

RICE UNIVERSITY

**A Computational Model of Jetliner Taxiing**

by

**Jeffrey C. Zemla**

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR  
THE DEGREE

**Master of Arts**

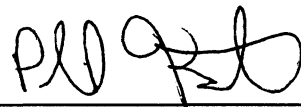
APPROVED, THESIS COMMITTEE:



Michael D. Byrne  
Associate Professor, Psychology



David M. Lane  
Associate Professor, Psychology,  
Statistics, and Management



Philip Kortum  
Professor-in-the-Practice, Faculty  
Fellow, Psychology

HOUSTON, TEXAS  
JANUARY 2012

## Abstract

### A Computational Model of Jetliner Taxiing

by

Jeffrey C. Zemla

The Next Generation Air Transportation System (NextGen) is transforming the way planes move on the ground as well as in the sky. Some of the proposed changes, such as automated scheduling algorithms to generate taxi clearances and speed-based taxi clearances, require thorough testing to ensure safety and reliability. Cognitive modeling is able to uniquely address these issues in a compromise between costly human-in-the-loop simulations and deterministic computer simulations. In this thesis, I present an ACT-R cognitive model to emulate pilot taxiing behavior in a dynamic environment, in order to predict human behavior in novel situations imposed by NextGen constraints. The model is validated by comparing taxi routes generated by the model to routes driven by real pilots while on the job.

## TABLE OF CONTENTS

Introduction.....	1
NextGen Overview.....	2
Technology.....	3
Surface traffic management.....	4
Speed-based taxi clearances.....	7
Testing.....	8
A Human Performance Model.....	11
Overview of ACT-R.....	12
External Environment.....	16
X-Plane.....	17
Virtual cockpit.....	19
Related Modeling Efforts in ACT-R.....	20
Modeling Overview.....	23
Nagivating.....	25
Maintain speed.....	29
Holding.....	33
Corrective steering.....	38
Turning.....	42
Scanning the taxiway.....	46
Missing links and future directions.....	47
Model Validation.....	49
Overview.....	49

Position data.....	52
Overall fit.....	52
Corrective steering.....	55
Turns.....	56
Velocity.....	61
General.....	64
Discussion.....	66
Potential Applications.....	66
Human-in-the-loop simulations.....	67
ACT-R.....	68
Future directions.....	68
Conclusion.....	69
References.....	70
Appendix.....	76

## LIST OF TABLES

Table 1. <i>List of parameters used to maintain speed.</i>	31
Table 2. <i>Free parameters used in steering equation.</i>	40
Table 3. <i>Mean, standard deviation, max deviations, and RMSE from the SODAA path for route one.</i>	52
Table 4. <i>Mean, standard deviation, max deviations, and RMSE from the SODAA path for route two.</i>	53
Table 5. <i>Mean, standard deviation, and max deviations from the centerline for route one and the corresponding SODAA data.</i>	55
Table 6. <i>Mean, standard deviation, and max deviations from the centerline for route two and the corresponding SODAA data.</i>	55
Table 7. <i>Average speed and time of the model compared to the SODAA data for route one, plus the correlation and RMSE between model velocity and SODAA velocity.</i>	62
Table 8. <i>Average speed and time of the model compared to the SODAA data for route two, plus the correlation and RMSE between model velocity and SODAA velocity.</i>	62

## **Introduction**

As air travel becomes more prevalent, the reliability of the air transportation system in the United States is being called into question. The rising demand for air travel has resulted in a very precarious logistic situation: a weather delay or mechanical problem causing delays at one airport may propagate, causing delays at other airports around the country, or around the world. Flight delays are not only a nuisance to passengers; they also cost airlines billions of dollars each year (Federal Aviation Administration, 2008). This is not a problem that can be solved simply by creating a larger fleet of planes and employing more pilots; the total capacity of the system needs to be increased in order to address these problems. Currently, the throughput of the system is limited by both safety regulations and technological shortcomings. To expand the capacity of the system, new technologies and procedures must be introduced that can improve efficiency without sacrificing safety.

To address this problem, the FAA, NASA, and other members of the aviation community have proposed an overhaul to the air transportation system known as the Next Generation Air Transportation System, or NextGen. NextGen is a broad plan to increase the overall performance of the system through the use of a collection of new technologies and procedures. In addition to reducing delays and saving costs, NextGen will also attempt to reduce the environmental impact of the air transportation system, all while maintaining the current high levels of safety associated with air travel. The FAA estimates that by 2018, NextGen will have saved

the industry and consumers \$22 billion, 1.4 billion gallons of fuel, and reduced flight delays by 21% (NextGen Integration and Implementation Office, 2010).

In order to make this vision a reality, NextGen must be thoroughly tested to ensure safety and reliability prior to implementation. Existing methods of testing, which include computer simulations and human-in-the-loop simulations, are often too expensive or limited in their ability to predict human behavior. In this thesis, I advocate a third approach: a realistic computational cognitive model that behaves like a human pilot, and can be used as a tool for future NextGen research. The model is implemented in ACT-R, a cognitive architecture that can be used for human performance modeling. After providing an initial overview of some of the problems addressed by NextGen and explaining why I believe current modeling approaches are insufficient, I will describe in detail how the model the model works. Finally, I provide a quantitative analysis of the model's performance by comparing data generated by the model to data from real pilots while on the job and discuss the implications of the model.

### **NextGen Overview**

NextGen addresses a variety of largely disparate problems or shortcomings with the current system in a piecemeal fashion. Thus, NextGen is used as an umbrella term for an assortment of solutions addressing a wide range of problems. This piecemeal approach allows for a gradual deployment of technologies, many of which are scheduled to be in effect by the year 2018 (NextGen Integration and Implementation Office, 2010). The overarching theme behind each of the NextGen

components is to provide a synthesis of relevant information to the people who need it, when they need it.

### **Technology**

Some of the proposed technologies will assist in reducing latency between takeoffs, which will increase the throughput of the system. For instance, Automatic Dependent Surveillance Broadcasting (ADS-B) is an enhanced positioning system in which planes broadcast their own position, acquired through the Global Positioning System (GPS), in order to provide positional information to other pilots and air traffic controllers at a level of detail not previously possible with traditional radar technology. Enhanced positioning allows for a decreased distance between aircraft while still maintaining safety. For example, in the Gulf of Mexico, where radar positioning was previously unavailable, ADS-B has reduced the minimum following distance between planes from 120 nautical miles to 5 nautical miles (Federal Aviation Administration, 2008).

Additional technologies improve the flow of traffic on the ground. For instance, Airport Surface Detection Equipment – Model X (ASDE-X) provides ground control with a detailed map of all planes and equipment on the airport surface. ASDE-X collects information from a variety of sources, including ADS-B, surface radar, and transponders, in order to facilitate prediction of runway incursions and provide ground controllers with a central information source. At airports in which ASDE-X has been deployed, it has been responsible for up to a 50 percent reduction in serious runway conflicts (NextGen Integration and Implementation Office, 2010).



Another technology, known as data-link (Kerns, 1991), will enable aircraft to receive and process instructions through digital displays instead of through the traditional method of verbal communication with ground control. This aids pilots in several ways. Foremost, it reduces the working memory load of the pilot, who is no longer forced to retain a long list of information given to him by ground control. Additionally, it reduces the opportunity for error in auditory perception, e.g., mishearing “runway 23” instead of “runway 28.” Lastly, it eliminates the possibility of miscommunication between pilots and ground controllers and frees up attentional resources of the pilot. All verbal communication with ground control occurs over a single radio channel, so a pilot is privy to communications between ground controllers and other taxiing planes. This sometimes results in a pilot responding to a command that was not meant for him (Flathers, 1987). In addition, this requires a pilot to be constantly vigilant as he listens for his own call number.

### **Surface Traffic Management**

In order to achieve NextGen’s vision, major airports need to improve upon the efficiency of their surface traffic management system. Technologies such as ADS-B and ASDE-X are creating opportunities for advancement in surface traffic management by automating the generation of taxi clearances. Currently, an aircraft’s route from the gate to the runway is determined by ground control shortly before it leaves the gate. The sequence in which planes are cleared and the taxi routes they travel are largely at the discretion of the controllers in charge. In the future, computer systems will automate a large portion of this process, generating taxi routes and departure sequences automatically, and dynamically

reprogramming them as necessary. This will allow ground controllers to focus more on safety issues, such as monitoring for potential runway incursions. This level of automation has not been possible in the past, as the position of each plane on the ground and nearby airspace was not known with enough precision to allow computer algorithms to operate effectively.

There are roughly three phases of surface traffic management to consider. The first phase is the *spot release planner*. During this phase, ground controllers must decide when and in which order to release each plane from its “spot”—the point at which a plane enters the taxiway system after leaving the gate. In order to determine the optimal spot-release time, one must work backwards from the optimal takeoff time, compensating for how long it will take the plane to taxi to the runway, in addition to congestion and delays caused by any other aircraft on the taxiways (Malik, Gupta, & Jung 2010).

The second phase to consider is the *taxi scheduler*. This refers to the route that each plane will take to get to the runway, including specific hold points. Ideally, each plane simply needs to take the shortest path possible from the gate to the runway. However, this is not always possible due to airport layouts, ground traffic, aircraft separation concerns, and the need to order runway takeoffs in a proper sequence. In addition, the *taxi scheduler* may provide temporal constraints in regards to when the plane needs to arrive at the runway.

The last phase to consider is the runway scheduler or *departure scheduler*. As planes approach the runway, they line up into several queues and await clearance for departure. Departure scheduling refers to the order in which each plane is

released from the queue and allowed to take off. Airports traditionally operate on a first-come first-served basis, so that whichever plane arrives at the runway queue first is the first to depart. However, this is not always optimal. Aircraft are required to meet minimum separation requirements when departing because of the wake vortex created by each plane. These separation requirements differ depending on the type of aircraft and are not symmetrical. Since larger aircraft tend to produce larger wake vortices, longer delays are needed after larger aircraft take off. Thus, having smaller planes take off before larger planes can minimize overall delay. Rathinam, Wood, Sridhar and Jung (2009) have developed an approach for optimizing the sequence of departures that results in approximately 12 minutes of saved time compared to a first-come first-served approach.

Although these three phases are typically evaluated as separate problems, they critically depend on each other (Balakrishnan & Jung, 2007). The departure scheduling sequence crucially depends on the accuracy of the spot-release planner and the time taken for a pilot to taxi to the runway. If one plane is late, it may end up in the wrong place in the queue, and the optimal departure sequence will have to be recalculated. Similarly, the spot-release planner is meant to work on an extended timeframe, scheduling up to one hour in the future (Malik, Gupta, & Jung, 2010). Since the spot-release planner works by anticipating taxi and departure times, the spot-release planner may not be of any use when these other phases are inaccurate.

With the advent of ADS-B and ASDE-X, there is sufficient information to normatively optimize all three of these phases. However, this optimization crucially

depends on pilots' ability to adhere to ground control clearances in a timely fashion. If pilots are unable to do so, these optimizations cannot be realized.

### **Speed-based taxi clearances**

Currently, ground controllers issue pilots a clearance that indicates the sequence of taxiways to follow, but does not indicate when they are expected to arrive at the runway. While pilots are expected to follow clearances quickly and safely, there are no quantitative guidelines on how long the taxiing procedure should take. As a result, the time that it takes for a plane to taxi from the gate to the runway is somewhat unpredictable, putting into question the viability of automated surface traffic management.

To improve the throughput of departures, the FAA is interested in providing speed-based taxi clearances, also known as time-based taxi clearances and 4D clearances. Unlike traditional clearance instructions that only convey the route that the plane will take, 4D clearances include a time component, to signify that the plane must pass through waypoints at certain times. If the plane is travelling too fast or too slow, the pilot must react accordingly. Alternatively, ground control may issue a speed requirement, which is derived from the time requirement, to let pilots know how fast they should be moving at any given moment. This requirement will likely result in additional cockpit instrumentation to let the pilot know how fast he should be moving.

Previous research has shown that while pilots are able to adhere to 4D clearances, performance may not be robust enough to allow implementation of long-term sequencing algorithms. Foyle, Hooey, Kunkle, Schwirzke, and Bakowski (2009)

found that pilots were able to comply with speed-based trajectories, with time of arrival (TOA) errors ranging from -24 seconds (early) to 53 seconds (late).

Subsequent research showed that providing pilots with additional constraints could mitigate this problem. Bakowski, Foyle, Kunkle, Hooey, and Jordan (2011) demonstrated that by instructing pilots to stay within speed bounds ( $\pm 1.5$  knots of the desired speed) and follow specific acceleration and deceleration profiles (2 knots/sec) resulted in much more accurate time of arrival predictions.

Unfortunately, a questionnaire in the same study revealed that pilots felt the speed conformance ranges were too narrow:

On average, participants reported that a  $\pm 3.7$  knot range would be reasonable. It should be noted that this would likely result in very poor TOA errors – a simple calculation indicates that depending on the speed, each 1 kt error bias over a 12,000 ft taxi results in 20-40 sec error (Bakowski, Foyle, Kunkle, Hooey, & Jordan, 2011).

Additionally, a majority of pilots ( $n=14$ ) felt that that following these procedures in the real world (as opposed to flight simulators) would compromise safety, while a minority ( $n=4$ ) indicated that it would not.

Computer generated taxi clearances and speed-based clearances are being touted as important components to accomplish NextGen's vision. However, in their current form, these technologies and procedures are not robust enough to produce the desired results. Further testing is required to determine how performance can be improved.

### **Testing**

Two traditional methods for testing new technologies and procedures have been human-in-the-loop (HITL) simulations and computer simulations. While these

are both very useful ways to test aspects of NextGen, they both have limitations as they exist in their current form.

Human-in-the-loop testing is a necessary component of the iterative design cycle. As evidenced by Foyle et al. (2009), pilots are not capable of safely adhering to changes in taxiing procedures advocated by computer simulations. HITL testing allows researchers to address this at an early stage, when it is still possible to refine the procedures. Unfortunately, HITL testing can also be extremely expensive and time consuming. The manpower required to run such tests can be quite large, including pilots, engineers, air traffic controllers, and others. It is not always monetarily or logistically feasible to use closed-system tests to predict the effects that will occur because of NextGen. In addition, HITL testing may not predict problems that may arise when a system is deployed on a large scale.

Alternatively, computer simulations can be used to predict the effects of NextGen changes. In theory, computer simulations should address both problems of HITL testing: they can be cheaper, and also predict problems that arise on a large scale. To date, however, these computer simulations have been severely limited in their ability to predict *human* performance within the system.

NASA's Safe and Efficient Surface Operations (SESO) group has addressed the surface traffic management problem by exploring the feasibility and efficiency of various taxi scheduling algorithms with Monte Carlo simulations using the Surface Operations Simulator and Scheduler (SOS<sup>2</sup>) (Wood, Kistler, Rathinam, & Jung, 2009). SOS<sup>2</sup> is an extensible, fast-time simulation architecture used primarily for developing and testing scheduling algorithms. SOS<sup>2</sup> is capable of generating 4D

clearances for pilots based on a set of inputs, including a node-link model of the airport, a list of aircraft to be scheduled, and a list of departure separation times. SOS<sup>2</sup> allows easy comparison of competing sequencing algorithms, including first-come first-served.

SOS<sup>2</sup> also allows modelers to specify custom functions that model aircraft dynamics and manipulate pilot parameters. These parameters include maximum speed and acceleration, look-ahead distance, and separation distance. However, these parameters are controlled programmatically, rather than using a cognitive model. As such, a plane will never breach the minimum separation distance, will always turn at the proper taxiway, and will always respond immediately to speed changes or other commands issued from ground control. Thus, the simulated pilots in these models typically act as non-intelligent automatons that comply perfectly and in zero time with instructions given to them.

This representation of pilots is critically inadequate for several reasons. First, real human pilots introduce a source of latency into the system that is not accounted for by these simulations. This latency will likely influence performance of the system, and should be accounted for in computer models. Second, the behavior of human pilots is stochastic. If given the same scenario twice, they will not perform identically. This variability needs to be accounted for in computer simulations. Lastly, human pilots are both intelligent and fallible. For instance, pilots do not always follow the clearance that is issued to them. Conversely, pilots are able to react to their environment accordingly: whereas an automaton might comply with instructions from ground control to cross a runway as another plane is passing, a

human pilot is able to determine that this is a dangerous action that should not be performed.

SOS<sup>2</sup> is effective for its ability to rapidly perform Monte Carlo simulations, resulting in a large amount of data to compare potential scheduling algorithms. However, it is not designed as a human performance model, and thus is insufficient to close the gap between existing computer simulations and human-in-the-loop simulations.

### **A Human Performance Model**

The aim of the current project is to provide a tool to predict the viability and efficacy of NextGen procedures by accurately modeling pilot cognition and behavior during the taxiing phase. The extant model has restricted the capabilities of the pilot for tractability, and the need to validate fundamental components before implementing supplemental features. However, the model is extensible, and additional features may build off of the existing code base in order to evaluate novel scenarios.

The model is implemented using ACT-R (Adaptive Control of Thought – Rational; Anderson, 2007), a sophisticated cognitive architecture capable of modeling complex cognitive tasks. ACT-R provides an infrastructure to accurately model pilot cognition. For example, ACT-R's goal-directed behavior can simulate a pilot's mental model of how information in the environment changes: When and where should a pilot allocate his attention? ACT-R's working memory systems may predict a pilot's situational awareness: How does a pilot navigate the taxiways and how does his recall of the taxi clearance degrade over time?



The model can be used to address many of the questions alluded to previously: How does implementing data-link instrumentation into the cockpit affect a pilot's ability to taxi effectively? Are pilots able to adhere to 4D taxi clearances, and what sort of procedural guidelines can be issued that do not compromise safety? How do various scheduling algorithms interact, and are pilots able to meet the requirements necessary to enforce these algorithms?

Additionally, the model can serve several other useful functions. The model operates in real time and can be viewed from arbitrary 3D camera angles on the airport surface, which potentially allows it to be used as a low-cost alternative for training air traffic controllers without the need for real pilots to serve in HITL simulations. The generalizability of the model makes it a potential candidate for any future aviation research that would normally use HITL testing, or for any aviation modelers in need of mechanism for incorporating pilot cognition into their simulations. The primary purpose of this thesis is to validate the model as a valuable tool for analyses of this type.

### **Overview of ACT-R**

The model is implemented using ACT-R, a framework for building computational models of cognition and behavior. Like previous cognitive architectures such as EPIC (Kieras & Meyer, 1997) or Soar (Laird, Newell, & Rosenbloom, 1987), the ACT-R framework attempts to model the underlying structural properties of the cognitive system (i.e., the mind). In this way, ACT-R follows in the tradition of Newell and Simon's General Problem Solver (Simon, Shaw,

& Newell, 1959) by creating a domain-general approach to modeling, rather than restricting itself to modeling a limited class of tasks.

This domain-general approach to problem solving makes ACT-R an ideal candidate for human performance modeling in aviation. As a unified theory of cognition, ACT-R allows the modeler to generate intricate models that involve the interplay of memory, vision, motor, and various cognitive components. Other popular cognitive modeling paradigms such as connectionism (e.g., McClelland et al., 2010) or Bayesian networks (e.g., Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010) are more often used to model lower-level psychological tasks. Modeling higher-level tasks can be prohibitively difficult in these paradigms. While there are other cognitive architectures that are in the same class as ACT-R, notably EPIC and Soar, ACT-R is arguably the most advanced in its capabilities. For these reasons, ACT-R has a proven track record of modeling complex tasks in human factors and human-computer interaction (e.g., Ball, Gluck, Krusmark, & Rodgers, 2003; Byrne, 2001; Fleetwood & Byrne, 2006; Salvucci, 2006; St. Amant, Horton, & Ritter, 2007).

ACT-R is made up of many different *modules* that represent different parts of the cognitive system, such as a declarative memory module and a visual perception module. Each module is encapsulated, so that, e.g., the visual perception module has no direct access to the memory module. However, each module can write to its own *buffer*, which stores information for use in the current task. A central executive processor then accesses these buffers and coordinates behavior across different modules.

ACT-R delineates between two distinct forms of memory: procedural memory and declarative memory. Procedural memory is modeled as a set of “if...then” rules called *production rules* (or simply *productions*). These production rules examine input to the cognitive system, specifically symbolic representations of the environment and internal mental states that are stored in the buffers. If the input matches the conditions of a production rule, the production rule fires and produces output. This output may take the form of behavior (such as eye movements, hand movements, or vocal utterances) or changes to mental states (such as manipulation of goals, attention, or items in memory). All production rules are considered as potential candidates in parallel with each other, but only one production can fire at a time. This process is known as *conflict resolution*. This production system approach to cognition is a major conceptual difference between ACT-R and other popular computational modeling approaches such as connectionism (e.g., McClelland et al., 2010).

The other memory system represented by ACT-R is declarative memory, which is modeled as a set of *chunks*, or symbolic knowledge specified in propositional form. Chunks represent declarative knowledge that exist both in the mind as well as in the environment. For example, one chunk stored in long-term memory might represent Abraham Lincoln, whereas another chunk in the environment may represent the percept of an apple in the visual field. These chunks can serve as input to a production rule, and can be manipulated as a result of a production rule being fired.

While these memory systems are described as symbols (chunks) and rules for interaction with symbols (production rules), there are also subsymbolic processes that govern both memory modules. Production rules have utilities assigned to them that determine which production fires during conflict resolution if multiple productions are capable of firing. The system also utilizes reinforcement learning through a variant of the Rescorla-Wagner rule (Rescorla & Wagner, 1972), in which utility values are modified by rewarding production rules that perform well or punishing those that do not. This allows for a system that is capable of learning. In addition, chunks are assigned activation values that determine their probability of being retrieved from memory, as well as the time required to retrieve a chunk from memory. This activation value is based on recency and frequency of retrieval, and is also influenced by spreading activation (Anderson, 1983).

ACT-R also contains a vision module to simulate high-level visual constructs, such as visual attention and visual search. ACT-R distinguishes between the notions of “what” and “where” visual systems (e.g., Goodale & Milner, 1992; Treisman & Gelade, 1980), in which limited information is available about the visual environment outside of foveal attention. Information from the “where” system is used to filter objects for visual search based on a set of features (such as color), whereas oculomotor movements are used to bring objects into focus and extract more detailed information using the “what” system. ACT-R also incorporates a type of “visual memory” by utilizing fingers of instantiation (FINSTs; Pylyshyn, 1989). The vision module does not simulate low-level features of the visual system, such as constructing percepts from raw sensory input.

ACT-R also contains several other modules, such as the imaginal and goal modules. The imaginal module is used to represent the current problem state, and stores chunks of information relevant to the current task. The goal module is used to represent intentional behavior. It allows the model to process a task using top-down knowledge of the task structure.

An ACT-R model is essentially a list of chunks and production rules that represent the knowledge of the person being modeled. The modeler may also specify the value of several parameters that govern how ACT-R modules behave, in addition to task-specific parameters such as production utility values. When the model is provided a task, ACT-R will simulate behavior and produce a *trace*. A trace is a time-stamped list of actions performed by the model, reduced to their atomic form, such as requesting an item from memory, or shifting attention to a different portion of the screen.

The trace is important because it is essentially a detailed record of *how* the model performs a task. Thus, in addition to observing the behavior produced by the model, the trace allows inspection into the internal mental state of the model. This is helpful in explaining *why* a certain behavior is produced, as opposed to simply predicting its occurrence. Such information can provide clues as to how we can modify the model to match human performance.

### **External Environment**

Most psychological tasks modeled by ACT-R are not purely cognitive in nature. In addition to cognitive components such as memory retrievals and manipulations of mental representations, most tasks require the model to interact

with some environment. The environment provides sensory input, by depicting the external world in a way that ACT-R can see and understand. Typically, ACT-R models can interact with the environment as well, producing behavior.

In this task, it is necessary for the model to see cockpit instruments (such as the speedometer), other planes on the airport surface, signage on the taxiways, and the taxiways themselves. The model must also be able to interact with the environment by manipulating cockpit controls, and moving its gaze to various parts of the display.

### **X-Plane**

When modeling an aviation task, the external environment can be quite complex. The dynamics of aircraft movement should be modeled as precisely as possible to ensure predictive validity. For this reason, the ACT-R model interfaces with X-Plane, a commercial flight simulator (shown in Figure 1). X-Plane is an extensible platform that provides realistic simulation of flight dynamics for many aircraft types. Additionally, it contains detailed maps of airports and taxiways at airports around the world.



Figure 1. A view of X-Plane from inside the cockpit.

Interfacing with X-Plane has both limitations and advantages. One limitation is that X-Plane only operates in real time, and thus fast-time simulations are not possible. This limits the ability of the model to rapidly produce data. Real-time simulations have several advantages, however. Operating in real time allows researchers a transparent research environment to work with; an aviation expert with limited or no modeling experience should be able to watch the model being run and make performance assessments. It also allows for the possibility of mixed human-computer simulations. For example, the model could potentially be used to train human air traffic controllers without requiring human pilots to pilot flight simulators, as is typically the case.



## Virtual Cockpit

Since ACT-R does not have machine vision capabilities, it is not able to directly perceive the environment from X-Plane. For the model to perceive its surroundings, the information available in X-Plane must be transformed into a representation that ACT-R can understand.

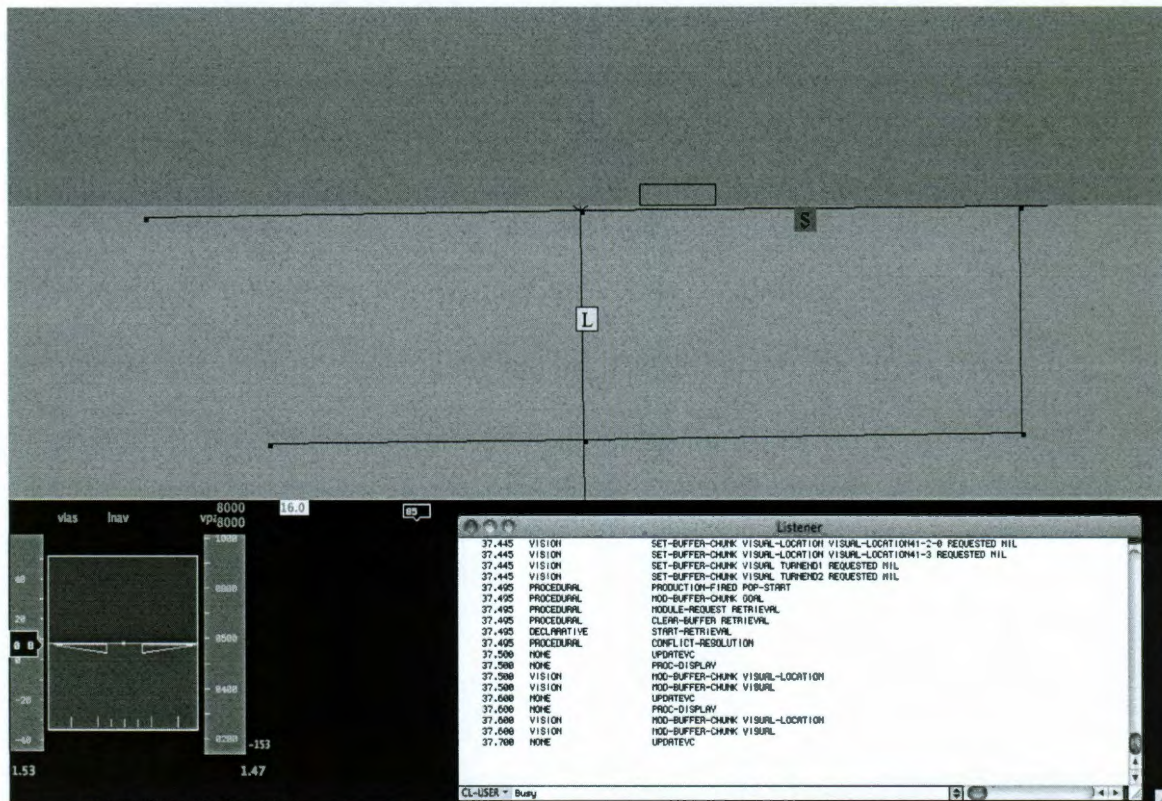


Figure 2. The virtual cockpit renders cockpit instrumentation, signage, taxiways, and planes. The ACT-R trace can be seen in the bottom right.

For this reason, it is necessary to create a mediator, known as the *virtual cockpit* (shown in Figure 2), which is essentially a Lisp program in which the X-Plane view has been redrawn and coded using a symbolic representation that ACT-R can understand—as a series of chunks. The virtual cockpit communicates with X-Plane through its plug-in infrastructure in order to retrieve state variables, such as



location, heading, and velocity. These variables are used to reconstruct the view of the taxiways, as well as readings on the cockpit instrumentation. The taxiways themselves are supplied to the virtual cockpit as a node-link model, with an additional list of signs and their locations. Although the fidelity of the virtual cockpit is inferior to that of X-Plane, it captures the fundamental structure of the environment that is necessary for the model to interact. Extraneous information, such as texture and perspective are not represented in the virtual cockpit.

The model is able to explore the environment through the virtual cockpit and act accordingly based on its current state. As productions fire in ACT-R, state variables are updated and sent back to X-Plane, which in turn modify the environment. In this way, the cognitive model can drive a plane in X-Plane. See Figure 3 for a visual depiction of this sequence.

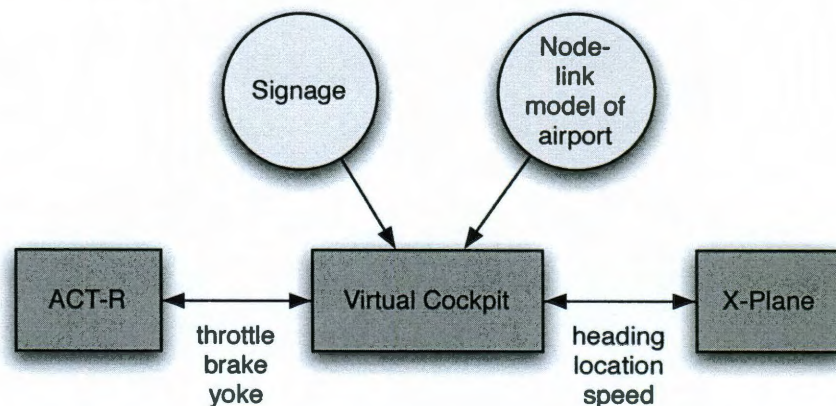


Figure 3. ACT-R interacts with X-Plane through the virtual cockpit. State variables are received from X-Plane, manipulated by ACT-R, and sent back to X-Plane.

### Related Modeling Efforts in ACT-R

While computational modeling has been applied to aviation tasks before, there have been no attempts to create a general-purpose computational cognitive

model of pilot taxiing behavior to my knowledge. There are several previous efforts, however, that provide useful insight into the current problem. In particular, there have been several efforts that use ACT-R to predict pilot performance on a variety of aviation related tasks.

Ball, Gluck, Krusmark, and Rodgers (2003) used ACT-R to simulate the operation of a Predator Uninhabited Air Vehicle (UAV). Their task involves manipulating the airspeed, altitude, and heading of a UAV in order to achieve a specific goal. Ball et al. also varied the level of declarative and procedural knowledge available to the model, and task performance was compared between three variants of the model, as well as performance by a human expert.

While the UAV task did not involve a driving component, the model is similar to the current work in several respects. Foremost, it highlights the complexity of manipulating multiple aircraft controls in a dynamic environment, and ACT-R's ability to achieve human-like performance at such a task. It also highlights the importance of imbuing the model with the proper knowledge in order to successfully model human behavior. Lastly, Ball et al. provide novel methods of validating performance on a complex aviation task that may be applied to the current model.

Byrne et al. (2004) examined the effect of adding a synthetic vision system (SVS) to the cockpit. An SVS is a computer-generated display of the outside world that can be used as a proxy for looking out the window when visibility is poor. However, the SVS is also overlaid with various pieces of information such as altitude, airspeed, and heading.

ACT-R was used to monitor flight controls while an autopilot performed a landing in a flight simulator. Although the addition of the SVS did not decrease task performance, it greatly influenced gaze location of the model (as well as the real pilots). In particular, the model did not use the SVS as a substitute for looking out the window as it was designed, but instead used the SVS as a substitute for looking at other cockpit gauges, such as the altimeter and speedometer.

Although performance on the landing task stayed the same, Byrne et al. note that this redistribution of attention within the cockpit may have a substantial effect when the task is more difficult. This suggests that the cockpit layout used by ACT-R may affect the predictive validity of a model, and that the effects of changes in cockpit layout may be adequately predicted by ACT-R. Additionally, the model presented by Byrne et al. may serve to complement the present work. Their model and the present model focus on two orthogonal tasks: landing and taxiing. Since both models use the same underlying architecture, one could possibly merge the two models to perform more complex aviation maneuvers.

Byrne and Kirlik (2005) examined pilot performance on taxiing to the gate after landing. Although pilots are supposed to remember and follow the exact clearance given to them by ground control, they often fall back on a set of heuristics for navigating to the gate. These heuristics include “follow the plane ahead of you” and “turn in the direction of the gate.” While these heuristics are often successful, they have the potential to result in a serious runway incursion. Using ACT-R, Byrne and Kirlik were able to predict which heuristic or strategy pilots use in different circumstances.

The taxiing model of Byrne and Kirlik is perhaps the most relevant to the current model. The taxiing strategies identified by them critically influence the timing and route of the plane. This decision-making aspect should be subsumed under a larger model of pilot taxiing behavior such as the present work.

Lastly, Salvucci (2006) created an ACT-R model of driving behavior that has been used to predict driving performance under varying conditions. Although this model is not aviation related, driving a car and driving a plane are similar in many respects. In particular, Salvucci defines a steering law that guides steering behavior. This law attempts to minimize the angle between the current and desired heading, while also minimizing the change in heading angle over time. Pilots may make use of a similar steering law, though the dynamics of the aircraft do not provide the same level of feedback as that of a car.

In addition, both Salvucci's driving task and the current task can be divided into a few subtasks that require time-sharing. A major factor in the effectiveness of both models is the degree to which ACT-R can monitor and perform these subtasks efficiently. A good model is one that is able to optimize performance across a range of subtasks.

### **Model Overview**

A thorough task analysis of pilot taxiing was conducted by gathering information from subject matter experts (commercial pilots), official airline taxiing documentation, and academic journal articles. Subsequently, this information was used as a rubric for creating the production rules in the ACT-R model. The task analysis identified several key goals that are required for successful performance on

a taxiing task. These goals include navigation, scanning the taxiway for potential incursions, maintaining the speed of the aircraft, steering, processing audio from ground control and nearby aircraft, and various pre-flight items.

The ACT-R model contains separate routines to perform each task identified in the task analysis. As such, the model requires a high degree of multitasking (more aptly, task switching). Although ACT-R is capable of performing several actions in parallel if those actions utilize different modules (such as motor and visual modules), it does not allow multiple simultaneous activities within a module. Thus, ACT-R commits to a psychological theory of multiple bottlenecks (e.g., Wickens, 1984) unlike some theories, which propose a general resource pool that allows multiple simultaneous cognitive processes to run in parallel, but at a fraction of the total efficiency (e.g., Moray, 1967).

To solve this problem, the model uses a goal stack strategy (Anderson, Kushmerick, & Lebiere, 1993) in order to transition between top-level goals and subgoals identified in the task analysis. At the top level, the model stochastically chooses a top-level goal based on the utility values specified in the model. These utility values roughly indicate a pilot's mental model of how often each task needs to be evaluated. While more complex theories of multitasking have been proposed for ACT-R, notably Salvucci and Taatgen's (2008) threaded cognition, such a mechanism would substantially increase the complexity of the model. However, as the capabilities of the current model are augmented, it may be necessary to reconsider such options.

Many complex motor control actions performed by the pilot, such as turning, corrective steering, and smooth braking cannot be adequately modeled using ACT-R's default motor system, which is designed for simple button presses and mouse movements. However, these motor actions can be modeled using a closed-loop dynamical systems approach, where information acquired from the environment serves as input into a motor control algorithm. ACT-R's motor capabilities have been extended by modeling each motor action as standalone function that is executed continuously in a loop until some exit criteria are met. The parameters of these functions have been tuned to reflect the knowledge of expert pilots, in order to simulate expert pilot behavior on a taxiing task.

### **Navigating**

**Overview.** The navigation top-level goal is required for the pilot to maintain situational awareness. Prior to taxiing, the pilot is issued a list of clearance instructions that indicate which taxiways to follow to get from the gate to the runway in addition to any spots where he must hold. The pilot must retain these instructions in memory as he navigates the taxiways. The pilot must also have an accurate mental map of where he is on the taxi surface, which taxiways he needs to turn onto, which way to turn, and where to hold. The pilot can create and update his mental map by looking at markings on the taxiways that indicate the current location, and comparing them to the pilot's taxiing route stored in memory. If the pilot believes that he is not where he is supposed to be, he must take proper recourse by getting back on track or notifying ground control.

**ACT-R Model.** The ACT-R model holds a chunk in the imaginal buffer that lists several pieces of information relevant to the plane's current state on the taxiway, including the plane's *current taxiway*, the *last taxiway* that the plane was on, and the *next taxiway* of importance.

In addition, several chunks are stored in declarative memory that indicate what to do at each point along the clearance by pairing a taxiway with an action. These chunks are referred to as *taxi-actions*. For instance, a *taxi-action* might state: "turn right at taxiway A" or "hold at taxiway EL." These chunks represent a list of rules that guide the pilot on the airport surface. A complete list of *taxi-actions* makes up a set of clearance instructions. In a real situation, these instructions are relayed from ground control over the radio, and the pilot has to retain these instructions in memory. In the future, data-link technologies will allow pilots to receive and retain a list of clearance instructions using a text-based display in the cockpit.

In the current model, clearance instructions are loaded into declarative memory prior to running the model. This obviates the need to receive auditory instructions from ground control by starting the model after this phase takes place, though future versions of the model may include a data-link component. Since items in declarative memory decay as time passes, the time taken to retrieve a *taxi-action* from memory increases the longer the model is running. In addition, the model may fail to retrieve the correct clearance instruction. This is particularly likely if the model is run multiple times without resetting its memory. This situation is analogous to one that may occur in real life if, for instance, a pilot who is used to

making a routine sequence of turns at an airport is suddenly presented with a different required action: turn left at taxiway B instead of right.

When the navigation top-level goal is called, the model begins visually scanning all of the signs that are currently visible on the taxiway, one by one, and decides what to do with each one. There are two types of signs that are present on the taxiways: black signs, which indicate the taxiway that the plane is currently on, and yellow signs, which indicate the crossing taxiway.

If the model reads a black sign, it compares its content to the *current taxiway* slot in the imaginal buffer. If the two are identical, no action is taken. If the two are different, the plane is on a different taxiway than the model believes. This should happen very rarely, but it is possible if the plane has mistakenly turned on to the wrong taxiway. The model currently contains no decision-making strategy for navigating back on course when it is on the wrong taxiway. In a real-life situation, subject matter experts have indicated that they will resolve the situation by radioing ground control or by navigating back to the correct taxiway using their knowledge of the airport layout or consulting a map. This task is made easier by the fact that commercial jetliners have both a pilot and a pilot not flying (colloquially known as the co-pilot), who is able to help navigate. Currently, the model simply records the error in the trace, so that one can see where the model went off course and perhaps determine the frequency of such errors.

If the model encounters a yellow sign, it compares its content to the *next taxiway* slot in the imaginal buffer. The *next taxiway* may either indicate a taxiway that the plane should hold at, or a taxiway that the plane should turn on to. If the



model compares the content of the sign with the slot in the imaginal buffer and they are not equal, no action is taken. If the two are equal, the model makes a declarative memory request for a *taxi-action* corresponding to the taxiway listed on the sign. If the *taxi-action* indicates that a turn should be made, the turn subgoal is called. If the *taxi-action* indicates that a hold should be performed, the hold subgoal is called. After the turn or hold is finished, the slots in the imaginal buffer are updated to reflect the pilot's new position on the taxiways. When the model has read all of the signs that are visible, the model chooses another top-level goal. This process is shown as a list of production rules and transitional states in Figure 4.

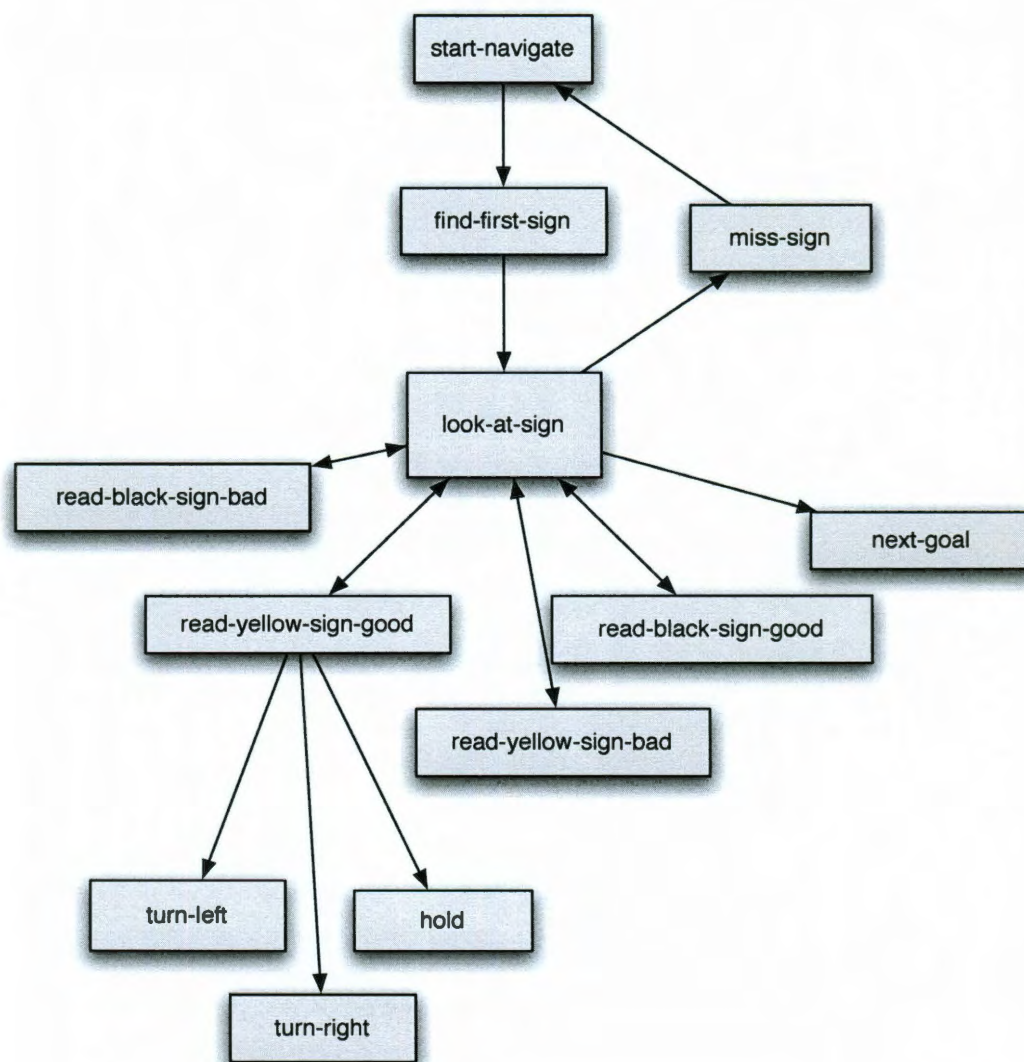


Figure 4. A rough flow chart of the production rules involved in the navigation goal.

### Maintain Speed

**Overview.** In order to maintain the speed of the aircraft, the pilot must occasionally look at the speedometer to be aware of the plane's speed. The pilot must then decide how to respond based on his current speed and a *target speed*. This requires a combination of manipulating the throttle and the brakes. In a typical large aircraft, idle thrust is quite high, which means that the pilot only needs to

move the throttle forward enough to get the plane moving (known as “breakaway thrust”). The pilot may then move the throttle back to the idle and the plane will continue to accelerate. Often the pilot will only use the brakes to control the plane’s speed after initial breakaway, though the throttle will need to be used after coming out of a sharp turn to regain speed.

Pilots are also able to control the speed of the plane without actually looking at the speedometer. They are able to do this by relying on visual motion cues and kinesthetic feedback. ACT-R has no built in mechanism for speed estimation in these situations, and this capability is not included in the maintain speed goal<sup>1</sup>.

**ACT-R Model.** In addition to storing the location of the plane on the airport surface, the imaginal buffer also stores the *target speed* of the plane. This represents the maximum speed that the plane should be moving, and differs depending on whether the plane is traveling on a straightaway or when the plane is performing a turn (see Table 1).

---

<sup>1</sup> However, this capability is simulated in the hold subgoal as described in a later section.

Table 1. List of parameters used to maintain speed.

Parameter	Value
Target speed on straightaway	15 knots
Target speed on 90° turn	9 knots
Braking	0.10 <sup>2</sup>
Throttle	0.03 <sup>2</sup>

When the maintain-speed top-level goal is called, the model fixates on the speedometer, and compares this value to the *target speed*. The model then executes one of three productions to manipulate the throttle depending on the current speed and position of the throttle. If the plane's speed drops to four knots below the target speed, a production fires that moves the throttle is to the forward position. If the plane's speed exceeds one knot below the target speed, the throttle is moved to the idle position. If neither of these conditions is met, a dummy production fires which does not move the throttle.

Subsequently, the model fires one of three productions that manipulate the brake. If the plane's speed is above the *target speed*, then the brakes are applied. If the plane's speed is three knots below the target speed, the brakes are released. If neither of these conditions is met, a dummy production fires which does not affect the current state of the brakes. These values are depicted in Figure 5.

---

<sup>2</sup> Brake and throttle parameters are unitless in X-Plane. They represent a proportion of total brake/throttle applied, where 0 is no brake/throttle and 1 is full brake/throttle. The relationship between velocity and throttle/brake parameters is shown in Appendix 1.

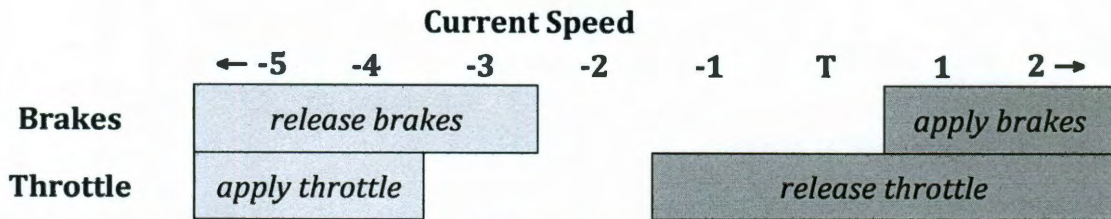


Figure 5. Manipulation of the brake and throttle is determined by the current speed relative to the target speed (T).

When the brakes or throttle are applied, it is done at a constant rate (indicated in Table 1). Thus, the amount of brake applied does not increase or decrease as the speed of the plane changes. Since this procedure keeps the speed of the plane relatively stable, dynamic braking and throttling is not utilized. The braking and throttle parameters have been chosen to mirror real-world data while staying within an acceptable range of +/- 2 knots/sec (Bakowski, Foyle, Kunkle, Hooey, & Jordan, 2011). Ultimately, these values are free parameters that can be manipulated by the modeler. One may wish to change these values based on several factors such as: the pilot being modeled (to introduce individual variation), the type of plane being modeled (to reflect the notion that smaller aircraft can safely travel at higher speeds), and the airline being modeled (to reflect different procedural standards). This process is shown as a list of production rules and transitional states in Figure 6.



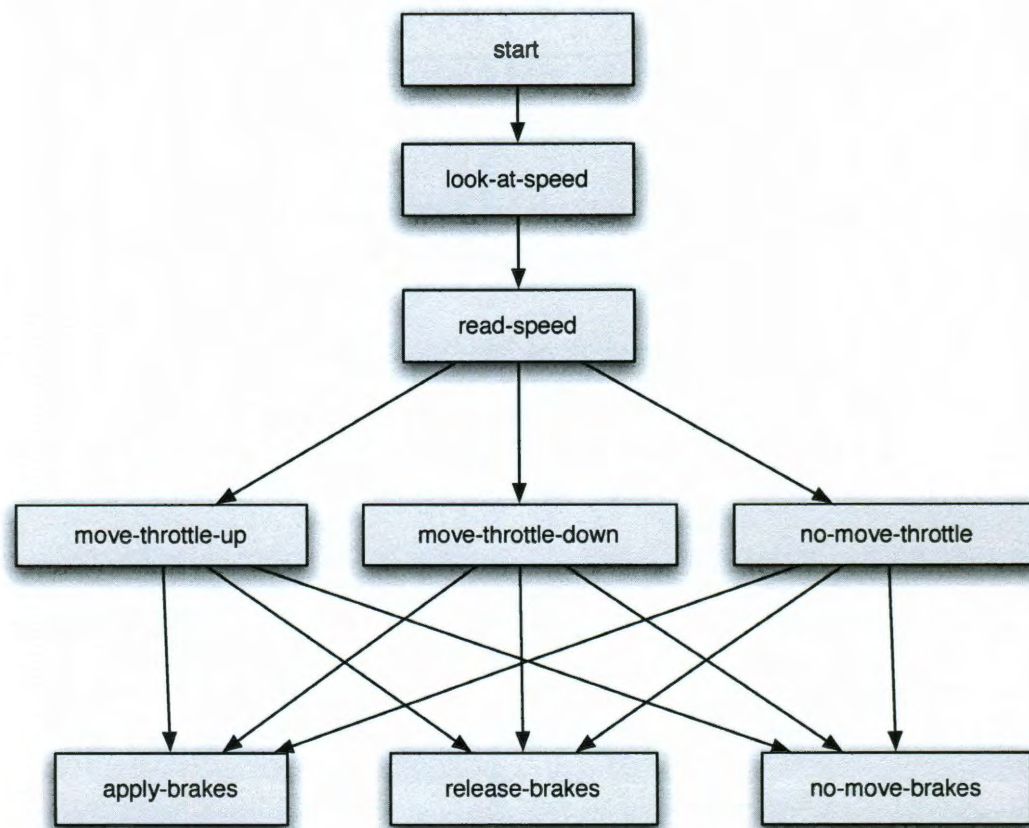


Figure 6. A rough flow chart of the production rules used to maintain speed.

## Holding

**Overview.** During a pilot's clearance, he may be required to come to a complete stop, or *hold*, at one or more points along the route. Pilots are required to hold before crossing any runway, and are often asked by ground control to hold at certain taxiways to let crossing traffic pass. When a pilot approaches a hold, he applies the brakes in a smooth motion to ensure safety and passenger comfort. In order to accomplish this in the smoothest motion, the pilot should apply the brake at the minimum level that will bring the plane to a halt at a given point. To determine how much brake should be applied, the pilot uses two pieces of

information from the environment: the current speed of the plane, and the distance to the hold point.

Several tests were conducted in X-Plane to determine the stopping distance for a Boeing 737-800 plane using several braking parameters. Starting from a speed of about 10 knots, a constant amount of brake was applied until the plane came to a halt. The total distance needed for the plane to come to a halt was measured and divided by the change in speed to determine an average stopping rate ( $\omega$ ) in meters per knot decrease in speed. The X-Plane simulations revealed a piecewise function for the stopping rate (see Figure 7), where the speed drops off precipitously under around 5.3 knots.

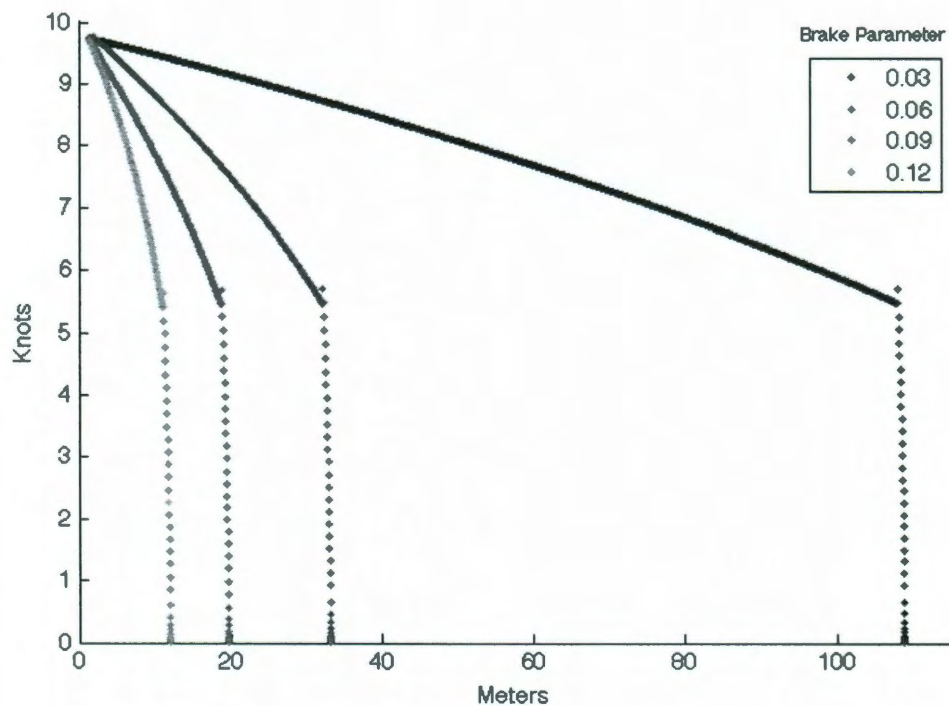


Figure 7. A plane's speed is plotted against meters traveled as it comes to a halt with varying amounts of brake.



The stopping rate was computed using only the first segment of the piecewise function under the assumption that any amount of brake would rapidly bring a plane to a halt when traveling at speeds lower than 5.3 knots. A stopping rate was computed for several different braking parameters, and the two values were plotted against each other (see Figure 8).

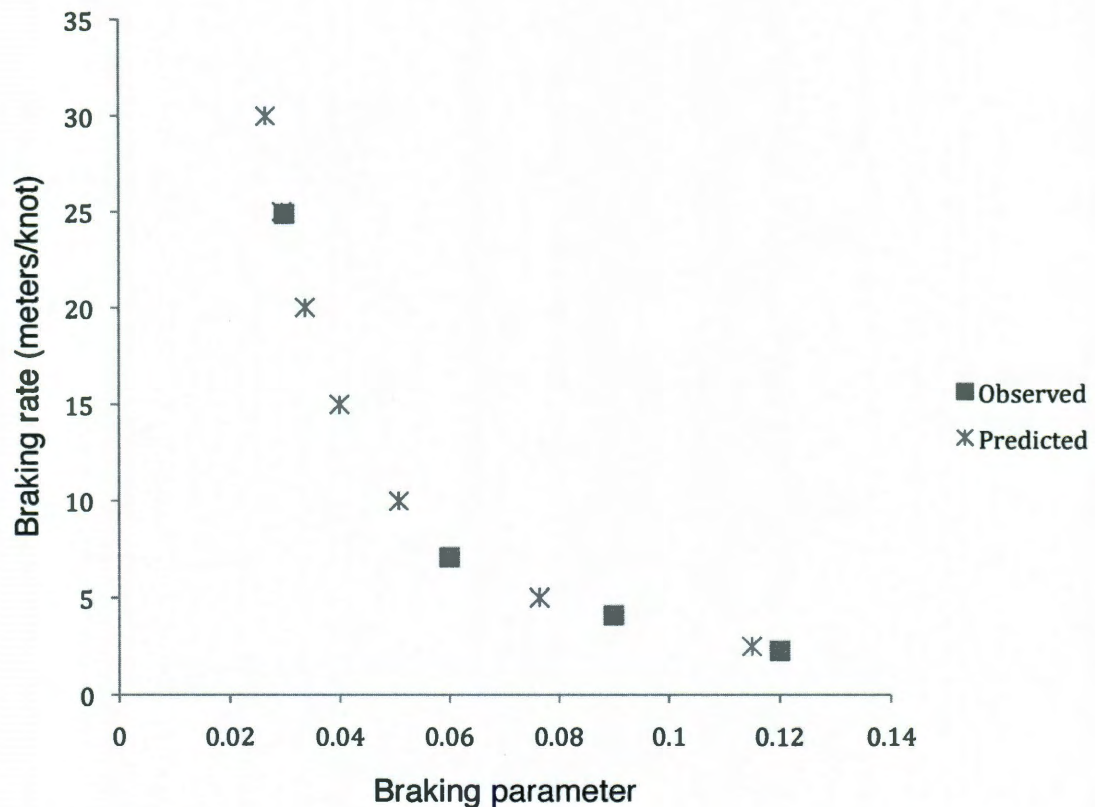


Figure 8. Several braking parameters were plotted against the observed stopping rate ( $\omega$ ). Blue squares indicate observed values, whereas red stars indicate predicted values based on the fit of Equation 1.

The relationship between braking parameter and stopping rate is fit well by the power function in Equation 1 ( $R^2 > .99$ ), where  $\beta$  is the X-Plane braking parameter.



$$\omega = 0.064\beta^{-1.695} \quad (1)$$

This function is rewritten in terms of  $\beta$  in Equation 2 in order to generate a function that can determine the proper braking parameter based on distance and speed.

$$\beta = \frac{0.2}{\omega^{0.59}} \quad (2)$$

Conceptually,  $\omega$  is simply the distance to the desired stop point divided by the current speed. However, this value needs to be adjusted to compensate for the piecewise function found in the plane's stopping behavior, as shown in Equation 3.

$$\omega = \frac{d}{s - 5.3} \quad (3)$$

As an example, suppose a plane is traveling at 10 knots ( $s=10$ ), and needs to stop 20 meters short of a hold line in 200 meters ( $d=180$ ). In this case, the pilot should apply a brake value ( $\beta$ ) of approximately 0.023 in order to perform a smooth stop.

**ACT-R Model.** If the navigation top-level goal identifies that a visible sign is identical to the *next taxiway* and a *taxi-action* indicates that a hold is to be made at that intersection, the hold subgoal is called. Once the hold subgoal is initiated, the model fixates on an object that represents the hold point, such as a sign or the intersection itself. The model begins visually tracking the object, so that the current distance to that point is continually stored and updated in the visual buffer. A hold production fires, which calculates and applies the amount of brake needed using Equation 2.

The distance supplied to this equation is the distance to the intersection stored in a slot of the visual buffer of the intersection, minus 20 to ensure that the

plane stops 20 meters<sup>3</sup> before the intersection. Unlike the distance value, the current speed is not directly available to the ACT-R model. Pilots do not (and probably should not) stare at the speedometer while bringing the plane to a halt, yet they still have a sense of how fast the plane is moving based on visual motion cues and kinesthetic feedback. The true velocity of the plane is supplied directly to the algorithm as a proxy for these senses.

This hold production will fire every 500ms, as long as the speed of the plane is above 5.3 knots and the hold point is still visible. If the current speed drops below 5.3 knots, a small, constant amount of brake is applied until the plane comes to a halt. If the hold point disappears from view, the pilot immediately applies full brakes, as he has just driven through a hold line. Although this is not necessarily a safe maneuver, it potentially avoids a much more serious runway incursion. Under normal circumstances, this does not occur in the model, but nonetheless a contingency is built in for off-nominal situations.

There are two methods for coming out of a hold. Firstly, if the pilot observes a plane crossing the runway in front of him while he is stopped, he will wait for that plane to pass and then resume. Alternatively, if the pilot comes to a complete stop and does not see a crossing plane, the model waits several seconds and then assumes it is safe to resume taxiing. This process is shown as a list of production rules and transitional states in Figure 9.

---

<sup>3</sup> In the current model, this value is simply a constant determined from observations of real-world data. A noise parameter may be added to add more variability to the stopping point.

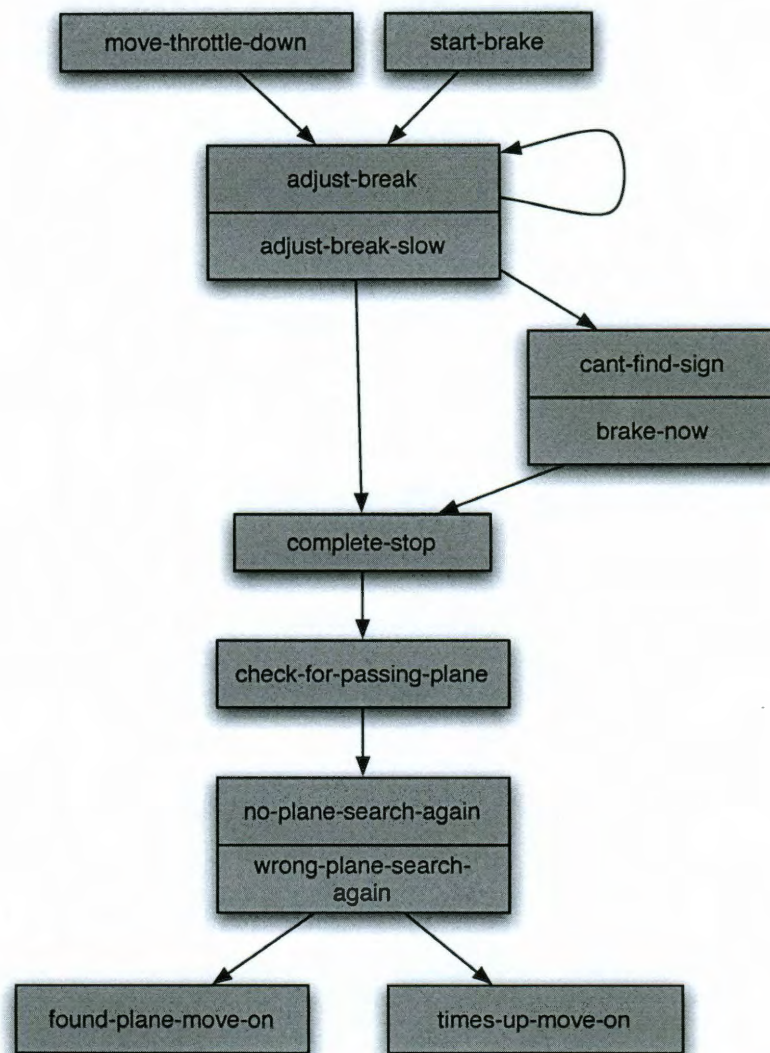


Figure 9. A rough flow chart of the production rules used to bring the plane to a hold.

### Corrective Steering

**Overview.** It is very important for the pilot to stay as close to the center of the lane as possible, as deviation from the centerline may potentially result in an incident in which the plane's wing hits an obstacle. Ideally, the front wheel of the plane should be on the center taxi line. Since airplanes are very large vehicles (a Boeing 737-800 has a wingspan of approximately 35.7 meters), it is not always

possible for a pilot to visually ensure that the plane is clear of any obstructions, such as towers or other planes. Staying on the centerline lets the pilot know that the plane is laterally clear of any other obstructions.

In order to stay on the centerline, the pilot must periodically attend to his location and adjust the yoke. Otherwise, the plane may start to veer slightly due to engine torque, wind, or simply because the yoke position must be very precise to follow such a narrow target. The model uses a one-point visual control model of steering that has been adapted from Salvucci and Gray (2004) and Salucci (2006).

The steering model relies on a single fixation point on the center of the taxiway several meters in front of the plane. From this point, two angles are derived: the visual angle to the fixation point ( $\theta_{\text{fixation}}$ ) and the angle (slope) of the taxiway itself ( $\theta_{\text{taxiway}}$ ) from the pilot's perspective. The pilot's steering angle ( $\varphi$ ) can then be computed through the regression equation in Equation 4<sup>4</sup>.

$$\Delta\varphi = k_1\theta_{\text{fixation}} + k_2\theta_{\text{taxiway}} \quad (4)$$

This equation attempts to keep the taxiway straight ( $\theta_{\text{taxiway}} = 0$ ), while at the same time keeping a visual fixation in the center of the visual field ( $\theta_{\text{fixation}} = 0$ ). In doing so, the model avoids jerky movements that may lead to oscillation. Free parameters used in the current model are listed in Table 2.

---

<sup>4</sup> The ACT-R model used to collect data mistakenly contained another term,  $\Delta t$ , which was multiplied by  $k_2$ . This term is a vestige from an earlier attempt at corrective steering which represents the time since the steering angle was last computed. However, this term remained relatively constant at .5 seconds. As such, the true  $k_1$  parameter used in the model (.25) is multiplied by  $\Delta t$  to arrive at the value for  $k_1$  shown here. This discrepancy should not affect the behavior of the model, but is mentioned in view of full disclosure.

Table 2. Free parameters used in steering equation.

Parameter	Value
$k_1$	.125
$k_2$	1.0

**ACT-R Model.** When the corrective steering top-level goal is called, the model directs its visual attention to a point on the center of the taxiway an arbitrary distance from the current position. This point, which represents the fixation point used in the steering model, is depicted in the model abstractly using a small green box (shown at point 4 in Figure 10).

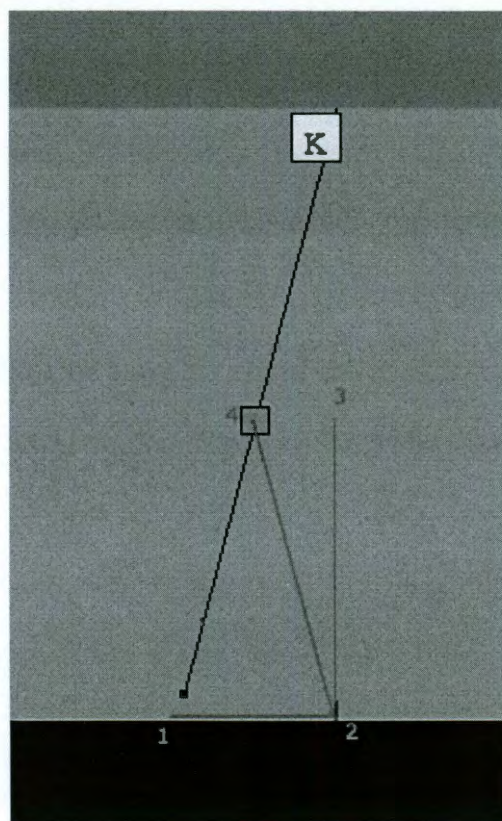


Figure 10. A cropped, close-up view of the virtual cockpit depicts both angles used in the model. Point 2 represents the current position of the plane, whereas point 4 represents the fixation point. Angle 423 is the fixation angle ( $\theta_{\text{fixation}}$ ) and angle 412 is the taxiway angle ( $\theta_{\text{taxiway}}$ ). Note that the red lines and numbers are not normally visible in the virtual cockpit.

If the visual angle to the fixation point is less than two degrees, it is assumed that the plane is roughly on the centerline, and the goal is exited. If the angle is greater than two degrees, it is assumed the plane is veering off course. If this occurs, the model begins visually tracking the fixation point in order to continuously update this angle. A steering production fires that computes a new steering angle to move the plane towards the centerline. Since the fixation point is being visually tracked,  $\theta_{\text{fixation}}$  can be computed directly from information in the visual buffer. The slope of the taxiway ( $\theta_{\text{taxiway}}$ ) is also indirectly computed using information from the visual buffer and the plane's current heading. The steering production rule fires roughly every 500ms until the angle to the fixation point is less than two degrees, at which point the goal is exited. This process is shown as a list of production rules and transitional states in Figure 11.

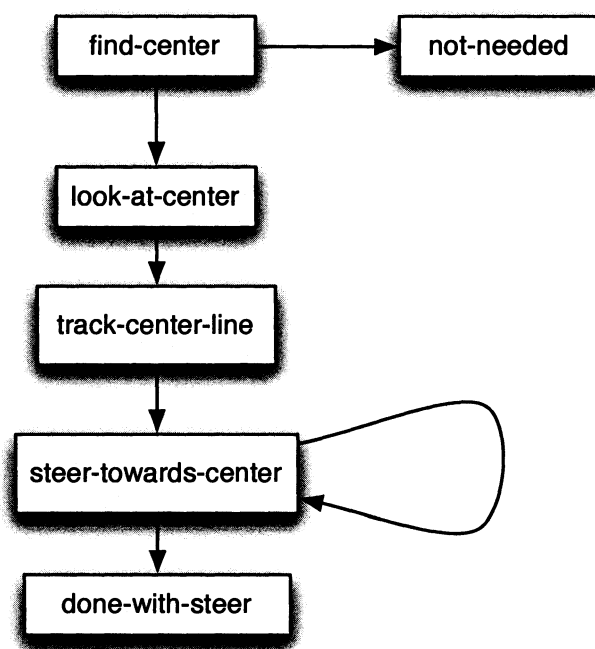


Figure 11. A rough flow chart of the production rules used for corrective steering.

## Turning

**Overview.** In order to model the trajectory of a turn, real-world positional data was used to construct a mathematical function for an average turn. This function is used to predict the instantaneous heading of the plane at any given point during the turn.

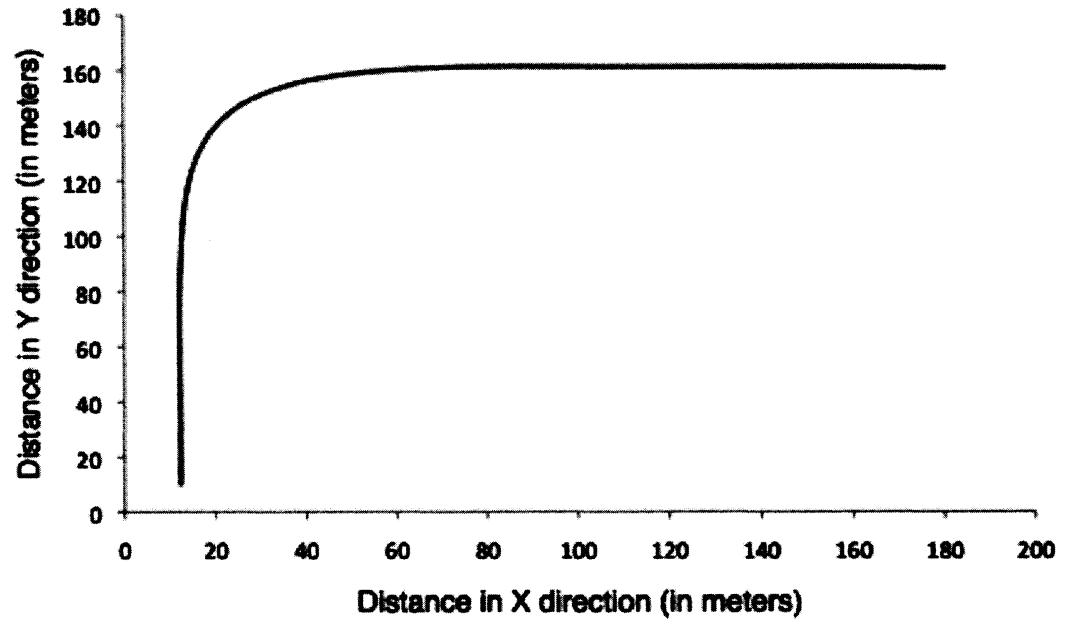


Figure 12. A turn performed by a human pilot is plotted in two-dimensional space.



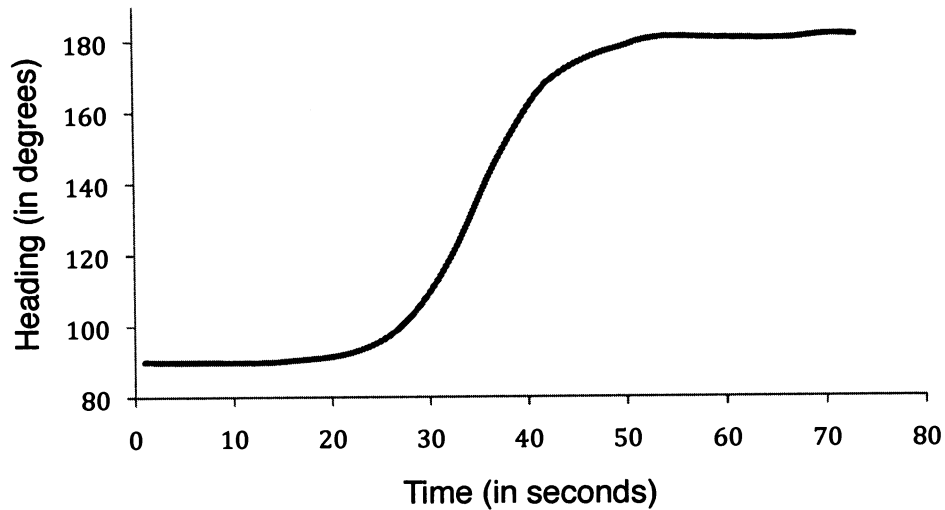


Figure 13. A plane's heading changes through time as it performs a 90 degree turn.

Figure 12 shows the position of a plane as a pilot performs a 90 degree turn. Figure 13 shows the heading of the plane through time for the same 90 degree turn. This second function was fit to a cumulative exponential function in Equation 5, where  $H$  is the predicted heading.

$$H = Qe^{-kx} + c \quad (5)$$

$Q$  represents a multiplicative constant to ensure the angle is in degrees, and  $c$  represents an additive constant that denotes the final desired heading of the plane.

Conceptually,  $x$  represents time and  $k$  is a constant. However, the situation is complicated by the fact that the shape of the graph (Figure 13) compresses when the plane is travelling at higher speeds. Using a single constant for  $k$  results in wider turns as the speed of the plane increases. To compensate for this,  $k$  needs to be computed dynamically, based on the speed of the plane. Equation 6 was found to



compensate for this effect. Using an adjusted value for  $k$  ensures that the equation results in a good turn, independent of speed.

$$k = \frac{s^3 + 6}{18800} \quad (6)$$

Though in theory Equation 5 should yield a good turn, in practice it was found that this algorithm did not converge on the final heading quickly enough. This is perhaps due to the fact that a new heading is calculated discretely, in roughly 500ms intervals, though the reason is not quite clear. A corrective factor was applied, transforming  $x$  by multiplying by the proportion of the distance that has been travelled. This corrective factor is shown in Equation 7, where  $\Delta t$  indicates the time passed since the turn began,  $D$  represents the total distance, and  $\Delta d$  represents the distance traveled so far.

$$x = \frac{\Delta t^2 \Delta d}{D} \quad (7)$$

### **ACT-R Model.**

If the navigation top-level goal identifies that a visible sign is identical to the *next taxiway* and a *taxi-action* indicates that a turn is to be made at that intersection, the turning subgoal is called.

When the turning subgoal is first called, the model lowers its *target speed* in anticipation of a turn. The model then scans the visual scene for a “turning marker”, depicted as a red square in the model. This artificial marker symbolizes the point at which the plane will finish its turn and stabilize its heading. This point is fixed in the virtual environment, unlike the dynamic marker used for corrective steering.

The model visually tracks the turning marker for the duration of the turn. The distance to this point is stored in a slot of the visual buffer, and serves as input to Equation 7. A turning production calculates a new heading, using Equation 5, and the plane's heading is updated accordingly. As the model performs a turn, the speed of the plane gradually increases. Sporadically, a production fires that taps on the brake in order to keep the speed of the plane from increasing too much. When the turning marker disappears from view, the turn is complete. The model raises its *target speed* and exits the turning subgoal. This process is shown as a list of production rules and transitional states in Figure 14.

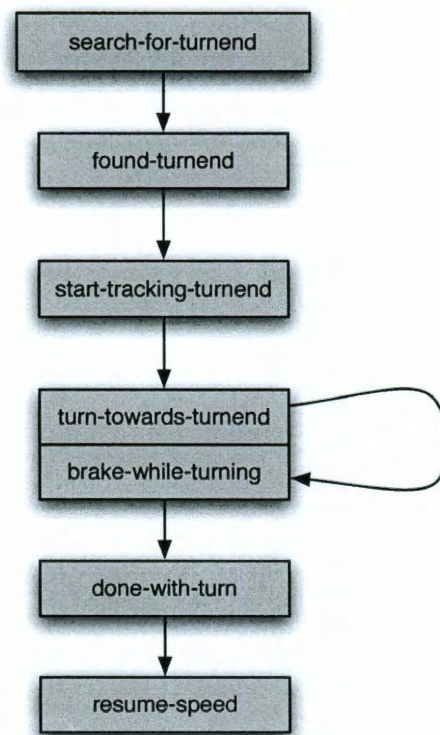


Figure 14. A list of production rules and transitions used to perform a 90 degree turn.

## Scanning the Taxiway

**Overview.** In addition to general navigation, scanning the taxiway helps the pilot maintain situational awareness. Scanning the taxiway involves looking out the cockpit window for potential incursions. While most often this means looking for other planes, it also applies to looking for fuel trucks, baggage trucks, or anything else that is not supposed to be on the taxiway or is in the path of the plane. The pilot must then decide what to do given the circumstances. If a collision is imminent, he may come to a complete stop or alter his course. If a collision is not imminent, the pilot may slow down, or maintain speed and continue to monitor the situation.

**ACT-R Model.** When the scan-taxiway top-level goal is called, the model begins looking for nearby planes, scanning the environment from left to right. If there is a plane less than 400 meters away, it will shift its attention to that plane. This process repeats for each visible plane within 400 meters. Currently, the model provides no recourse for when a plane is too close. However, the action is recorded in the trace. A list of production rules and transitions used to scan the taxiway is shown in Figure 15.

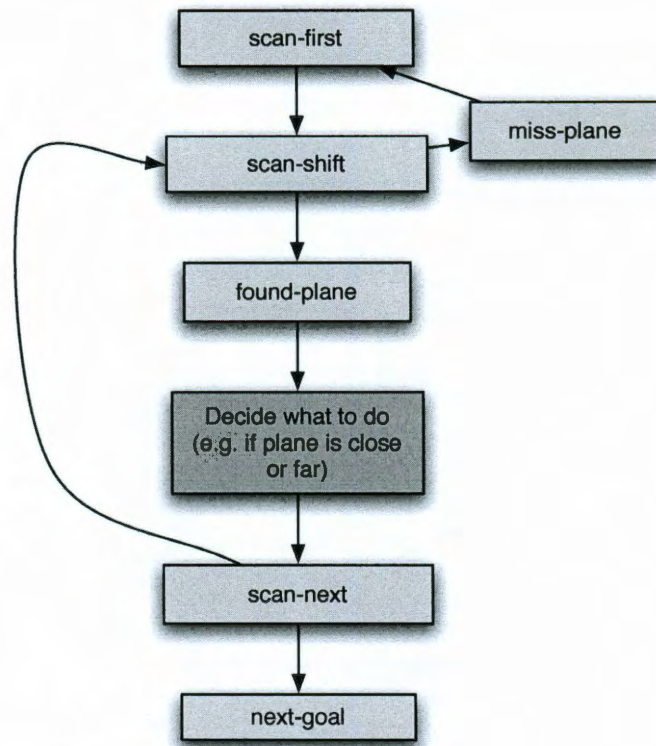


Figure 15. A list of production rules involved in scanning the taxiway.

### Missing Links and Future Directions

There are several additional tasks that pilots are responsible for that are not presently included in the model. While not all of these are necessarily critical for model simulations, these aspects need to be addressed and perhaps included in subsequent versions.

Currently, scanning the taxiway is the most underdeveloped of all of the features implemented in the virtual cockpit. Its present purpose is simply to act as a filler for the time it takes for a pilot to examine his surroundings. Notably absent is any sort of recourse when a nearby plane is too close, or any procedure for following a leading plane. This could potentially lead to loss of separation requirements between planes. In theory, it should be possible to maintain

separation by employing a braking procedure very similar, if not identical to the mechanics employed using Equation 2 for holds. However, this procedure has yet to be tested for enforcing separation requirements, and as such is not included in the extant model.

Future versions of the model must also include a method for communication between ground control and pilots, most likely in the form of a data-link component in the cockpit. The alternative, relaying clearance instructions over radio, necessitates audition and natural language—neither of which are a strong suit of ACT-R (though see Ball, Heiber, & Silber, 2007 for attempts). This will allow dynamic situations that are more typical of pilot taxiing. As is, the current simulations are likely to overpredict a pilot's ability to follow clearance instructions accurately.

The model is also likely to benefit from integration with previous efforts, such as Byrne and Kirlik's (2005) research on taxiing heuristics. Incorporating taxi heuristics into the model will strengthen the ability of the model to predict navigational errors. Lastly, there are a variety of miscellaneous tasks that pilots perform that are not included in the model, such as pre-flight checklists and maintenance checks. However, subject matter experts have informed us that most of these tasks are done by the co-pilot, whereas the pilot is primarily responsible for actually driving the plane. As such, it is believed that incorporating these secondary tasks may not be necessary.

## **Model Validation**

### **Overview**

By using a task analysis as a basis for constructing the model, it is believed the model has a high level of content validity (Nunnally & Bernstein, 1994). ACT-R has been thoroughly vetted as a model of basic psychological principles, and domain knowledge acquired from subject matter experts contributes to the plausibility of the model. However content validity alone is not enough; the model also needs to be assessed for predictive validity.

To ensure predictive validity, the cognitive model must be validated against human data. Typically, human subject data is obtained by bringing people into a laboratory to perform a task. Though this method is often adequate, it has several limitations. Foremost, the current model is intended to emulate the behavior of highly skilled pilots. Therefore, unlike many ACT-R models that emulate general human capabilities, this model requires a very specific subset of the population in order to be validated. Bringing real pilots into the laboratory to collect enough data is prohibitively expensive and logistically difficult. Secondly, data generated by human subjects in a lab setting is often criticized for its ecological validity, and sometimes with good reason. Although X-Plane is certified for use as an FAA flight simulator, it is only a proxy for real-world data. It is likely that data generated in a laboratory will approximate real-world behavior, but contain some amount of error.

Fortunately, there is a better alternative to the traditional approach for obtaining human subject data for this model. NASA's Safe and Efficient Surface Operations (SESO) has helped develop a surface management system called SODAA

(Surface Operations Data Analysis and Adaptation; Brinton, Lindsey, & Graham, 2010). The SODAA tool is designed to provide ground controllers with detailed information about all of the planes within their air space and, importantly, records the position of every plane as it taxis.

Human data to validate the current model was obtained using the SODAA tool at Dallas-Fort Worth (DFW) airport. This data is then fed into X-Plane, recreating a time slice of taxiing behavior from DFW. Since the ACT-R model can interact with X-Plane, it is possible to selectively remove a plane from the SODAA data and see how well ACT-R performs in its place. Thus, there are two streams of positional data that can be compared with each other: that generated from a real pilot, and that generated from the ACT-R model (see Figure 16). The trajectory of the plane generated by the model will be compared to the trajectory of a real plane following the same clearance at Dallas Fort Worth (DFW) airport.

Two taxi routes from the SODAA data were chosen for model validation. Both routes were driven using Boeing 737-800 in the SODAA data, and contain important features such as 90 degree turns, holds, and have both short and long stretches of taxiway. Data are collected from the model using an X-Plane plug-in that records position and velocity information roughly every 25 milliseconds. For a standard taxi route, this results in over 10,000 data points per model run. In the analyses below, three runs of each taxi route by the model are examined. These runs were randomly chosen after first eliminating any run containing serious errors—namely, any run in which the correct taxi route was not followed. This exclusionary criterion ensures that each model run can be analyzed with respect to all measures of validation.

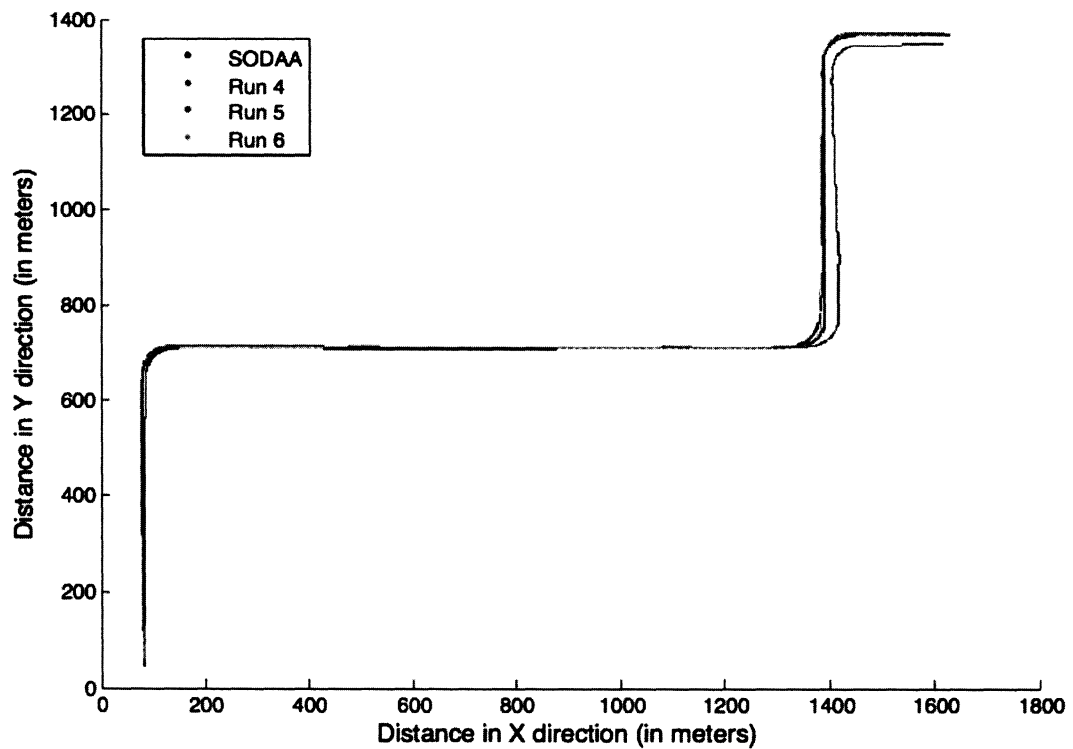
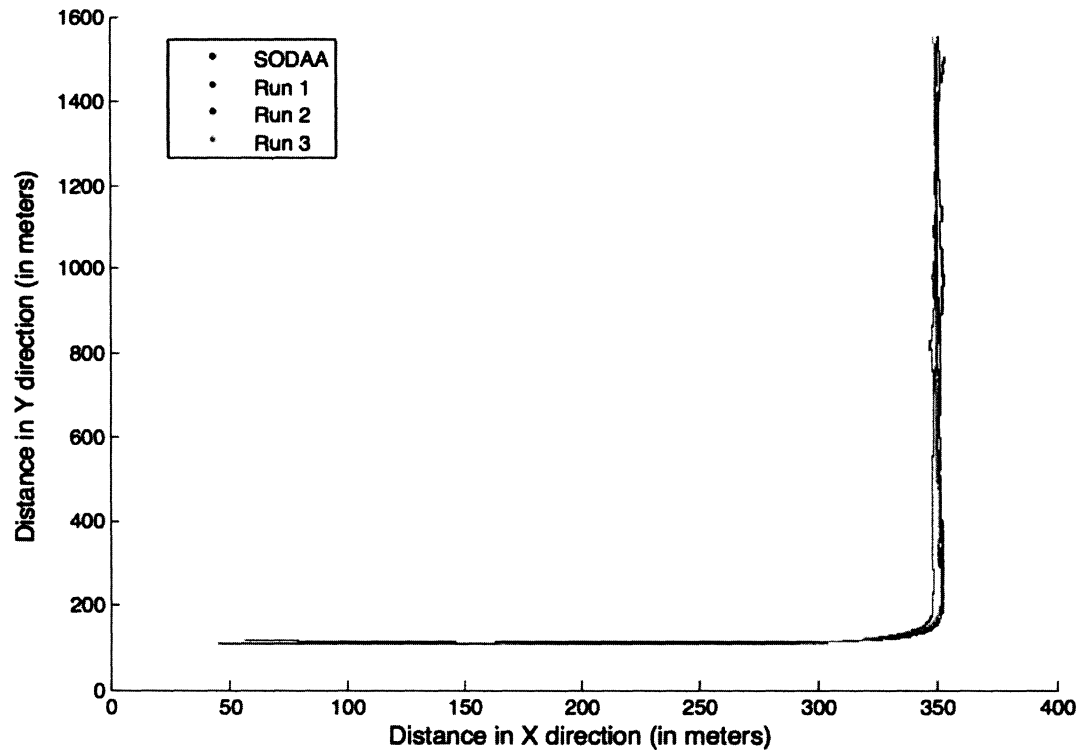


Figure 16. Trajectories generated by the ACT-R model are compared to the trajectory from the SODAA data for route one (top) and route two (bottom).



## Positional Data

**Overall fit.** An important criterion for how well the model performs is the positional fit of the model trajectory to the SODAA data. For each data point in the model's trajectory, the nearest point on the SODAA data's trajectory was found and an offset was computed. In this way, positional offset from the SODAA data can be examined irrespective of time. The results, which are summarized in Table 3<sup>5</sup> and Table 4, indicate a good fit for the first route (mean deviation of 1.99m, average max deviation of 3.62), but a worse fit for the second route (mean deviation of 5.83m, average max deviation of 19.4). However, it seems the values for the second route may be inflated due to a poor fit from Run 4. It is important to remember that the positional fit shown here is relative to the SODAA data, which may contain some error itself, rather than an "ideal" taxi route.

Table 3. Mean, standard deviation, max deviations, and RMSE from the SODAA path for route one. All values are in meters.

	<b>Run 1</b>	<b>Run 2</b>	<b>Run 3</b>	<b>Average</b>
<b>Mean</b>	1.36	2.67	1.93	1.99
<b>Stdev</b>	0.96	1.06	1.07	1.03
<b>Max</b>	3.81	6.50	4.36	3.62
<b>RMSE</b>	446.45	252.97	242.75	314.06

---

<sup>5</sup> For technical reasons, the model did not start in the identical location as its SODAA counterpart for route one, which artificially inflated the values in Table 3. For this reason, the beginning of the model data prior to convergence with the SODAA path (roughly 500 data points) was eliminated from the present analysis.

Table 4. Mean, standard deviation, max deviations, and RMSE from the SODAA path for route two. All values are in meters.

	<b>Run 4</b>	<b>Run 5</b>	<b>Run 6</b>	<b>Average</b>
<b>Mean</b>	10.58	3.72	3.27	5.83
<b>Stdev</b>	10.48	2.15	2.18	4.94
<b>Max</b>	36.12	13.09	9.00	19.40
<b>RMSE</b>	52.24	157.75	164.90	124.96

Another way to look at the position data is to look at the offset between the model and the SODAA data at each point through time. Unlike the first analysis, this approach is sensitive to the speed at which the planes are travelling. The results, shown in Figure 17, demonstrate a noticeable difference in position that grows steadily over time. Root mean square error (RMSE) is calculated for each model run and listed in Tables 3 and 4.

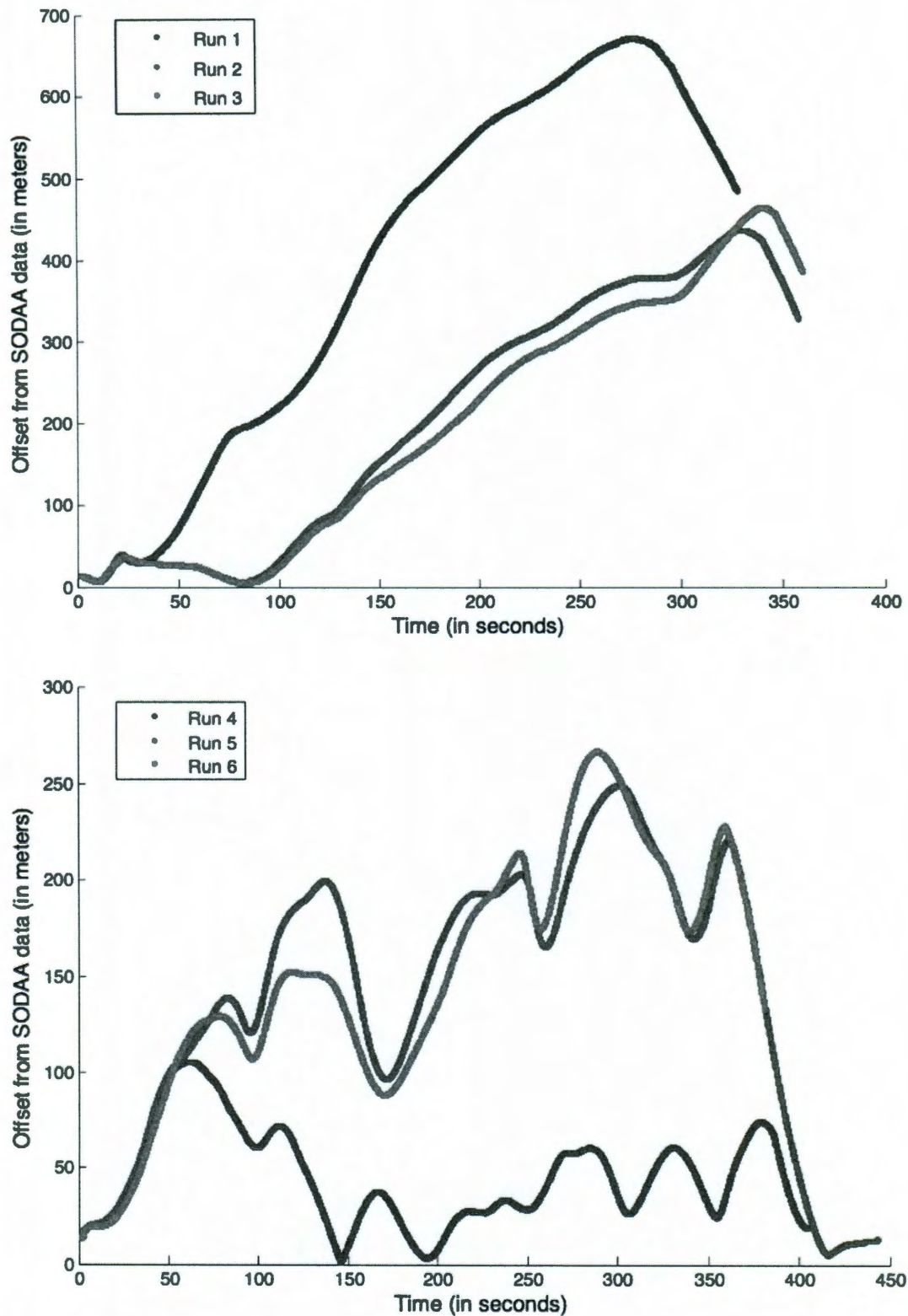


Figure 17. Offset between the model and the SODAA data for route one (top) and route two (bottom). Each line represents a single run of the model.

In light of the previous analysis, it is suspected that the RMSE is driven almost entirely by the differing velocities of the planes, covered in a later section. As a cursory examination, it is worth noting that the average speed of the model in route one is 9.22 m/s compared to an average speed of 8.15 m/s for the SODAA data. Over the course of 350 seconds, the model is likely to pull ahead by roughly 376 meters, which fits in well with the RMSE values observed here.

**Corrective steering.** Both taxi routes contained one long stretch of taxiway in which the model must stay on the centerline. For this stretch, deviations from the centerline were recorded for each model run as well as the SODAA data (Tables 5 and 6). Several of the model's runs actually outperformed the human SODAA data in its ability to stay on the centerline, while a few runs fared worse.

Table 5. Mean, standard deviation, and max deviations from the centerline for route one and the corresponding SODAA data. All values are in meters.

	<b>Run 1</b>	<b>Run 2</b>	<b>Run 3</b>	<b>Average</b>	<b>SODAA</b>
<b>Mean</b>	1.56	0.71	4.60	2.29	1.29
<b>Stdev</b>	0.80	0.52	1.26	0.86	0.59
<b>Max</b>	3.15	1.74	7.68	4.19	2.52

Table 6. Mean, standard deviation, and max deviations from the centerline for route two and the corresponding SODAA data. All values are in meters.

	<b>Run 4</b>	<b>Run 5</b>	<b>Run 6</b>	<b>Average</b>	<b>SODAA</b>
<b>Mean</b>	1.69	1.10	2.34	1.71	3.32
<b>Stdev</b>	1.15	0.69	1.43	1.09	0.65
<b>Max</b>	3.59	2.47	5.30	3.78	4.83

Previous estimates of centerline deviations are typically more constrained. One study measuring centerline deviations for a Boeing 747 suggested an average

deviation of roughly .15 meters and a standard deviation of .6 meters, though a maximum deviation of over 6 meters (Cohen-Nir, & Marchi, 2003). Nonetheless, there are several reasons to believe the levels of deviation reported here are more than adequate for modeling purposes.

First, even the SODAA data does not reach the level of accuracy suggested by Cohen-Nir and Marchi. This is likely a result of noise inherent in the data recording process. However, it stands to reason that it is unnecessary to outperform the level of the recording equipment used by ADS-B. Second, the aforementioned study looked only at Boeing 747 aircraft, whose wingspan is roughly 23 meters larger than the Boeing 737-800 used in the model. The smaller plane size allows for significantly more leeway in centerline deviations. Lastly, while such deviations may not be acceptable for *real* taxiing, they should be acceptable for most computer simulations.

**Turns.** Another important characteristic of the model is how well it can perform turns. Figures 17 depicts a close-up of two turns as performed in routes one and two, respectively. Figure 18 shows the heading of each plane as it performs a 90 degree turn. The model heading correlates highly with the heading of the plane from the SODAA data through time (all  $r > .85$  for route one; all  $r > .98$  for route two)<sup>6</sup>. However, in top part of Figure 19, it is apparent that part of the discrepancy is that the model routinely starts turning earlier (in time) than the human pilot. This

---

<sup>6</sup> The SODAA data contained significantly less data points compared to the ACT-R model, due to the rate at which it samples heading information. To correct for this, all data was refitted by interpolating each vector to contain an equal amount of data points. This ensures that all data points are properly lined up in temporal order for correlational analyses.

can be corrected by shifting the human pilot curve back 10 seconds, resulting in an even better fit (all  $r > .99$ ) for route one.

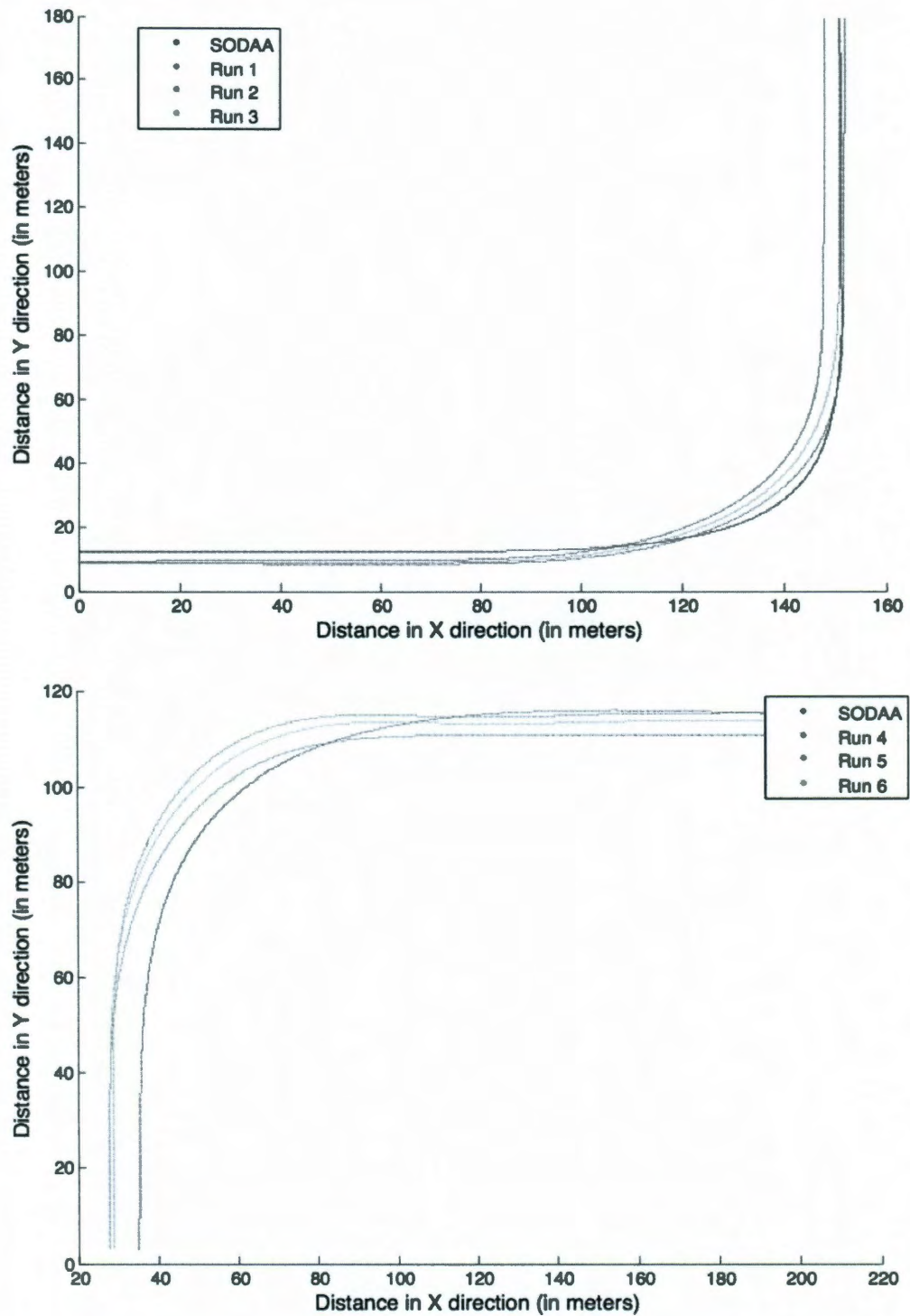


Figure 18. A 90 degree turn is performed in route one (top) and in route two (bottom).



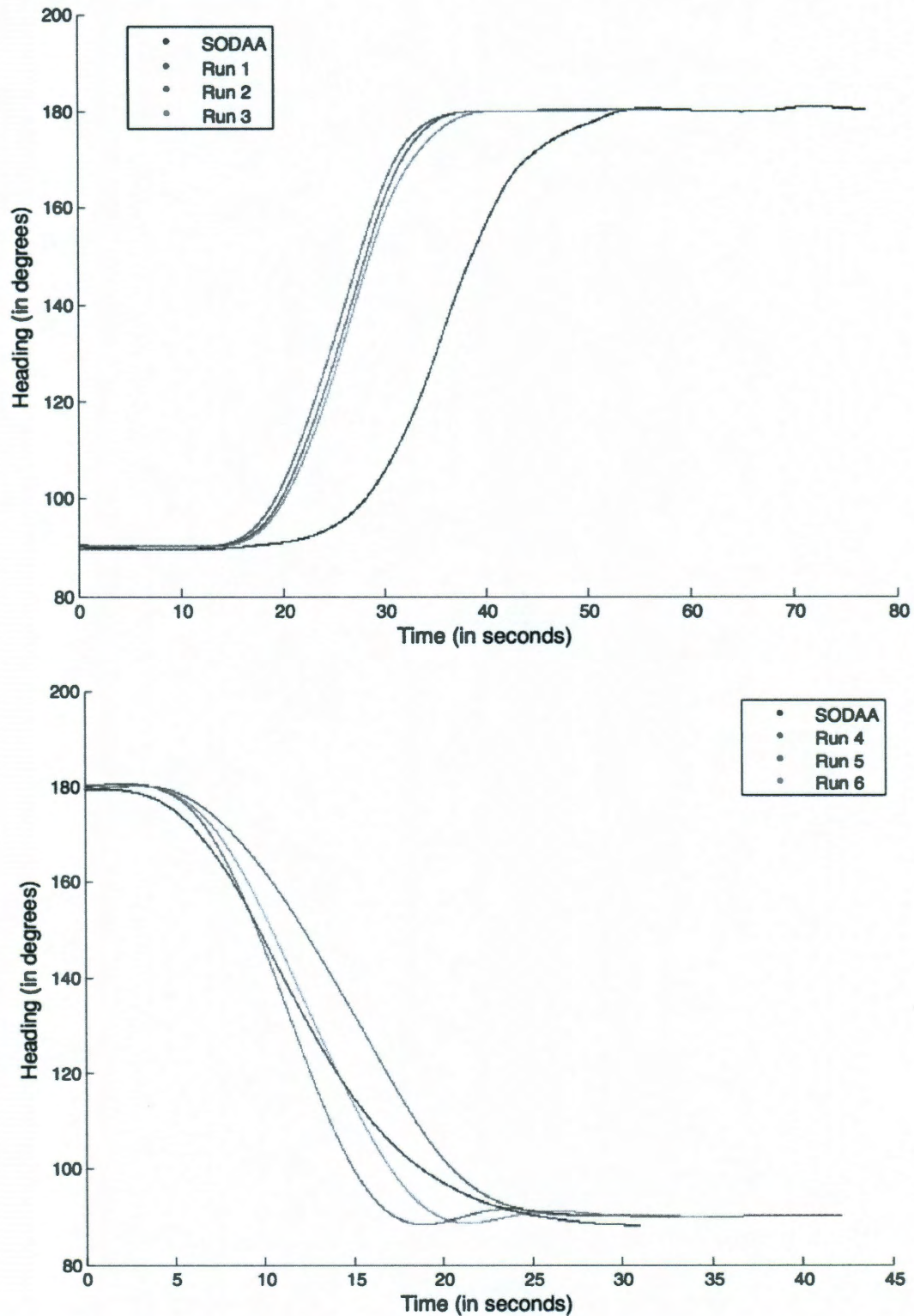


Figure 19. The heading of a plane as it performs a 90 degree turn is shown for a turn in route one (top) and route two (bottom).

Another way to examine performance on turns is to look at the heading of a plane by its position, instead of time. This analysis is perhaps more telling, as one would expect the model and SODAA data to have the same heading at any spot along the turn, regardless of velocity; the same might not be said of a plane's heading through time. Again, the model's heading correlates very highly with the SODAA data (all  $r > .99$ ). However, Figure 20 makes it apparent that the turns are not quite identical. This figure plots the difference between the model's heading and the SODAA heading by position, as it approaches the turn. Initially, the headings are almost identical, but they begin to deviate towards the end of the turn. This is a result of the SODAA data performing a wider turn than the model, as can be seen in Figure 18. In all, however, the model seems to do an excellent job at performing turns.



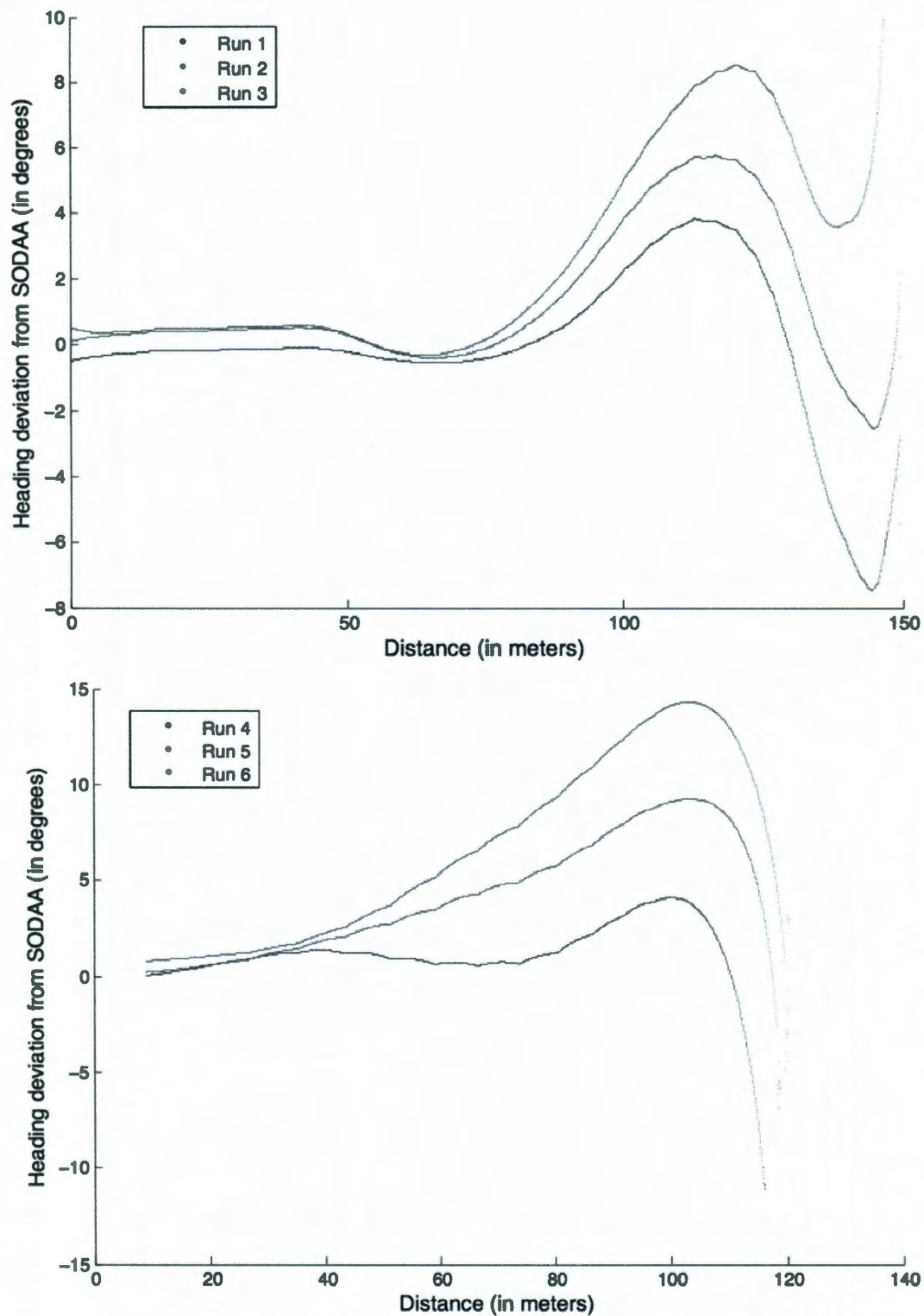


Figure 20. The difference between the SODAA heading and the model heading is plotted against the planes position as it makes a 90 degree turn for route one (top) and route two (bottom).

## Velocity

Velocity control is an important component of the model. The ability to control the plane's speed in a human-like way differentiates this model from deterministic models such as SOS<sup>2</sup>, and provides an opportunity for future research involving 4D taxi clearances.

Overall, the model's average speed for route one was 9.22 knots (including holds) compared to an average speed of 8.15 knots in the SODAA data, and 13.33 knots compared to an actual 15.18 knots for route two. The pattern of acceleration and deceleration (shown in Figure 21) was measured by a correlation between the velocities of model and the SODAA data. The model performed respectably, with average correlations of .61 and .57 for the two routes.

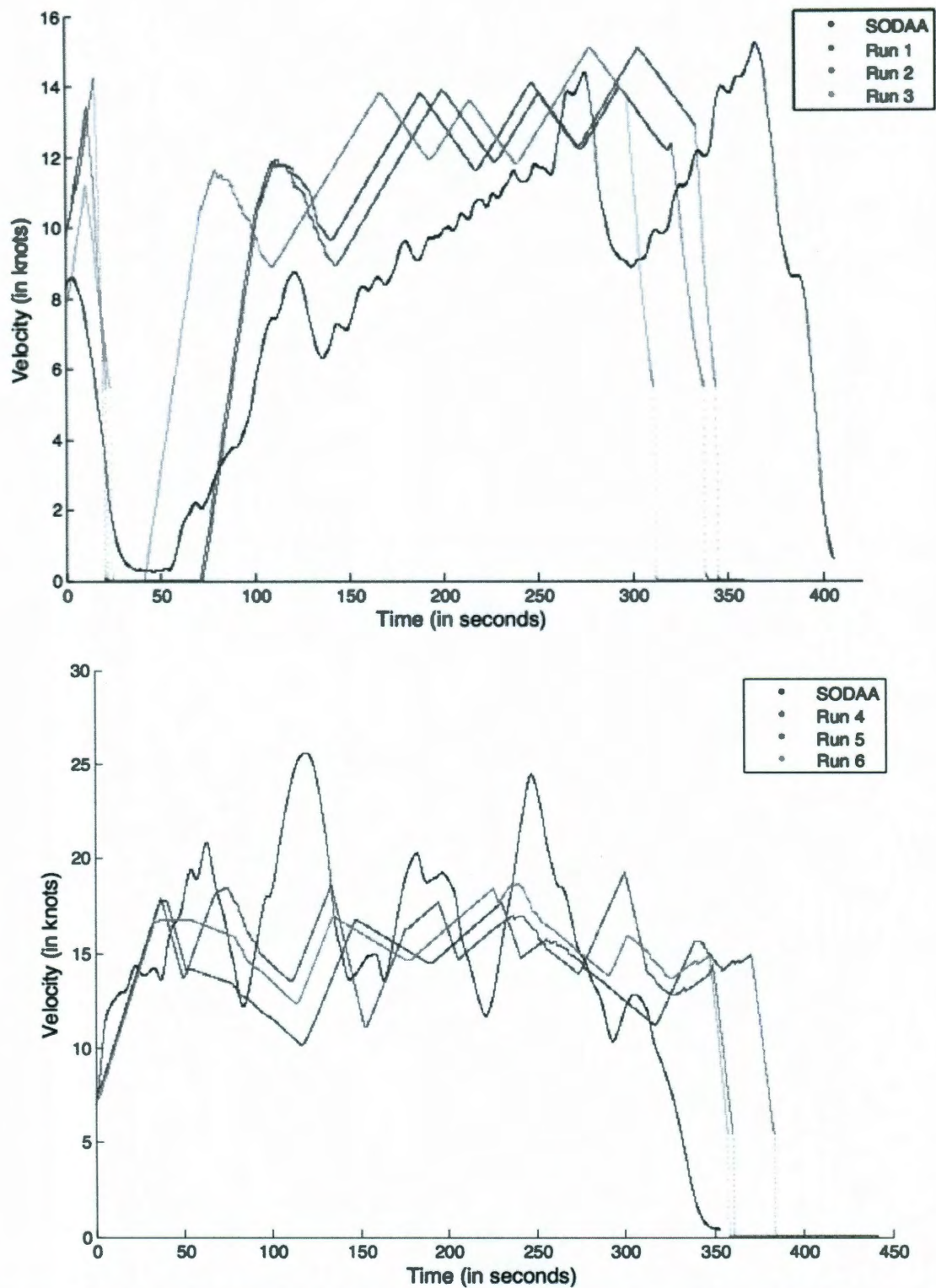
In route one, the model tended to run a bit faster than the SODAA data, while in route two the model tended to run a bit slower. Although the fit could be improved by modifying the target speed of the model, this would deviate from the advice of subject matter experts, and also risk overfitting a small set of model runs at the expense of generalization to novel taxi routes. However, if one has data regarding the distribution of individual differences in taxiing speed, the target speed parameter could be used to represent various pilot profiles.

Table 7. Average speed and time of the model compared to the SODAA data for route one, plus the correlation and RMSE between model velocity and SODAA velocity. Speed values are in meters per second.

	<b>Run 1</b>	<b>Run 2</b>	<b>Run 3</b>	<b>Model Average</b>	<b>SODAA</b>
<b>Speed</b>	8.87	9.08	9.71	9.22	8.15
<b>Time</b>	4:57	5:40	5:48	5:28	6:32
<b>Correlation</b>	.72	.78	.32	.61	n/a
<b>RMSE</b>	3.79	3.55	5.52	4.29	n/a

Table 8. Average speed and time of the model compared to the SODAA data for route two, plus the correlation and RMSE between model velocity and SODAA velocity. Speed values are in meters per second.

	<b>Run 4</b>	<b>Run 5</b>	<b>Run 6</b>	<b>Model Average</b>	<b>SODAA</b>
<b>Speed</b>	13.34	12.37	14.28	13.33	15.18
<b>Time</b>	6:26	6:03	6:00	6:10	5:34
<b>Correlation</b>	.47	.72	.51	.57	n/a
<b>RMSE</b>	4.96	5.19	4.58	4.91	n/a



## General

The most worrying aspect of the model is the number of qualitative errors it predicts—namely, it overpredicts the number of missed or late turns. While the model should predict these error types, one would expect them to be much less frequent than they are. In a sample of 30 turns, 20 of the turns were performed adequately (66%), 2 of the turns were started too late to be successful (7%), and 8 turns were not performed at all (27%).<sup>7</sup>

Much of this is due to the inadequacy of the current goal structure. While top-level goals are assigned utility values based on their importance, there are several components that are not included in the current model. For instance, the model has no concept of inhibition of return for previous goals. If a top-level goal is selected and then executed, the same top-level goal is just as likely to be chosen a consecutive time as any other top-level goal. That is, the selection of top-level goals are treated as independent events, which leads to some goals being performed too often and others being ignored for too long. Additionally, the model does not keep track of the goal's current status. Ideally, upon exiting goal, the model should be able to reassess the utility of the goal based on its completion status, a method of prospective goal encoding (Trafton, Altmann, Brock, & Farilee, 2003). If a goal is thoroughly complete, its utility should be low so that it is not executed for a while,

---

<sup>7</sup> This error rate can be improved significantly by modifying the utility values of the top-level goal productions. For instance, by raising the utility value of the *navigate* production from 2.0 to 2.3, the model successfully performed 26 turns (86%), 2 late turns (7%), and 2 missed turns (7%). This adjustment may have a deleterious effect on the other goals; optimal utility values have not yet been determined. The error rates presented in the text reflect the error rates achieved using a *navigate* utility of 2.0 in order to be consistent with the other data reported.

whereas if an early exit from a goal is required (and thus the goal is only partially satisfied) a higher utility would ensure that the goal is readdressed in a timely manner. These concerns may be addressed by incorporating a more sophisticated goal selection mechanism (Trafton & Altmann, 2002).

Another consideration is that each goal is self-contained: once a goal is initiated, a sequence of productions will fire without much regard for exogenous attention. That is, the current goal structure relies too heavily on a top-down goal hierarchy, rather than being influenced by feedback from bottom-up events in the environment. Part of the problem lies in ACT-R itself, which has no mechanisms for multiple object tracking (Pylyshyn & Storm, 1988) or multiple attentional loci (McMains & Somers, 2004; Kramer & Hahn, 1995), which makes it difficult for ACT-R to notice external events while it is busy with other tasks. It may be possible to attenuate this problem by using a modified version of EMMA (Salvucci, 2001), a more sophisticated model of visual attention for ACT-R. Alternatively, it may be possible to enable more efficient task-switching, resulting in lower latencies goal checks, using threaded cognition (Salvucci & Taatgen, 2008). In sum, the extant model appears to be approaching its limits with respect to multi-tasking behavior. Future versions of the model that demand even more from the simulated pilots may require an overhaul of the model's multitasking mechanism.

As mentioned previously, ACT-R is also capable of generating many other dependent variables, such as eye movements, which can be used to validate the model. However, the SODAA data does not provide the necessary data to verify these values. In future experiments, these dependent variables may be of more

importance. For instance, one may wish to look at the eye gaze of a pilot after adding new cockpit instrumentation, or when using a heads-up display.

### **Discussion**

While NextGen promises to make great improvements to our air transportation system, there is a critical gap in the methodologies used in the research and development of new technologies. Computer simulations that do not incorporate human cognition fail to capture an important component of the system, which may lead to faulty predictions. Human-in-the-loop simulations are intended to complement such research, but are often very costly and sometimes fail to capture large-scale emergent behavior. The cognitive model presented here addresses this gap by providing a cost-effective solution for implementing human cognition into larger simulations.

### **Potential Applications**

One potential use of the model is to train ground controllers. Currently, ground controllers may be trained in a simulated environment, in which the simulated planes are driven by real pilots. A more cost effective solution would be to have these planes driven by a cognitive model. This may be particularly effective when considering the changes in air traffic control duties that are likely to result from NextGen and automated scheduling algorithms.

Another potential use of the model might be to simulate 4D taxi-clearances, which entail changes to both piloting procedures as well as cockpit instrumentation. The model might also aid in the evaluation of real-time taxi scheduling algorithms being proposed for NextGen. Though the model is currently designed to operate in

real-time through its communication with X-Plane, the model may be adapted for use in a fast-time environment such as SOS<sup>2</sup>.

The above examples highlight the advantages of using cognitive models over conventional methods, but are only meant to scratch the surface of possibilities. One great advantage of using the cognitive model is that it is modular, and can be implemented into a wide range of systems. While there are undoubtedly some situations in which the proposed cognitive model does not adequately reflect human behavior, the model can be augmented as needed to address issues and limitations as they arise. Thus, the development of the model can proceed independently of other research, and it is not tied to any particular implementation.

### **Human-in-the-loop simulations**

Although the model is intended as a tool to replace or augment human-in-the-loop testing, it also provides insight into how to design a better HITL simulation. For instance, although the model used a single set of parameters to model multiple pilots, it is clear from the SODAA data that these pilots can be better modeled using distinct parameters. HITL simulations often employ only a few pilots, which may not be enough to capture the true variability of individual differences in pilots in the real world. Care should be taken when selecting pilots for participating in HITL simulations to ensure that this variability is accounted for, perhaps even going so far as to experimentally manipulate individual differences by instructing pilots to follow certain guidelines.



## **ACT-R**

The model also tests the viability of using of ACT-R to model aviation related tasks. In all, the model re-affirms the use of cognitive architectures as a valuable and reliable tool for human performance modeling (Foyle et al., 2005). However, the model also highlights some of the weaknesses of ACT-R. Although the default motor system was augmented to properly account for motor operations, the limitations of the visual system made it difficult to model certain behaviors.

Perhaps one of the more impressive accomplishments of the current model is its ability to seamlessly interact with an external environment (X-Plane) that was not built for the purpose of cognitive modeling. Although the current work is not the first ACT-R model to interface with an external simulator, most ACT-R models interface with a custom environment written by the model authors. Using an independent external environment forces modelers to address issues that may not have arisen otherwise, and enhances the ecological validity of the model. This practice may become more prevalent in the future, as the questions modelers seek to answer become more complex and nuanced.

## **Future Directions**

The extant model still lacks some capabilities that are necessary in order to replace humans in HITL simulations. These capabilities include a method for communication between ground control and simulated pilots, a decision-making component to aid in taxi navigation, and a more sophisticated collision avoidance system that responds to nearby aircraft.

Apart from augmenting the model, several critical questions remain unanswered. While I am confident that the error rate of the model can be lowered, it remains to be seen whether the model can replicate the true error rate of off-nominal situations. Despite thousands of flights every day, the runway incursion rate in the United States remains very small at roughly one per day<sup>8</sup> (Air Line Pilots Association International, 2007). Can ACT-R accurately model such low probability events?

Additionally, the validity of the current model has been examined quantitatively, but not qualitatively by subject matter experts. Expert pilots may notice deficient aspects of the model that were not addressed in a quantitative analysis. Perhaps a Turing test, in which real pilots attempt to delineate between videos of planes driven by the model and planes driven by the SODAA data, may illuminate aspects of the model that have gone unnoticed.

## **Conclusion**

The present work may help fill an important role in the development of NextGen by providing an extensible, cost-effective method of testing that incorporates the best aspects of human-in-the-loop simulations and computer-driven simulations. Although many opportunities remain for extending the capabilities of the model, the current validation suggests that the model is capable filling this role. It is believed that the model can guide future decisions regarding the development and deployment of NextGen.

---

<sup>8</sup> Note that a runway incursion is defined as a plane located in a spot on the taxiway or runway where it is not supposed to be, in a situation that may lead to a collision. A runway incursion in itself does not necessitate a collision.

## References

- Air Line Pilots Association International (2007). *Runway incursions: A call for action* [White paper]. Retrieved from <http://www.alpa.org/portals/alpa/runwaysafety/RunwayIncursionwhitepaper.pdf>
- Altmann, E.M., & Trafton, J.G. (2002). Memory for goals: An activation based model. *Cognitive Science*, 26, 39-83.
- Anderson, J.R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22, 261-295.
- Anderson, J. R., Kushmerick, N., & Lebiere, C. (1993). The Tower of Hanoi and goal structures. In J. R. Anderson (ed.), *Rules of the Mind*. Hillsdale, NJ: L. Erlbaum, p. 121-142.
- Anderson, J.R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.
- Bakowski, D. L., Foyle, D. C., Hooey, B. L., Kunkle, C. L., & Jordan, K. P. (2011). NextGen flight dec surface trajectory-based operations (SBTO): Speed-based taxi clearances. *Proceedings of the Sixteenth International Symposium on Aviation Psychology*, 44-49. Dayton, OH: Wright State University.
- Balakrishnan, H., & Jung, Y. (2007). A framework for coordinated surface operations planning at Dallas-Fort Worth International Airport. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. Hilton Head, SC.
- Ball, J. T., Gluck, K. A., Krusmark, M. A., & Rodgers, S. M. (2003). Comparing three variants of a computational process model of basic aircraft maneuvering.

- Proceedings of the 12th conference on Behavior Representation in Modeling and Simulation*. Orlando, FL: Institute for Simulation and Training.
- Ball, J., Heiberg, A., & Silber, R. (2007). Toward a large-scale model of language comprehension in ACT-R 6. *Proceedings of the 8th International Conference on Cognitive Modeling*. Ann Arbor, Michigan, USA.
- Brinton, C., Lindsey, J., & Graham, M. (2010). The Surface Operations Data Analysis and Adaptation tool: Innovations and applications. *Proceedings of the 29th Digital Avionics Systems Conference*.
- Byrne, M.D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies*, 55, 41-84.
- Byrne, M.D., Kirlik, A., Fleetwood, M.D., Huss, D.G., Kosorukoff, A., Lin, R., & Fick, C.S. (2004). A closed-loop, ACT-R approach to modeling approach and landing with and without synthetic vision system (SVS) technology. *Proceedings of the Human Factors and Ergonomics Society 48th Annual Meeting*, 2111-2115. Santa Monica, CA: Human Factors and Ergonomics Society.
- Byrne, M.D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology*, 15, 135-155.
- Cohen-Nir, D., & Marchi, R.F. (2003). Preliminary Analysis of Taxiway Deviation Data and Estimates of Airplane Wingtip Collision Probability. *Transportation Research Board Annual Meeting*.

Federal Aviation Administration (2008). NextGen Defined. Retrieved December 3<sup>rd</sup>, 2010 from

[http://www.faa.gov/regulations\\_policies/reauthorization/nextgenvideos/](http://www.faa.gov/regulations_policies/reauthorization/nextgenvideos/)

Flathers, G.W. (1987). *Development of an air ground data exchange concept: Flight deck perspective*. (NASA Contractor Rep. No. 4074). Hampton, VA: NASA Langley Research Center.

Fleetwood, M.D., & Byrne, M.D. (2006). Modeling the visual search of displays: A revised ACT-R model of icon search based on eye-tracking data. *Human-Computer Interaction, 21*, 153-197.

Foyle, D.C., Hooey, B.L., Byrne, M.D, Corker, K.M., Deutsch, S., Lebiere, C., Leiden, K., & Wickens, C.D. (2005). Human performance models of pilot behavior. *Proceedings of the 2005 Human Factors and Ergonomics Society*. Orlando, FL.

Foyle, D.C., Hooey, B.L., Kunkle, C.L., Schwirzke, F.J., & Bakowski, D.L. (2009). Piloted Simulation of NextGen Time-based Taxi Clearances and Tailored Departures. *Proceedings of the 2009 IEEE/AIAA Integrated Communications, Navigation and Surveillance Conference (ICNS)*. Arlington, VA, May 2009.

Goodale, M.A., & Milner, D. (1992). Separate visual pathway for perception and action. *Trends in Neurosciences, 15*, 1, 20-25.

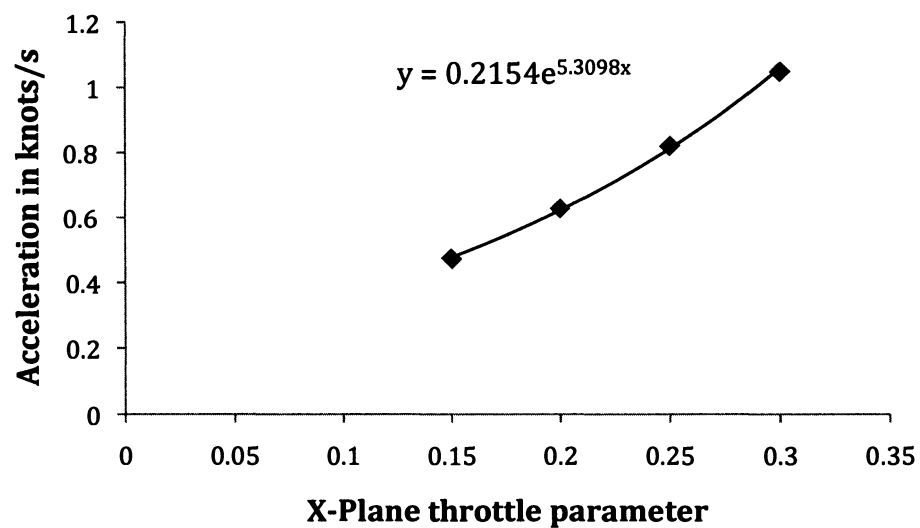
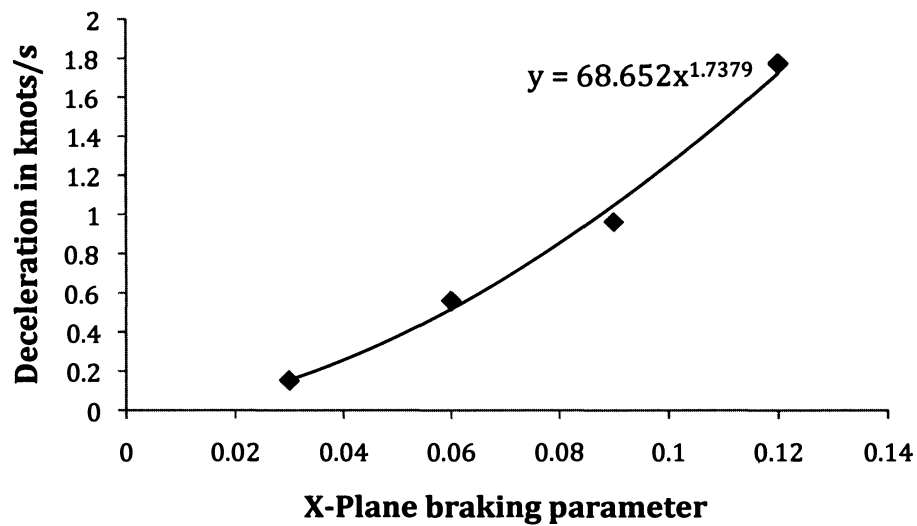
Griffiths, T.L., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J.B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in Cognitive Sciences, 14*, 357-364.

- Kerns, K. (1991). Data-link communication between controllers and pilots: A Review and synthesis of the literature. *International Journal of Aviation Psychology, 1*, 181-204.
- Kieras, D.E., & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*, 391-438.
- Kramer, A.F., & Hahn, S. (1995). Splitting the beam: Distribution of attention over noncontiguous regions of the visual field. *Psychological Science, 6*, 381-386
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.
- Malik, W., Gupta, G., & Jung, Y.C. (2010). Managing departure aircraft release for efficient airport surface operations. *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Toronto, Canada.
- McClelland, J.L., Botvinick, M.M., Noelle, D.C., Plaut, D.C., Rogers, T.T., Seidenberg, M.S., & Smith, L.B. (2010). Letting structure emerge: Connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences, 14*, 348-356.
- McMains, S.A., & Somers, D.C. (2004). Multiple spotlights of attentional selection in human visual cortex, *Neuron, 42*, 677-686
- Moray, N. (1967) Where is capacity limited? A survey and a model. In A. Sanders (Ed.) *Attention and performance*. Amsterdam: North-Holland.

- NextGen Integration and Implementation Office (2010). NextGen implementation plan. Retrieved December 3<sup>rd</sup>, 2010, from [http://www.faa.gov/about/initiatives/nextgen/media/ngip\\_3-2010.pdf](http://www.faa.gov/about/initiatives/nextgen/media/ngip_3-2010.pdf)
- Nunnally, J.C. & Bernstein, I.H. (1994). *Psychometric theory* (3rd ed.). New York: McGraw-Hill.
- Pylyshyn, Z. (1989). The role of location indexes in spatial perception: A sketch of the FINST spatial-index model. *Cognition*, *32*, 65-97.
- Pylyshyn, Z.W., & Storm, R.W. (1988). Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, *3*, 179-197.
- Rescorla, R.A. & Wagner, A.R. (1972) *A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement*, Classical Conditioning II, A.H. Black & W.F. Prokasy, Eds., pp. 64–99. Appleton-Century-Crofts.
- Salvucci, D.D. (2001). An integrated model of eye movements and visual encoding, *Journal of Cognitive Systems Research*, *1*, 201-220.
- Salvucci, D.D. (2006). Modeling driving behavior in a cognitive architecture. *Human Factors*, *48*, 362-380.
- Salvucci, D. D., & Gray, R. (2004). A two-point visual control model of steering. *Perception*, *33*, 1233–1248.
- Salvucci, D.D., & Taatgen, N.A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*, 101-130.

- St. Amant, R., Horton, T.E., & Ritter, F.E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction, 14*, 1-24.
- Trafton, J.G., Altmann, E.M., Brock, D.P., & Farilee, E.M. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies, 58*, 583-603.
- Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12*, 97-136.
- Wickens, C.D. (1984). Processing resources in attention. In R. Parasuraman & D.R. Davies (Eds.), *Varieties of attention* (pp. 63-102). New York: Academic Press.
- Wood, Z., Kistler, M., Rathinam, S., & Jung, Y. (2009). A simulator for modeling aircraft surface operations at airports. *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Chicago, Ill.*





Appendix 1. These graphs and equations can be used to translate X-Plane braking and throttle parameters into more meaningful units, expressed as acceleration or deceleration in knots per second.