actions taken by the controller and the pilot. Because of their ability to effectively capture the essence of such decentralized decision making, the option of using neural networks to model complex environments appears very attractive. Neural networks are capable of learning the operational environment that exists at the airport and its immediate vicinity. Through such learning, neural networks can provide projections earlier, thereby enabling controllers to have more decision time in which to take corrective action.

9. FUTURE WORK

Enhancement of Integrated System

The standard version of AMASS calculates and then stores the separation distance between two airplanes and uses a linear prediction algorithm to estimate future separation distances. Using purely classical laws of motion, it attempts to identify situations where two aircrafts may come close to each other.

The above technique attaches no importance to decisions likely to be made by the controller on the ground and by the pilots of the two airplanes. At each airport, there is a very definite landing pattern which influences decisions that change the speed and direction of each aircraft.

The neural networks designed by the project team are able to "learn" the pattern of such decisions purely from earlier flights in the vicinity of the Chicago O Hare airport. The trained model performed extremely well on data related to other flights in the Chicago area. One now needs to build upon the capabilities of the neural network that has been integrated into the AMASS system. This process will involve testing the efficiency of the integrated systems using extended amounts of data from various airports across the country. It will further involve detailed examination of real and potential alert scenarios from such airports. The identification of a subset of data related to alert conditions is a manually tedious process; efforts will be made to see if part of this task could be automated.

The goal will be to design and develop techniques that would facilitate faster and easier customization of the integrated system to characteristics of additional sets of airports.

Multi-surface Analysis

Currently AMASS is implemented to predict potentially adverse incidents in one surface, meaning that both airplanes are located in the same runway. But many situations involving incursions take place when two airplanes are moving on different tracks towards a common point. These cases can be identified by carefully analyzing the distance between the projected trajectories of every relevant pair of planes. One example is the case of a runway incursion involving an airplane from a taxiway while another airplane is using the runway for takeoff. In this situation, the current AMASS system does not make any estimate of the relative movement of the airplanes until the taxi airplane enters the runway and the event is

considered to be a one surface situation. At that point an alarm is reported if the separation distance is less than a stipulated distance or if the estimated separation distance at a certain point of time in the future is less than that stipulated distance.

Taking into account that the length of the runway is usually smaller than these values, AMASS shows the alarm as soon as both airplanes are considered to be in the same surface. Therefore AMASS does not provide any alarm until the taxi airplane reaches the runway, and at that point no estimations are needed. The incursion is detected as soon as a taxi aircraft approaches a runway or an intersection classified as "in-use". Figure 34 illustrates a near collision incident that occurred on April 1st, 1999 at ORD involving two 747s; here the red dot depicts an airplane taking off on runway 14R and the blue dot is another airplane entering the same runway. The alarm is detected by AMASS just 5 seconds before the potential encounter. On the other hand, Figure 35 depicts the proposed approach. Here the big dots that represent real airplane positions and small dots represent position predictions; in this case the alarm is given earlier, possibly when the airplane shown in blue has entered the runway (25 seconds before the potential encounter).

In contrast with linear estimators, models based on NN will be able to make proper estimation of the future position in a variety of situations: acceleration, deceleration and taxi. Therefore it should be possible to detect runway incursions, potentially dangerous events involving intersecting runways, and LAHSO and multi-surface scenarios.

Another important characteristic of a general approach based on position prediction is that it will require less tuning of the system for its use on a new airport. Currently, AMASS requires tuning of the safety cells and associated parameters. Although some parameters come with default settings, several months of effort is involved in fine-tuning the system before it is declared operational. Using the position prediction approach, one will only need to provide the definition of runways coordinates and intersecting areas.

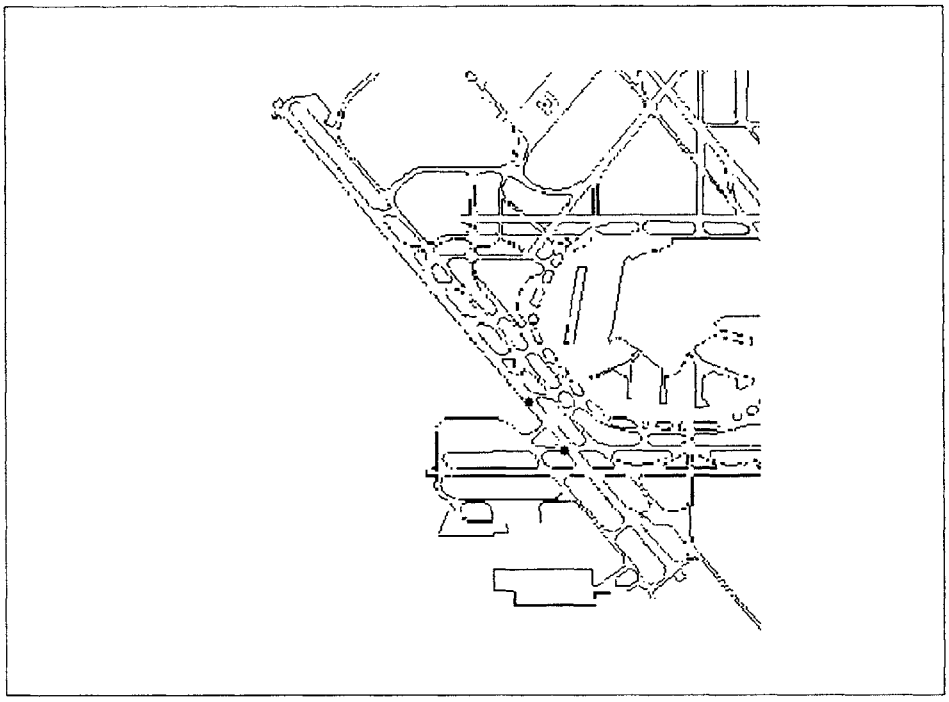

Figure 34: Detection of incursion without prediction

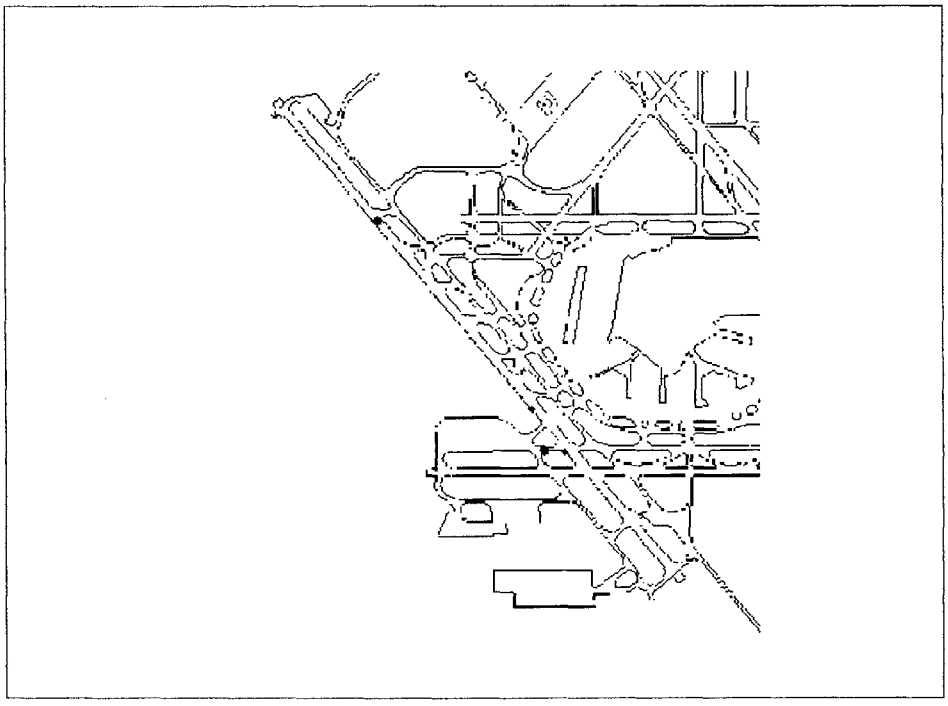

Figure 35: Detection of incursion with position prediction

Training of Neural Network for Multiple Airports

Currently, optimized configuration file has been developed for Chicago O'Hare airport. The neural network performances at other airports have not been investigated. Two possible approaches for evaluating neural network capability are as follows:

- Create a unified neural network configuration that caters to the operational characteristics for multiple airports.
- Develop neural networks specifically trained for each airport and the unique characteristics of each airport.

In order to investigate the feasibility of a unified neural network, one will need to take an NN designed for one airport and then validate and optimize this NN for other airports. Initial results were obtained using net_001 configuration file at the San Diego (SAN) airport.

SAN20011826	Neural Network (ft)	Linear Prediction (ft)	% Improvement
Mean Absolute Error	260.07	284.23	8.50%
Sq. Mean Square Error	411.5	346.92	-18.62%

In the case of San Diego, neural network errors and the linear model errors are similar, and in many cases neural network errors are higher than the linear model. This is because the neural network had been trained using data from ORD. This shows that the particular characteristics of the airport play a very major role in determining the degree of relevance for a model developed using data from another airport.

Developing a unified neural network could be potentially accomplished by training NN with data obtained from airports with dissimilar characteristics. Accordingly, one would first need to classify airports into different categories based on runway configuration, volume of flights, weather conditions and other parameters. Next, one would need to gather adequate amounts of training and validation data, identifying all possible events that may occur in the relevant set of airports. Data mining techniques could be employed to extract and classify events after sustained training and validation. However, this will be a very challenging task given the wide disparity in the characteristics of the different airports and the broad variations in the other underlying parameters.

On the other hand, training a neural network specifically for a certain airport will be relatively simple and the tools required for extracting data and training of neural network have already been developed to a significant extent.

Additionally, the current version of NN-AMASS has been designed to accept change in the neural network configuration (weights and biases). Accordingly, the strategy for future work envisages a hybrid approach in which there is separate sets of neural network configuration files (NN.txt) for each airport and one will attempt to incorporate the NN.txt for the corresponding airport. This will allow each airport to be modeled independently to give full weightage to the characteristics of each airport. At the same time, this approach provides the basis of utilizing a modular approach so that the data mining and knowledge discovery model developed for one airport can form the foundation for new training endeavors for the next airport.

Intersecting Runways

Only one airport with intersecting runways (ORD) has been tested. Future training and testing on NN models would need to focus on more multiple runway operations in order to learn the characteristics of the relevant multi-surface situations. The separation distance between a stopped object and a moving object or between two objects moving at different speeds in different directions possesses non-linear characteristics. Most situations analyzed so far involve airplanes in three cases: both at low speed or both moving in the same direction, or one stationary and the other at high speed but with a large separation distance between them. These situations are essentially linear in nature.

More testing is necessary to determine if separation distance is a good measure to prevent incursions and near-collisions if intersecting runways are being used simultaneously.

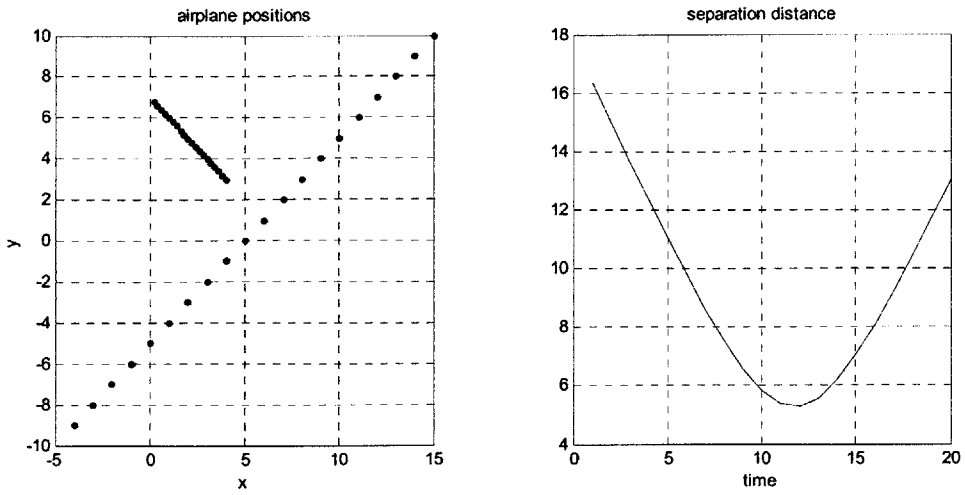

Figure 36: Non-linear behavior of separation distance for airplanes moving in different direction and at different speed.

Data Enhancement Endeavor

While running the NN models one discovers a number of issues related to the quality of the data available in AMASS. Based on the originating radar equipment and the preprocessing that is performed on such data, one finds instances of missing and conflicting data. The instances of missing data range from missing single values to blank records extending over periods of several seconds, even minutes. Such instances currently require manual intervention. Significant potential exists for design and development of automated techniques that can identify erroneous data and address the situation using heuristic techniques. The use of such techniques will enable the implementation of more accurate alert mechanisms.

Airport Capacity Enhancement

The capacity for handling incoming and outgoing flights at airports is heavily dependent on the designated separation distances and separation times. The magnitude of these parameters has been gradually reduced over the last fifty years. With the advent of new approaches for predicting separation distances with greater accuracy, as well as for generating alert conditions earlier than before, one can examine the possibilities for reducing the minimum acceptable separation distances and separation times. Such changes will allow airports to handle more flights while still preserving, or even enhancing the aspect of air safety.

Integration with evolving 3D techniques

FAA is investigating new technologies which give greater consideration to the collection and potential use of data related to the height of the airplanes. Just as GPS techniques allow pinpointing an object with very high accuracy, new technologies being gradually adopted by FAA will yield huge volumes of multilateration data. Neural Networks based data mining techniques offer great capability to analyze very large amounts of information and to gain unprecedented insights into such data. As these 3D oriented techniques progress further with respect to deployment at selected airports of the country, it would be pertinent to design and implement the data analysis components in parallel.

10. REFERENCES

[1]. R. Heitmeyer, "Biometric ID and MRTDs Will Help Generate Future Airport Capacity", *Int'l Airport Rev.*, vol. 5, no. 1, 2001, pp. 60–63.

[2] National Transportation Safety Board, "Most wanted transportation safety improvements 2004-2005", http://www.ntsb.gov/recs/brochures/MostWanted_2004_05.pdf, November 2004.

[3] Jim Upton. *Boeing 777. Airliner Tech Series Vol.2.* Specialty Press Publishers and Wholesales. ISBN: 1-58007-001-9, 1998.

[4] Piazza E., "Increasing airport efficiency: Injecting new technology", *IEEE Intelligent systems*, vol 17, Issue 3, pp. 10-13, May-Jun 2002.

[5] Cassell, R.; Evers, C.; Sleep, B.; Esche, J.; "Initial test results of PathProx - a runway incursion alerting system", *The 20th Conference on Digital Avionics Systems, DASC 2001.* Volume: 1, Pages:2D3/1 - 2D3/7, 14-18 Oct 2001.

[6] Cassell, R.; Evers, C.; Esche, J.;"Safety benefits of PathProx-a runway incursion alerting system", *The 22nd Conference on Digital Avionics Systems, DASC 2003.* Volume: 2, Pages:9.B.4 - 91-10, 12-16 Oct 2003.

[7] J. Griffith, *Order 7110.118: Land and hold short operation*, U.S. Department of Transportation, Federal Aviation Administration. Jul 2000.

[8] Federal Aviation Administration, and ASA Staff, *FAR/AIM 2004: Federal Aviation Regulations/ Aeronautical Information Manual*, Aviation Supplies & Academics, ISBN: 1-56027-499-9. 2003.

[9] National Transportation Safety Board, "Safety Recommendation to the Federal Aviation Administration to Prevent Runway Incursions", http://www.ntsb.gov/pressrel/2000/000613.htm, June 13, 2000

[10] FAA Office of Runway Safety, "FAA Runway Safety Report 1999-2002", Federal Aviation Administration. July 2003

[11] CNN news, "Congress gets critical report on airport runway safety", http://edition.cnn.com/2000/US/03/22/runway.safety.02/, March 23, 2000.

[12] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, *Time Series Analysis. Forecasting and Control.* Third Edition. Prentice Hall, NJ, USA, ISBN 0-13-060774-6, 1994.

[13] P.E. Wellstead, M.B. Zarrop, *Self-tuning systems. Control and signal processing.* John Wiley & Sons, NY, USA. ISBN 0-471-93054-7, 1991

[14] G. Bisignani, "IATA Annual Report 2004", International Air Transportation Association, June 2004.

[15] M.A. Stamatopoulos, K.G. Zografos, A.R. Odoni, "A decision support system for airport strategic planning", Transportation Research Part C, vol. 12, pp. 91-117, 2004.

[16] Farina, A.; Ferranti, L.; Golino, G.; "Constrained tracking filters for A-SMGCS", Proceedings of the Sixth International Conference of Information Fusion, Volume: 1, Pages:414 - 421, July 8-11 2003

[17] Garcia, J.; Berlanga, A.; Molina, J.M.; Besada, J.A.; Casar, J.R.; "Planning techniques for airport ground operations", Proceedings. The 21st Digital Avionics Systems Conference, Volume: 1, Pages:1D5-1 - 1D5-12, 2002.

[18] Neural Network-based Modeling and Simulation for the Optimization of AMASS Safety Logic, Research Project Report for Federal Aviation Administration AND-410, November 30, 2003

[19] Cooney, G.T, "Optimizing Neural Networks for Enhancing Air Traffic Safety", MIT MS thesis, February 2004.

[20] Nakanishi, Y. "Optimal Neural Network for Airport Movement Area Safety System", MIT Advanced Undergraduate Project, December 2003

11. APPENDIX

11.1 Appendix A: Definitions and Abbreviations

Multiple Layer Perceptron (MLP)
Definition: An MLP can be thought of as a cluster of many SLPs. Unlike SLPs, an MLP can emulate any real continuous function. MLPs consist of a set of source nodes that constitute the input layer, one or more hidden layers of computation nodes, and an output layer. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. MLPs have been applied successfully to solve difficult problems by training them in a supervised manner with the popular back-propagation algorithm. An MLP has two distinguishing characteristics from SLPs. First, all its nodes or neurons utilize a nonlinear activation function. The presence of non-linearity enables MLPs to map the input to the output using any real function. Second, the hidden layer or layers enable the network to learn complex tasks.

Separation Distance (SD)
Separation distance is the distance separating two consecutive aircrafts. (Note that the separation distance does not discriminate between the states of the aircraft, for example, landing, departing, taxi or stopped.)

Separation Velocity (SV)
Separation velocity is the velocity difference between two consecutive aircrafts.

Landing-Departure (L-D) Condition
The landing-departure case is a pair of successive flights landing on and departing from the same runway.

Landing-Landing (L-L) Condition
The landing-landing case is a pair of successive flights landing on the same runway.

LAHSO (Land and hold short operation)
A LAHSO operation is used when it is necessary to have aircraft cross a runway while another aircraft is landing on that same runway. The pilot is responsible for holding short of the intersection used for crossing traffic. In Figure A1, the line labeled "Separation Distance" refers to the value that the neural network will predict. This separation distance differs from the separation distance for intersecting runways in AMASS safety logic. AMASS projects a distance to the intersection for each aircraft.

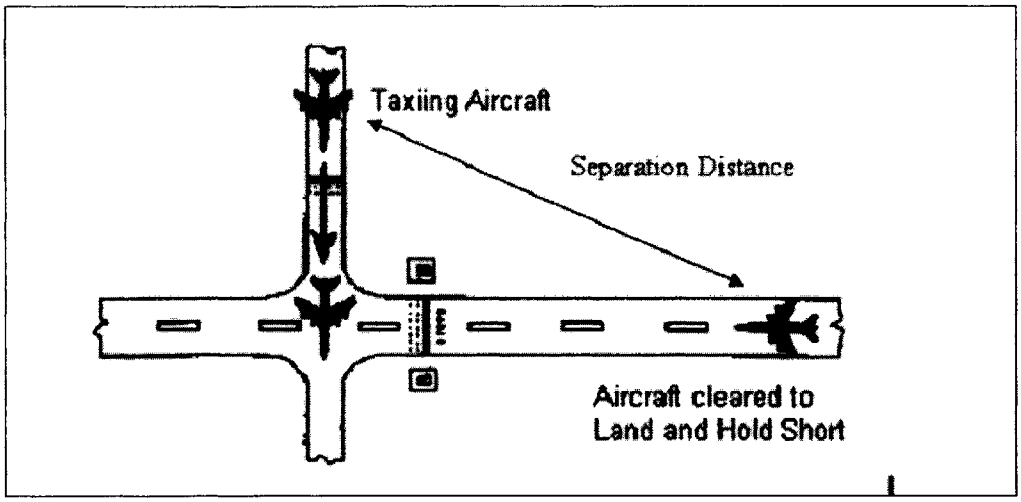

Figure A1 - Land and Hold Short Operation (Source: http://www.aopa.org/images/asf/pubs/lahso/lahsofig2.gif)

11.2 Appendix B: Incursion Distance Calculation script

```
function [IncDist]=calcIncDist(x1,y1,x2,y2,dx1,dy1,dx2,dy2)
%
% [IncDist]=calcIncDist(x1,y1,x2,y2,dx1,dy1,dx2,dy2)
%
% Calculate Incursion Distance by projecting CurrentSD and Delta vector over the most
likely runway.
%
% This function is valid even if dx==0 and dy==0 simultaneously, therefore it can also be
used to
%   evaluate LAHSO events.
%
% OMEGA...  rounded values (runways directions)
% omega...  real values
%
% Rafael Palacios Nov/2003, May/2003
%
% May/2004: Added more flexible mathing (+/-10 degree) between runway direction based on
current positions
%    and runway direction based on fastest target.
%

%Modules (in general d will be a one-row matrix)
d(:,1)=sqrt(dx1.^2+dy1.^2);
d(:,2)=sqrt(dx2.^2+dy2.^2);
CurrentSD=sqrt((x1-x2).^2+(y1-y2).^2);


%angles
alpha(:,1)=atan2(dy1,dx1);
alpha(:,2)=atan2(dy2,dx2);
alphaSD=atan2(y1-y2,x1-x2);

%runway detection, base on first row, then used for all rows (in general d will be a one-
row matrix)
if d(1,1)>d(1,2)
  fastest=1;
else
  fastest=2;
end
OMEGAdeg=round(alpha(1,fastest)*180/pi/10);
if OMEGAdeg==-18, OMEGAdeg=18; end
OMEGArad=OMEGAdeg*10*pi/180;

alphaSDround=round(alphaSD(1)*180/pi/10);
if alphaSDround==-18, alphaSDround=18; end

%Cheking direction
if alphaSDround==OMEGAdeg  %ok, target1 is behind target2
  T1=1;
  T2=2;
elseif alphaSDround+18==OMEGAdeg | alphaSDround-18==OMEGAdeg % ups, change target1 and
target2
  T1=2;
  T2=1;
  alphaSD=alphaSD+pi;
else
  disp('WARNING, runway direction contradiction.');
  %I will use alpahSD direction instead of OMEGA direction if discrepancy is within 10
degrees
  %  since fastest airplane may have errors in direction due to incorrect estimation
  if alphaSDround==OMEGAdeg | alphaSDround==OMEGAdeg-1 | -alphaSDround==OMEGAdeg-1 |
alphaSDround==OMEGAdeg+1
    T1=1;
    T2=2;
    disp('CORRECTED, runway direction contradiction.' );
  elseif alphaSDround+18==OMEGAdeg | alphaSDround-18==OMEGAdeg ...
       | alphaSDround+17==OMEGAdeg | alphaSDround-17==OMEGAdeg ...
       | alphaSDround+19==OMEGAdeg | alphaSDround-19==OMEGAdeg
    T1=2;
    T2=1;
    alphaSD=alphaSD+pi;
    alphaSDround=alphaSDround+18;
    disp('CORRECTED, runway direction contradiction.' );
  else
    disp('ERROR, runway direction contradiction. Unable to correct the problem');
    IncDist=NaN;
    return
  end
```

```
  OMEGAdeg=alphaSDround;
  OMEGArad=OMEGAdeg*10*pi/180;
end

%Display runway direction
runway=9-OMEGAdeg;
if runway<0
  runway=runway+36;
end
%%%%%%%%%%%%%%%%%fprintf('Runway: %02d\n',runway);

%omega angles
omegaSD=alphaSD-OMEGArad;
omega=alpha-OMEGArad;

%Incursion Distance for any category
IncDist=CurrentSD.*cos(omegaSD) + d(:,T1).*cos(omega(:,T1)) - d(:,T2).*cos(omega(:,T2));
```

11.3 Appendix C: sd_analysis script

The following MATLAB script was written for the NN model, and similar scripts were written for the CS, AR models.

```
function values = sd_analysis_newNN(nnoutput, netASR, netASDE)
%
%
% Computes a matrix of SD values to compare performance of models
% function values = sd_analysis_newNN(nnoutput, netASR, netASDE)
%
% nnoutput: name of the NNoutput file which results from running the log file in AMASS. (in
XLS format)
% netASR, netASDE:  Neural Network to be used instead of using NN prediction from the
nnoutput file.
%
% values:   Matrix containing coordinates and the NN prediction in the first columns,
%           AMASS predicted incursion distances in the second set of columns, and
%           real separation distances in the last set of columns
%           [x1 y1 x2 y2
%              18 empty columns
%              newNN05, newNN10,...newNN30]
%
% example:
% >load net20040604
% >values = sd_analysis_newNN('NNOUTPUT810.xls',net.ASRnet, net.ASDEnet);
%
% Anuja Doshi, Rafael Palacios
% Version 1.0, abr 15, 2004
% Version 2.1, may 02, 2004 accuracy_anal
% Version 3.2, may 15, 2004 accuracy_anal
% Version 4.0, may 25, 2004 accuracy_anal2
% Version 5.0, jun 09, 2004 all together: NN, AMASS, real, newNN, cte, lin
% Version 6.0, ago 06, 2004 change name to sd_analysis.  Separete code by models
% Version 7.0, dec 10, 2004 remove first 10 samples of each event

%Load XLS file
fprintf('Loading XLS...')
[a, text]=xlsread(nnoutput);  % a stores numbers, text stores strings
a=a(2:end,:); %remove header
text=text(2:end,:); %remove header
%leading text columns are not added by xlsread
%this is confusing, so lets add two columns of NaN values at the begining and at the end of
a.
a=[ones(size(a,1),2)*NaN,a,ones(size(a,1),2)*NaN];
%now a and text have the same size

%checking
%size(a)
%size(text)
fprintf('Loaded. %d samples\n',size(a,1))

% Remove the taxi-taxi, and taxi-stop events from the nnoutput file.
fprintf('Filtering Taxi...')
L = length(text(:,1));
TT = 'Tax-Tax';
ST = 'Stp-Tax';
TS = 'Tax-Stp';
filt_nn = [];

for i=1:L
    if (strcmp(text(i,9), TT) | strcmp(text(i,9), ST) | strcmp(text(i,9), TS))
        %do nothing
    else
        filt_nn = [filt_nn; a(i,:)];
    end
end
fprintf('Filtered. %d samples remaining\n',size(filt_nn,1))

%   Create a new matrix, 'values', with the following format
%   values = [x1, y1, x2, y2, ID5..ID30, SD5...SD30, realSD5..realSD30];
%
%   SD =  Distance, computed by AMASS linear models
%   ID = Incursion Distance, computed by Matlab using NN estimations found in NNoutput
%   realSD = actual values computed using x1,y1,x2,y2
%
%  For each set of NN-predicted positions, calculate the correct incursion distance, and
```

```
%  add those values to the matrix 'values'. (Currently, NNoutput doesn't contain the
correct
%  calculation of incursion distance. 4/04); Then add the SD5...SD30 values to values.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Basic Coordinates
%values = filt_nn(:,10:13); %columns 10 - 13 are 'CurrentT1X CurrentT1Y CurrentT2X
CurrentT2Y'
L=length(filt_nn(:,1));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% newNN Incursion Distance
c=0; %counts the number of samples of a given event
n=1; %determines what row of the "values" matrix to add to
event_id = filt_nn(1,5:8); %sets the first event_id variable, which includes the PID and
TID of both planes.

for i=1:L
    if filt_nn(i,5:8) == event_id
        c = c + 1;
        if c > 10
            %fprintf('\nIncusrion Distance: Sample %d of %d\n',i,L)

            %Use appropriate NN (ASDE or ASR)
            if strcmp(text(i,end-1),'ASDE')
                deltaX1=sim(netASDE,filt_nn(i,14:2:33)');
                deltaY1=sim(netASDE,filt_nn(i,15:2:33)');
            else
                deltaX1=sim(netASR,filt_nn(i,14:2:33)');
                deltaY1=sim(netASR,filt_nn(i,15:2:33)');
            end

            %Use appropriate NN (ASDE or ASR)
            if strcmp(text(i,end),'ASDE')
                deltaX2=sim(netASDE,filt_nn(i,34:2:53)');
                deltaY2=sim(netASDE,filt_nn(i,35:2:53)');
            else
                deltaX2=sim(netASR,filt_nn(i,34:2:53)');
                deltaY2=sim(netASR,filt_nn(i,35:2:53)');
            end

            % calcIncDist(x1,y1,x2,y2,dx1,dy1,dx2,dy2)
            %ID5

values(n,23)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(1), deltaY1(1), deltaX2(1), deltaY2(1) );
            %ID10

values(n,24)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(2), deltaY1(2), deltaX2(2), deltaY2(2) );
            %ID15

values(n,25)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(3), deltaY1(3), deltaX2(3), deltaY2(3) );
            %ID20

values(n,26)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(4), deltaY1(4), deltaX2(4), deltaY2(4) );
            %ID25

values(n,27)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(5), deltaY1(5), deltaX2(5), deltaY2(5) );
            %ID30

values(n,28)=calcIncDist(filt_nn(i,10),filt_nn(i,11),filt_nn(i,12),filt_nn(i,13), ...
                deltaX1(6), deltaY1(6), deltaX2(6), deltaY2(6) );
            n = n+1;
        else
            %fprintf('\nSample %d of event %d thrown out.', c, event_id(1,1))
        end
    else
        c = 1;
        event_id = filt_nn(i,5:8);
        %fprintf('\nSample %d of event %d thrown out.', c, event_id(1,1))
    end
end
```

11.4 Appendix D: comparison script

```
function error=comparison(values)
%
% function error=comparison(values)
%
% computes a matrix to compare models accuracy while being used for SD
% Argument: values (n x 40) matrix built with accuracy_anal_all function
% Columns in values:
%   1-4: x1, y1, x2, y2,
%   5-10: AMASS-NN (Geoffs version)
%  11-16: AMASS linear
%  17-22: Actual values (looking ahead in the log file) <-- reference
%  23-28: New NN model (Rafael version)
%  29-34: CS model (physically base constant speed)
%  35-40: AR model
%
% Rafael Palacios Aug/2004
%
col=[17,5,11,23,29,35];
pred(:,:,1)=values(:,col);   %5s
pred(:,:,2)=values(:,col+1); %10s
pred(:,:,3)=values(:,col+2); %15s
pred(:,:,4)=values(:,col+3); %20s
pred(:,:,5)=values(:,col+4); %25s
pred(:,:,6)=values(:,col+5); %30s
%
for i=1:6  %5s, 10s, 15s, 20s, 25s, 30s
  %ANASS-NN
  error(i,1)=mmae(pred(:,1,i),pred(:,2,i));
  error(i,2)=mmse(pred(:,1,i),pred(:,2,i));
  error(i,3)=mmax(pred(:,1,i),pred(:,2,i));
  %AMASS
  error(i,4)=mmae(pred(:,1,i),pred(:,3,i));
  error(i,5)=mmse(pred(:,1,i),pred(:,3,i));
  error(i,6)=mmax(pred(:,1,i),pred(:,3,i));
    %newNN
  error(i,7)=mmae(pred(:,1,i),pred(:,4,i));
  error(i,8)=mmse(pred(:,1,i),pred(:,4,i));
  error(i,9)=mmax(pred(:,1,i),pred(:,4,i));
    %CS
  error(i,10)=mmae(pred(:,1,i),pred(:,5,i));
  error(i,11)=mmse(pred(:,1,i),pred(:,5,i));
  error(i,12)=mmax(pred(:,1,i),pred(:,5,i));
    %AR
  error(i,13)=mmae(pred(:,1,i),pred(:,6,i));
  error(i,14)=mmse(pred(:,1,i),pred(:,6,i));
  error(i,15)=mmax(pred(:,1,i),pred(:,6,i));
end

%print results on screen
fprintf('%8s %8s %8s   %8s %8s %8s   %8s %8s %8s   %8s %8s %8s   %8s %8s
%8s\n','NN','NN','NN','AMASS','AMASS','AMASS','newNN','newNN','newNN','CS','CS','CS','AR','
AR','AR');
fprintf('%8s %8s %8s   %8s %8s %8s   %8s %8s %8s   %8s %8s %8s   %8s %8s
%8s\n','MAE','MSE','max','MAE','MSE','max','MAE','MSE','max','MAE','MSE','max','MAE','MSE',
'max');
for i=1:6
   fprintf('%8.2f %8.2f %8.2f   %8.2f %8.2f %8.2f   %8.2f %8.2f %8.2f   %8.2f %8.2f %8.2f
%8.2f %8.2f %8.2f\n',error(i,:));
end
```

11.5 Appendix D: List of all alerts generated in AMASS

Key: R- Real, N- Nuisance, MP- Multi-Path, X- Not reproducible, NED- Real, but not enough data.

DATE	Type	Alert Information
August 10, 2001		No Alerts
August 11, 2001		No Alerts
August 12, 2001		No Alerts
August 13, 2001		No Alerts
August 14, 2001		No Alerts
August 15, 2001		No Alerts
August 16, 2001		No Alerts
August 17, 2001	**N**	1: 1.3.4.c, W: 3032, LDG, RWY 22R, OCCUPIED Fri Aug 17 21:06:06 2001
	X	2: 1.1.6.b, W: DEP, RWY 32L, OCCUPIED Fri Aug 17 03:28:38 2001
August 18. 2001	**N**	1: 1.3.4.c, W: 5324, LDG, RWY 22R, OCCUPIED Sat Aug 18 00:44:01 2001
August 19, 2001	**MP**	1: 1.5.4.d, W: 7334, GO-AROUND, RWY 09R, OCCUPIED Sun Aug 19 16:31:15 2001
	MP	2: 1.5.1.d, W: 7334, GO-AROUND, RWY 09R, OCCUPIED Sun Aug 19 16:31:03 2001
	MP	3: 6.4.3.c, W: 2173, LDG, RWY 09R, OCCUPIED Sun Aug 19 11:35:00 2001
	MP	4: 6.4.1.g, W: DEP, RWY 32L, OCCUPIED Sun Aug 19 11:32:51 2001
	MP	5: 6.4.1.d, W: DEP, RWY 22R, OCCUPIED Sun Aug 19 11:32:35 2001
	MP	6: 6.4.1.c, W: DEP, RWY 32L, OCCUPIED Sun Aug 19 11:11:26 2001
	MP	7: 6.4.1.g, W: DEP, RWY 32L, OCCUPIED Sun Aug 19 11:11:24 2001
	MP	8: 6.4.5.c, W: 6617, GO-AROUND, RWY 27L, OCCUPIED Sun Aug 19 04:54:31 2001
	MP	9: 6.4.5.g, W: 6617, GO-AROUND, RWY 27L, OCCUPIED Sun Aug 19 04:54:30 2001
	MP	10: 6.4.5.c, W: 6617, GO-AROUND, RWY 27L, OCCUPIED Sun Aug 19 04:54:29 2001
	MP	11: 6.4.3.c, W: LDG, RWY TWY M4, OCCUPIED Sun Aug 19 04:47:19 2001
	MP	12: 1.3.4.d, W: 6274, LDG, RWY 27L, OCCUPIED Sun Aug 19 04:47:15 2001
	MP	13: 1.5.4.d, W: 6274, GO-AROUND, RWY 27L, OCCUPIED Sun Aug 19 04:47:14 2001
	MP	14: 1.5.4.c, W: 6274, GO-AROUND, RWY 27L, OCCUPIED Sun Aug 19 04:47:12 2001
August 20. 2001	**MP**	1: 1.1.4.c, W: DEP, RWY 32L, OCCUPIED Mon Aug 20 04:14:59 2001
August 21, 2001	**R**	1: 6.4.3.c, W: 2763, LDG, RWY TWY F, OCCUPIED Tue Aug 21 02:35:41 2001
	R	2: 1.5.4.c, W: 2171, GO-AROUND, RWY 09R, OCCUPIED Tue Aug 21 02:35:36 2001
August 22, 2001	**MP**	1: 1.1.4.c, W: DEP, RWY 09R, OCCUPIED Wed Aug 22 22:41:17 2001
	MP	2: 6.4.3.d, W: 7076, LDG, RWY 22R, OCCUPIED Wed Aug 22 22:19:29 2001
August 23, 2001		No Alerts
August 24, 2001	**MP**	1: 6.4.1.c, W: DEP, RWY 32L, OCCUPIED Fri Aug 24 03:32:41 2001
August 25, 2001	**MP**	1: 1.1.1.h, W: DEP, RWY 09L, OCCUPIED Sat Aug 25 21:00:45 2001
	MP	2: 6.4.1.c, W: DEP, RWY 04L, OCCUPIED Sat Aug 25 02:54:16 2001
	MP	3: 1.1.6.b, W: DEP, RWY 09L, OCCUPIED Sat Aug 25 02:52:35 2001
	MP	4: 1.1.4.c, W: DEP, RWY 09L, OCCUPIED Sat Aug 25 02:52:30 2001
August 26, 2001	**NED**	1: 1.5.4.d, W: 5366, GO-AROUND, RWY 09L, OCCUPIED Sun Aug 26 20:48:20 2001
August 27, 2001		No Alerts