1.0 Introduction and Purpose
In NASA’s position as an aeronautical technology supplier to the United States, they are trying to anticipate the demand for personal air vehicles by developing technology to facilitate this emerging and necessary mode of transportation. Limitations of the current ground and air transportation systems combined with expanding population and demand for affordable mobility will drive the development of future transportation technology and policy. There is a lot of information available on what a Personal Air Vehicle (PAV) could be and how it would likely be used as a component of the transportation system-of-systems (See Figure 1). In short, the PAV envisioned can be summarized as a mass-produced, user-friendly version of current general aviation small aircraft. Such a vehicle would generally fill a transportation niche between commercial aircraft and automobiles because it would have greater speed and range than an automobile and would have better on-demand utility than airlines. The airframe and propulsion technology for PAVs is currently available, so most of the technology effort has been focused on making this type of vehicle accessible and appealing to a larger group of people in terms of price, utility, ease of use, and safety.

There are various anticipated problems related to an increased utilization of the national air space by small aircraft. Increased noise, engine emissions, and air traffic density are examples of problems that would likely result from widespread use of PAVs. Currently, dense air traffic is handled by Air Traffic Control (ATC) where aircraft pilots are in radio communication with a human controller who routes aircraft using their positions and headings via radar tracks, transponder signals, etc. It is expected that this system, as it is now, will break down with the introduction of thousands of additional PAVs, each depending on ATC for navigation through dense air traffic. RhinoCorps has been given the task of looking at how this additional air traffic might be efficiently handled.

2.0 Task and Approach
When developing tactics for navigation in a hypothetical airspace densely filled with PAV traffic, explicit air traffic control immediately seems like a complicated and daunting approach. This may be what it comes down to, but it’s worthwhile to use a computer simulation to try out simpler approaches first. To this end, our initial tactics for PAV navigation are comparable to the way on-demand automobile traffic navigation is accomplished. Each PAV is under its own control, and it follows certain “rules of the road” which allow other PAVs to cooperatively utilize the same airspace, similar to the way automobile traffic uses road networks. We started out with the goal of seeing how far we could take this set of free flight navigation tactics to determine at what level of traffic density problems start to occur.

For this study we are primarily interested in developing and testing ideas on how to manage a large quantity of PAV air traffic in a given volume. With this in mind, some assumptions were made to help focus the simulation efforts. In order to concentrate on developing navigation tactics, various hardware or technology
components were not simulated explicitly. For instance, no communication equipment is simulated in detail. There are no antenna patterns or signal-to-noise calculations being performed, nor are frequency bands, communication networks, message routing, transmission delays, busy channels, message transmission delays, etc. modeled. We assume that the PAVs are able to communicate with whomever they need to, and exactly how this physically occurs is unimportant for our purposes. Similarly, we assume that PAVs know where they are and know where nearby PAVs and other obstacles are. Whatever specific sensor hardware or data fusion technology is needed to accomplish this situational awareness is not modeled in detail. We also assume that PAV pilots are competent and can make the PAV maneuver according to their decisions within the aerodynamic limitations of the vehicle. Human thought processes are modeled and indecision can be seen in the way some PAVs navigate, but once a pilot decides on a maneuver, the only limitations are the vehicle performance parameters. Again, the specific glass cockpit technology, haptic feedback interface, engine performance, fuel consumption, etc. are not specifically simulated.

Simajin® ("sim-aj-in" simulate + imagine) is an event stepped simulation with a proven history. The primary advantage of this type of simulation architecture when used in this application is speed. Interactions between entities are decoupled as much as possible so that only required calculations are performed. A contrasting simulation architecture is the time stepped simulation, where all possible calculations are performed for each increment of time. For example, if an aircraft takes off and has an uninterrupted path for 5 minutes before it has any possible interaction with another entity within the simulation, then an event is scheduled for that time in an event stepped architecture. Simajin jumps from one such event to another, skipping the interspersed intervals of time when nothing of note occurs. For this example aircraft, a time stepped simulation would calculate state updates for every time increment within the 5 minute period. Time stepping quickly becomes impractical \((O(n^2))\) as the number of simulated entities and length of game time increases. Simajin can be thought of as a just in time approach to simulation computations as opposed to a just in case approach.

Using a 32-bit memory architecture (~4GB), Simajin can simulate roughly 100k separate, moderately detailed entities. There is a wealth of information available on the simulation technology used in Simajin and its predecessors (such as JIMM, SUPPRESSOR, SWEG, etc.).

3.0 Scenario
The simulation scenario is the setting, or environment (playground) in which the simulated entities (players) are placed. The scenario is first established by specifying its physical geometry, time, and date. Then geographic features like terrain and man made structures are added, followed by other objects of interest, and various players. The scenario also establishes the initial conditions of the simulation.

3.1 Morning Commute
New York City (NYC) was selected for the first PAV simulation scenario, as it represents a worst-case test for PAV navigation tactics. Specifically, the area of interest is a rectangular lat/long section centered on New York City, extending roughly 100 miles in each direction (approximately 200 by 200 miles square). PAVs are more suited to point-to-point navigation where mass transit mechanisms have not been previously established, so free flight navigation tactics are stress tested by choosing the New York City morning commute scenario where many PAVs from surrounding areas are concentrated as they approach their destinations downtown during a typical morning commute. As one would expect, there is still some PAV traffic headed out of the city, and some traffic going point-to-point elsewhere within the 200x200 mile area, but the downtown destinations in our scenario are weighted heavier so that the general trend of commuter traffic is into downtown New York City.

There are several major airports in the NYC area that support commercial air traffic throughout the western hemisphere. Data for this traffic was extracted from Global Commercial Flight Database supplied by NASA LARC. This world wide data was pruned down to only include flights going to or coming from the commercial airports
in the NYC area. John F. Kennedy, Newark International and Laguardia airports are the largest of these commercial airports. These commercial flights are used to more accurately render the environment that PAVs will negotiate.

The simulation scenario is set up to start at midnight when the data for commercial flights starts, and continues through a morning commute until 10:00AM. The first 6 hours of the simulation mainly serve to establish the commercial air traffic pattern, particularly with respect to commercial air traffic inbound to the NYC area. PAVs begin to take off at 6:00AM.

As part of avoiding commercial traffic and commercial airports, PAVs in the scenario only take off and land at what we will term: PAV ports. PAV ports are comprised of existing small general aviation (GA) airports and theoretical downtown locations. Existing airport data was procured from Federal Aviation Administration (FAA) National Aeronautical Charting Office (NACO) (www.naco.faa.gov) publications on the internet and using python scripts it was pruned, sorted and converted into our simulated PAV port locations. The airport pruning process was based on an airport having a location within the 200x200 mile area, and having an airport identifier that doesn't send or receive any commercial traffic as described by the Global Commercial Flight Database. Only Heliports were excluded from being converted into PAV ports: helicopter pads are typically 50x50 feet and therefore too short to be useful for Very Short Take Off and Landing (VTOL) aircraft that this study encompasses. All other types of entries were included: general aviation airports, blimp bases, seaports, gliderports, etc. which all have enough land or water area to ostensibly support PAV operations.

In order for PAVs to have more destinations available in the downtown area, theoretically possible PAV ports were added. Physically, these types of PAV ports could be barges on the river, parking structures with rooftop landing facilities, or similar theoretical concepts. These downtown PAV ports were placed in the simulation scenario by hand (as opposed to processing available data) along the rivers and bays, at subway and train line termini, and in other areas that looked relatively open in United States Geological Survey (USGS) (www.usgs.gov) aerial photographs of the area. Internet sites showing train and subway lines, and aerial/satellite photographs provided the data that was used to manually place these PAV ports.

USGS 30 meter resolution terrain data was downloaded from the internet in separate DEM files, assembled into a single contiguous file covering the scenario, and then cropped to the confines of the 200x200 mile area. The full resolution terrain uses about 800MB of computer memory. Running the simulation with full resolution terrain would require computers with a large amount of ram, thus significantly limiting the number of (desktop) machines on hand to perform simulation runs for study purposes. For this reason, the terrain data interval was scaled by both 2 and 4 and elevations interpolated to use fewer points, and less memory. Comparisons were done between results from simulations using high resolution terrain and those using lower resolution(s), and no significant difference in the outcomes (number of incidents, impacts with the terrain, etc.) were evident.

Data for state boundaries was also downloaded from the internet in a latitude/longitude text file. This data was converted to Simajin database format and cropped to the area of interest with a custom Java conversion utility. The state boundaries correlate well with the terrain data along the coast and river valleys, which helps demonstrate the validity of the freely available data. The state boundaries are only for visualization purposes since they are political boundaries that have no bearing on PAV air traffic. Their file is only loaded when graphics are enabled, so memory usage was not a concern for generating study matrix data.

3.3 PAVs/TAXIs

In the simulation, two primary types of private aircraft were simulated: PAVs and TAXIs. A PAV is an on demand Personal Air Vehicle and is used by one to three people for commuting purposes. A TAXI is a privately owned, for hire Air Taxi that is larger than a PAV and carries four to six people. In this document PAV is sometimes used in a more general sense to include both PAVs and TAXIs. Specific representative aircraft
capabilities were supplied by NASA LARC and are used in the simulation to establish limits for max/min speed, climb/dive rate, altitude, turn rate, and landing and takeoff distances. In general, PAVs fly slower than TAXIs, and they fly at lower altitudes. This works well in combination with the kinds of trips typically made by each class of vehicle. PAVs usually make shorter trips within the scenario while TAXIs make longer trips at higher altitudes.

Each trip is generated automatically using Python scripts with the following parameters: a trip is initiated by a PAV or TAXI taking off from a PAV port that is selected randomly from a list of “origin” PAV ports, with the goal of landing at another PAV port that is selected randomly from a list of “destination” PAV ports. Trips are then filtered by range and biased towards origins and destinations with a higher weight. PAV trips less than 30km and TAXI trips less than 90km are deemed unrealistic and are not generated.

Certain PAV ports are weighted heavier (they have a greater probability of being selected from the respective list) as a means of establishing general traffic trends while still generating all the traffic in a stochastic and automated fashion. For example, PAV origins in areas with higher per capita incomes were weighted heavier so that more traffic comes from those areas. Demographic data from the US Census Bureau web site (http://factfinder.census.gov) as well as the number of available PAV ports within each census tract was used to adjust the weight of each PAV port as an origin. Similarly, PAV ports in the downtown area were weighted significantly more on the destination list. This establishes a morning commute scenario where the general bias in traffic direction is from outlying areas towards destinations in the downtown NYC area.

It should be noted that each trip begins at takeoff and ends at landing. Ground transportation prior or subsequent to the PAV trip itself is not modeled, nor is preflight, fueling, or other time consuming components of a complete one-way commute. Door to door commute times could derived by combining an average for ground transportation, preflight, etc. with actual trip times from the simulation. Queuing for takeoff is not modeled. The takeoff rate at each PAV port is limited by a Poisson distribution of time from one takeoff to the next. The probability of takeoff each minute was varied to adjust the traffic density over time for different numbers of total PAVs and TAXIs.

Total PAV and TAXI traffic is a variable used in many of the simulation runs, and is achieved by scaling the number of PAVs and TAXIs at each origin and the probability of takeoff until the desired total trip count is reached. Traffic generated this way is still subject to the trip conditions described above. The numbers of TAXIs and PAVs generated can be varied independently, but the volume of TAXI traffic was generally thought to be smaller than that of PAV traffic, so the range of possible TAXI traffic is set at 10% of the PAV range.

3.4 Sensors

Each PAV and TAXI is equipped with up to three different sensors with adjustable ranges and update rates. From a runtime standpoint, it is desirable to limit the sensor ranges and update rates. It is desirable to have sensor ranges only as large as required in order to limit possible interactions that need to be computed by the simulation engine. In reality, radar might see hundreds of miles, but to equip each PAV with a sensor that can see that far is both unrealistic and computationally expensive. If each PAV can see thousands of other vehicles, there is a large overhead associated with collecting perceptions and filtering out irrelevant data to make each decision. Even a sensor that sees a crowd instead of many individual entities would benefit from sensible range limitations. If you consider how far you need to look ahead to successfully navigate an automobile, you can see how limiting the sensor range can help to focus the pilot’s attention and simplify the decision making process.

The first sensor is a navigation sensor and is connected to the PAV thinker (aka decision maker, or pilot), and feeds aircraft type, location, heading, and speed data of nearby traffic into the navigation tactics. The range (volume of airspace) of this sensor is adjustable in several ways, including a simple 3D range (sphere) and a
cylinder of 2D range with altitude. We experimented with these values for PAVs and TAXIs both to optimize runtime and to maximize effectiveness of navigation tactics. For this study we settled on an offset cylinder section such that each PAV doesn’t see (or know about) other traffic if it’s far enough above, below or behind. As mentioned earlier, this is a simplified sensor that doesn’t represent sensor errors, failures, signature attenuation associated with environmental conditions, or weather patterns, etc. For our purposes, PAVs have accurate knowledge of the traffic near them (within the sensor volume).

The other two sensors have a more limited range and are only for “incident” and "collision" data collection purposes. These sensors are the most direct way to output incidents from the simulation without post processing path data of all vehicles. The incident sensor is set up as a cylinder shaped volume matching the FAA distances for defining an incident: 500 feet in altitude and 1000 feet in 2D horizontal distance. The collision sensor is the same idea, but with an even shorter range. These sensors have filters applied to them so that only the first sensor hit is logged, which means that a close encounter between two vehicles will only result in two incidents as each vehicle sees the other with its incident or collision sensor. Additionally, these sensors are only active while the PAVs and TAXIs are airborne, which omits possible encounters that would otherwise be logged during ground operations, landing and takeoff. Commercial aircraft do not have these sensors, so any incidents or collisions with commercial aircraft are only logged from the individual PAV/TAXI perspective with the PAV/TAXI sensors. This means that when comparing incident counts, incidents with commercial aircraft should be doubled (or incidents between PAVs halved).

The update rate required for this short range sensor to work properly is influenced by how far it can see and how fast another vehicle might cross through its range. For example, if the sensor is configured with a range of 300m, and the maximum possible closing speed with another object is 150m/s, then the required update rate can be estimated with a head-on collision case where one vehicle travels from just outside the sensor range in front of to just outside sensor range behind the other vehicle (600m), which would take 4 seconds. The actual rate should be set somewhat higher that this, considering that a slightly offset head-on scenario will reduce the time interval in which one vehicle is in sensor range of the other. In practice, we varied the sensor rate until incidents were missed (all else being equal). We found that doubling the rate estimated using the method presented is quite sufficient, especially when altitude bands are being used where most traffic at a given altitude has approximately the same heading and speed (lower closing speed). Giving the sensor a fast update rate slows the simulation down (increased runtime), so we would like to use the slowest rate that accurately captures the incident history.

It should be noted that complete time/path histories of each vehicle in the simulation can be recorded for a simulation run. This means that if a more sophisticated analysis of PAV incidents or interaction is needed, a post processor can be built to read the path data from the simulation and obtain the distilled output.

To get an idea of how frequent incidents are in comparison to actual collisions, we ran a set of runs with the incident sensor configured as a collision sensor with a 15m range and accordingly increased update rate. These runs used considerably more CPU time due to the rapid sensor update rates required to capture possible collisions, and we were surprised to see that about 10% of incidents could be considered actual collisions. Data for these runs is shown below in the results section. Our current explanation for the high rate of incidents/collisions is that when one PAV is focused on avoiding another vehicle, it may cross paths with two or more different PAVs that are already engaged in maneuvers to avoid yet other PAVs. This is a sort of tunnel vision and the solution to this would be for the PAV to select an avoidance maneuver based on multiple simultaneous threats, or maybe to prioritize threats by some means and react to them in order. It may also be worthwhile to relax the FAA definition of an incident to allow closer “formation” flying if PAV flight hardware is expected to permit it. Adding a heading
component or closing velocity to the definition of an incident would allow PAVs to fly with the same heading at less than 1000 feet range (highway in the sky), while still defining a head on (high closing velocity) situation at less than 1000 feet as an incident.

It is important to note that even though filters are in place to limit multiple incidents between unique entities, when it comes to actual collisions there is no mechanism in the tactics or simulation umpire functions to remove those players from the scenario following a collision. This was not implemented because it would significantly impair runtime to constantly monitor a collision sensor at a high rate. The result is that the total measured collisions may be inflated (particularly in scenarios with dense traffic) since players are able to continue towards their planned destinations after a collision, and may collide again with another vehicle later on. While this situation is unrealistic, it can be overlooked from the standpoint of developing and testing navigation and avoidance tactics because the traffic density at a given test point is maintained instead of being reduced by attrition. With this in mind, interpretation of the resulting incident data remains valid.

3.5 Tactics

Rather than try to plan various complex maneuvers ahead of time for numerous possible traffic situations, tactics were developed that allow a string of simple decisions to generate complex behavior. At any given moment there are only a few different maneuvering options available to a PAV pilot: turn, change altitude, or change speed. Breaking a trip into a series of smaller decisions and calculations performed reactively along the way allows manageable rules and tactics to be built. These types of tactics are easier to develop, test, and debug. Also, they are often more robust to changing conditions, where more comprehensive rules are likely to entangle with one another and require greater effort spent on accounting for contingencies.

PAV tactics for maneuvering are divided into two categories: navigation and avoidance. Navigation tactics are used for planning the general course of the flight (i.e. long term takeoff to landing) while avoidance tactics handle short term course deviations implemented reactively to avoid other traffic. Navigation tactics are based on prior knowledge of the destination and any intermediate obstacles, like closed airspaces. Avoidance tactics are based on new information supplied en route by the navigation sensor. Avoidance tactics take precedence over navigation tactics, such that if navigation tactics determine a course to the destination through a series of closed airspaces, an avoidance maneuver may cause the PAV to violate the closed airspace if needed. Navigation tactics are executed once on takeoff, and once after every avoidance maneuver (or series of maneuvers) to re-compute a new course to the destination. This is necessary because after an unplanned maneuver to avoid other PAV or commercial air traffic, the course needed to navigate an obstacle may be different than the original course. For example, it may be shorter to go the other way around a restricted airspace (no-fly zone) after one or more avoidance maneuvers.

Heading based altitude bands are incorporated into the navigation tactics. Following takeoff, a cruise altitude is picked based on heading. PAVs are assigned altitudes up to 10,000 feet, TAXIs have 11,000 to 20,000 feet and commercial traffic adhere to the flight paths supplied in the Global Commercial Flight Database and often remain above 20,000 feet. If a PAV's heading is changed enough due to avoidance maneuvers, a different altitude may be selected once normal navigation is resumed. Various numbers of altitude bands, as well as the heading intervals that determine the bands were experimented with. With a limited volume available for altitude bands, irregular heading intervals may help with specific regions, depending on general traffic flow. We noticed a bias towards north/south traffic due to the coastline in the NYC scenario, so we tried adding more bands for north/south headings (leaving fewer for east/west headings). Because TAXI traffic is generally going on longer trips, they were assigned the higher altitude bands. Commercial and TAXI traffic doesn’t interact with PAV traffic except for potentially during climb and descent. Terrain limitations take precedence over altitude bands, and a more mountainous or otherwise volumetrically complicated scenario would warrant further
Avoidance tactics were developed iteratively. We started with just a few PAVs in a simplistic artificial scenario (See Figure 2) to test common traffic interactions that might be experienced in the full scenario. A head-on tactic was first put in such that if conditions indicated a possible head-on collision, each PAV should turn to its right. There are many criteria that can be used to establish tactics for avoiding other traffic; some of the ones we found most useful include: 2D distance, relative heading, relative radial speed, and 3D relative location (up/down, left/right, front/back). Path crossing and merging tactics were added next, such that if two PAVs have similar but intersecting courses the PAV on the right has right of way, and the other will slow down and/or turn slightly to facilitate crossing of paths. A speed up maneuver was also experimented with, such that the PAV with right of way would speed up slightly to increase separation distance. Another slow down maneuver was implemented to see if PAVs would naturally queue up and space themselves out. This tactic states that if one PAV is following another PAV that has about the same heading and speed, the trailing PAV will slow down until a suitable distance is reached and fall in behind the leading PAV. The thought here was to space out traffic headed in the same direction at the same speed.

All avoidance tactics are based on traffic at a similar altitude. A PAV will not try to avoid another PAV if it is far enough above or below a predefined (though adjustable) range. The navigation sensor will not even return data on traffic if the relative altitude is great enough. This makes the heading based altitude band strategy (established in the navigation tactics) very important from a collision avoidance standpoint. Different strategies were tried for placing traffic with similar headings at similar altitudes vs. widely alternating them. The best solution based

---

**Figure 2** Preliminary tactics development scenario
on our testing was to have each altitude band neighbor bands based on similar headings. This means a course correction resulting in a different altitude band will result in a smaller required change in altitude. With this method there is only one large altitude/heading discontinuity left after 360 degrees, and we placed this at due west for the NYC scenario.

An avoidance based reactive altitude tactic was also experimented with. This tactic was to change altitude, ignoring the navigation specified altitude band, if local traffic was deemed too dense at the current altitude. This tactic was usually only employed in scenarios with a generally high level of traffic, and it only reduced the total number incidents by a small fraction.

The specific numbers used as parameters for many of these tactics, like the range and closing speed used to determine a likely head-on incident, were first guessed at then tested as a variable in a small study matrix. For example, a 2.0km look ahead range might be an estimate for a good starting point. Running a series of simulations where the only change is that number, say from 0.5km to 5.0km, would indicate which specific value yields the best performance, if any. The simulation outcome is sensitive to some parameters, while others may have little impact; less time was spent experimenting with the latter.

Our development of PAV tactics did not test each parameter exhaustively, but this approach helped us to quickly try new tactics by comparing back-to-back batches of runs with and without a new tactic, and also to tune tactics that showed a benefit with a series of runs.

Because current ATC resources and procedures are believed insufficient for controlling airspace filled with thousands of additional PAVs, one of the basic premises of this scenario is that there is no interaction between PAVs and commercial traffic. This reduces the anticipated impact on established ATC procedures and allows for experimentation on more innovative, modern systems to handle the large numbers of planned PAVs. In our study, commercial aircraft are given a wider space by PAV and TAXI traffic. The commercial aircraft are tied to their scheduled paths and do not try to avoid PAV or TAXI traffic, further minimizing impact on current commercial air traffic and ATC systems. Similarly, PAVs are configured to give TAXIs a little more space than they do other PAVs due to the difference in speed.

Air traffic control for problem areas identified by a high density of incidents can be implemented through the navigation tactics by letting an external entity specify destinations intermediate to the final destination that each PAV is trying to reach. This slightly generalized level of regional traffic routing would be analogous to a traffic light for automobiles. We were unable to determine how this might be best implemented in the NYC scenario. With the downtown area as the focal point of traffic and all altitude bands in that area occupied, there doesn't seem to be any free volume to work with. Having a region with rules based on time-of-day may be a possible solution to these focus points. An example of this would be a rule for downtown that lets PAV traffic approach at any altitude in the morning, while traffic departing the region must follow a smaller, possibly inconvenient corridor that is avoided by the (majority) approaching traffic. This rule would reverse or simply not apply for the evening when traffic is dispersing from the downtown focal point. Another possible application of a regional traffic rule would be west of downtown, where the majority of traffic is headed east in the morning and west in the evening. A time-of-day, region specific rule for this area might allow more altitude bands with east headings in the morning, which reverse in the evening similar to HOV lanes in many large cities today. Minority traffic would have to avoid this region or relegate to fewer altitude bands.

4.0 Study Procedure
The Simajin databases are text files that are divided into three main categories called pattern, episode, and exercise. The pattern (See Figure 3) has data that describes in detail the various entities (players, systems, etc.) that can be used in the simulation. The episode describes a physical time and place, invokes specific instances of the entities defined in the pattern, and establishes hierarchies or other specific links between players. The exercise sets parameters for the execution of the simulation, such as run speed,
desired output (graphics, binary, text, or XML), network or shared memory interfaces, etc. For example, the pattern would describe what a PAV is and what it can do; the episode creates one or more particular PAVs and establishes an environment for them to do something in, and the exercise tells the simulation how to execute the scenario, how to display what happens and what output to record while the simulation runs.

For a single simulation run, the Simanij executable is invoked with a single control file that tells Simanij where to load the database files, the binary terrain file, the language parsing binaries, and where to put any output files. To change a parameter, a user could open the appropriate text file, change a number or setting, save the file, rerun the simulation, and then look at the resulting output. This manual process quickly becomes painstaking and error prone when a lot of simulation parameters must be systematically permuted, run thousands of times on dozens of machines, and the resulting data collected and organized.

Simanij ("sim-'an-ij" simulate + manage) solves this problem by automating the entire process from modifying the text input databases to collecting the pertinent results from various computers over the network. Simanij also provides a user interface that is based on the text files it processes (See Figure 4). This feature makes the user interface easy to customize to a particular application or set of users. Setting up Simanij for use with Simanij begins with augmenting the simulation text database files with Simanij tags. A tag is an XML-like piece of text that turns a portion of the original text file into a

![Figure 3 Partial sample Simanij database file with Simanij tag](image_url)
variable in Simanij. Tags can be integers, floating points, strings, coordinates, etc. Each tag variable becomes a field in the user interface and is also now available for automatic or random permutation in batch or study matrix processing. For example, the original database may set a target “speed of 100.0 MPH.” This number could be replaced with a tag called “Speed.” At this point the database is not readable directly by Simanij because it would see the tag as a syntax error, but if the new database is loaded first in Simanij, the tag will be replaced with a number, say 110.0, and the resulting database can then be read by Simanij.

A user can run Simanij and open the text database files that have been customized with Simanij tags. Simanij presents a user interface populated with the variables from the text files it has read, and the user can quickly configure the database parameters and run the simulation with the new settings. There are four options available for processing text in Simanij. A user can generate databases, run a single run, run a batch, or design a study matrix. The Generate Database option creates and saves one set of files that have the tags replaced with values corresponding to the settings selected in the user interface. Single, Batch and Study Matrix options also generate the output files with tags replaced appropriately, but in addition these options queue up execution steps for each set of output files. The execution step can be set up to invoke any external program, which for our purposes is Simajin. This run queue can also be distributed over a network to parallelize processing of the queue entries. Each execution step includes a data extraction phase where the output files from Simajin are scanned for results of interest, which are then returned to the computer that issued the job. Data from single and batch runs is displayed in a table, while study matrix data is displayed in a matrix arranged by variable permutations. Several plotting options are available, as well as tabular format and XML for importing the data into other software.

Because significant portions of the database files are static from one run to another, they don’t need...
to be processed for tags by Simanij. By separating the databases into different files we can save memory in Simanij by only processing files with tag variables, as well as network traffic by only distributing static files to other computers once. The terrain is an example of this. The terrain doesn’t change from one run to another, so it is sent over the network once and is used in the execution step. Other large static components of the databases are handled the same way, and are often called “global” files. So for each simulation run, the made-to-order section of the database is sent over the network where it is combined at runtime with other static data already in place.

Simanij is written in java and is platform independent. To distribute simulation runs over the network, another java server program called simserver is installed and started on a separate computer that is accessible through a TCP/IP network. This is easy to do with linux and Mac OS machines; Windows can be set up by hand or with a service. A free port must be available for Simanij to communicate with the simserver, which may require changes to firewall settings.

Simanij, Simajin, and various computers around the RhinoCorps Albuquerque and Washington D.C. offices were used to collectively generate the data needed for this study. A total of 27 different computers (3 separate processors) participated in the study. About 15 computers were available full time, and the others were used in evenings and weekends. Hardware ranged from 500MHz Intel Celerons with 192MB of ram to dual processor Power PC G5s with 8GB of ram. Several older desktop and laptop computers that were retired from daily office use were loaded with linux and connected to the network. Windows, linux, and Mac OSX operating systems were used. Dual processor machines with enough disk space and ram were set up to run two or more simultaneous simulations. Hyper-threaded Intel machines also showed a slight speed benefit from multiple simultaneous runs.

These servers were used both for parametric evaluation of tactics during development, and for generation of data for this document after tactics and scenario components were sufficiently advanced. Data from these runs was saved, then imported and plotted with Microsoft Excel. Some analysis of variance and hypothesis tests of means was conducted to ensure sufficient sample sizes. Some adjustments were made on the fly, to memory array size, refresh rates and other Simanij tag settings as data started coming back from the servers. This was done to correct minor oversights or errors encountered during study matrix execution. For example, the 38,000 PAV scenarios’ output data was large enough that Java ran out of memory and all individual servers had to have a command line option added to increase the Java memory heap size. This was done in the middle of a study matrix and the failed runs were restarted. Once the servers were back on line the study continued to completion. Some parameters were added after initial runs to get more detail in transition areas, and other data points were limited due to extended run times required to generate the data. Individual computers were used to generate specific data for sections that didn't require more than a few simulation runs, and also to capture screen shots used in this document.

5.0 Results
The following sections present data and screenshots that we found interesting while building and testing the databases. There are other areas of study that could build on what we have found.

5.1 Statistical Nature of Results
Because the simulation runs relatively quickly, it is possible to run a simulation with the same initial conditions many times and generate a distribution of outcomes. This makes comparisons between simulations with different initial conditions both easier and more legitimate, particularly when the individual outcome is highly variable. When studying stochastic processes, it is more revealing and easier to analyze many results, rather than relying on a single data point to draw conclusions. With more data, representative distributions can be used (rather than assuming normal distribution), and hypothesis tests comparing two or more means (or variances) can be done with higher confidence.

5.2 Supportable Traffic Density
In general, changes to tactics were tested by running the scenario repeatedly with increasing
levels of traffic to look for the region in which the tactics begin to scale poorly. That is to say, when the incident rate begins to increase significantly faster than the rate of traffic increase. The following plots (See Figures 5, 6, and 7) show this for different altitude bands, different PAV/TAXI cruise speeds, and with and without closed airspace near major commercial airports.

5.3 Path Complexity
The complexity of paths from origin to destination can be used to estimate the general amount of avoidance maneuvers used by PAV traffic. The ratio of the actual path length of a PAV to the straight line distance from an origin to destination increases when the PAV has to turn to avoid other PAVs, avoid restricted airspace, as well as the extra distance necessary for a PAV to climb to a cruising altitude and subsequently land. Comparing this ratio under various conditions can show which conditions result in more or less course adjustment. Worse conditions result in more course corrections, a longer total path, and ultimately longer than expected flight times. The chart below shows some example figures taken from path data. The ratios were usually close to one (See Figure 8). For example, between a 1,000 PAV scenario (average: 1.049:1) and a 10,000 PAV scenario (average: 1.056:1) the theoretical to actual distance ratio increased only by 0.6%, and only 1.5% of the 10,000 PAV scenario paths were considered excessive (i.e. actual distance was 20% more than the theoretical point to point distance).

The following charts help show the complexity of the path data. The first screen shot (See Figure 9) shows just the commercial traffic going to/from the commercial airports for 10 hours of simulation time. Each flight path is partially transparent, so a brighter line indicates that more commercial flights have taken that same path. It becomes readily apparent that commercial air traffic is quite structured, as there are only a handful of inbound/outbound corridors that the

![1000' Range vs. Altitude Bands](image)

**Figure 5** 1000' incident detection range vs. altitude bands
Figure 6 500' incident detection range vs. altitude bands

Figure 7 Collision sensor vs. altitude bands
<table>
<thead>
<tr>
<th>PAV Name</th>
<th>Start Time</th>
<th>End Time</th>
<th>Travel Time</th>
<th>Actual</th>
<th>Theoretical</th>
<th>Distance</th>
<th>Cracked?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT74 0-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>SCT5 50-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT26 75-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT96 125-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT07 200-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT39 325-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT90 500-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>236 510-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT11 485-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT14 485-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>799 510-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT15 535-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT16 585-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT42 610-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>GON 635-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>HFD 650-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT19 695-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT44 720-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>N04 770-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT20 795-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>980 820-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>MMO 845-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>HVN 880-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT51 905-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>11H 940-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CT24 1000-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>CTO4 1025-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
<tr>
<td>SCT7 1050-1</td>
<td>2162.96877</td>
<td>2380.05467</td>
<td>211.085896</td>
<td>170.376109</td>
<td>172.7272754</td>
<td>1.032038</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 8** PAV flight path complexity data sample (actual distances 20% longer than theoretical highlighted in red)
commercial aircraft adhere to. This is a stark contrast to the subsequent plots of TAXI and PAV traffic (See Figures 10 and 11). Again, the brighter areas are more frequently used airspace, and indicate denser traffic regions.

5.4 Incident Density
Figures 12 and 13 show the geographic distribution of all incidents of individual runs with and without restricted airspace respectively with a 500 foot sensor detection range. Each incident is drawn as a partially transparent dot; so several incidents in the same spot will result in a brighter shade of that color. Geographic regions with a high density of incidents are possible candidates for ATC centers or other special tactics as is the case around the New Jersey/Pennsylvania border just north of Philadelphia (See Figure 10). The cause of the increased incident density is that the PAV port and trip density in that locale is higher than the rest of the scenario. The PAVs taking off from these ports all have similar destinations and the augmented traffic density taxes the rules-of-the-road tactics that are currently implemented. Since the PAVs have a general heading toward NYC, the traffic density only gets higher as more and more PAVs take off over time. The incident numbers also increase in this area since there are more intersecting flight paths with different speed vectors (merging traffic) from recently departed PAVs attempting to gain altitude.

For the purposes of this study we thought that aircraft having just left the ground have priority over cruising aircraft as climbing aircraft are theoretically at 100% throttle, attempting to climb as fast as possible and have a higher workload to manage than the cruising aircraft. This is an attempt to emulate behavior encountered with automobile traffic merging onto freeways.

Figure 9 Commercial flight paths over 10 hours, (inset: downtown NYC)
Figure 10 Point to point flight path plot for 10,000 PAVs without no-fly zones
Figure 11 Point to point flight path plot for 10,000 PAVs around no-fly zones
Figure 12 Incident density window for 20,000 PAVs without No-fly zones
Generally it is easier for both parties for the vehicle at speed to make a slight course adjustment and allow the merging vehicle to accelerate to a cruising speed (and altitude). The problem is that the traffic density is high enough in places that cruising PAVs’ course corrections cause near-by cruising PAVs to maneuver themselves which in turn results in a congested area much more difficult to negotiate.

Again, Figures 11 and 13 show the incident density and flight paths for a 10,000 PAV scenario with closed air space around the commercial airports. This restricted airspace creates bottlenecks where displaced PAVs are forced to skirt the airspace and effectively reduces the amount of navigable airspace within the 200X200 mile scenario, especially in the already congested downtown area. Since the traffic is artificially increased around those zones, they
show more incidents as the PAV density increases. Figure 14 is an example of recorded incidents around the periphery of a restricted airspace north of Hartford, Connecticut for a 20,000 PAV scenario. The arc represents 18 near miss incidents from displaced PAV traffic that would have otherwise have flown through the air space surrounding that commercial airport.

Closer investigation of where most of the incidents were occurring revealed that most of them are located near PAV ports. This indicates the need for more refined takeoff and landing tactics, possibly including holding patterns, formation landings, go-around or alternate destination selection for over-capacity destinations.

We used the FAA definition of an incident, which is two aircraft within 1,000 feet horizontally and plus or minus 500 feet in altitude. For the purposes of detecting collisions we used a 30 meter diameter sphere. Figures 5 and 7 show that for the 10,000 PAV scenario utilizing 8 altitude bands the average incident rate detected was 28.6% of total PAVs. Of those incidents collisions comprised only 4.02% of the detected incidents at 1,000 foot range, or 115 collisions. 115 collisions is only slightly more than 1% of total PAV traffic volume. These percentages increase significantly with fewer altitude bands as a way to segregate traffic. If the PAVs’ incident detection distance were refined to allow 2 aircraft within 500 feet (as discussed in section 3.4 of this report), then the reported number of incidents would drop as much as 58%, as demonstrated by Figure 6. These plots show incident total and incident rates (percent of total) for a variety of conditions. In all cases the heading based altitude bands significantly reduced the number of incidents. The free-for-all reactive altitude selection was not very successful when compared to even limited altitude band strategies. Comparing results between the graphs shows how much of an affect the definition of an incident has. For ~40k PAVs, there are many incidents according to the FAA definition (~25%), but only ~2% could be considered collisions. As mentioned earlier, the greater than unity incident rates are possible because PAVs are not removed from the scenario following an incident. Although the overall figures would be unacceptable from an operational standpoint, the tactics tend to scale well despite the increasingly cramped quarters of the 200x200 mile scenario. There is a slight knee in the curves between 5k and 10k total PAVs, but up to ~40k PAVs the graphs still show a relatively linear increase in the numbers of incidents; we were expecting more of an exponential growth.

5.5 Study Hardware and Run Time Statistics
The actual simulation work was spread over a variety of computer hardware with processors ranging from Intel 1Ghz P3 and 500Mhz PowerPC G4s up to more modern Hyper-threaded 3+Ghz Intel P4 and dual 2Ghz PowerPC G5 processors. Due to the nature of our Simajin simulation farm, overall run time was slower than if we had a dedicated farm for the simulations. In lieu of a dedicated farm, we utilized spare CPU cycles from unattended computers at RhinoCorps Albuquerque available after business hours in addition to a couple remote computers in Washington DC. Simulation run times ranged from 30 seconds per run up to 48 hours of continuous calculations. Again, runtime was affected by what applications individual employees had running on their computers, even some of the newest hardware did not perform as well as the more average computers with more CPU cycles available. The extended runtime of some simulations disabled use of employees’ desktop machines for all but the weekend where up to 48 hours of uninterrupted processing time could be harnessed. This is only an issue for similar, non-dedicated farms. Studies performed on hardware that does not have CPU cycles otherwise occupied by other applications, with 24 hours per day availability would facilitate overall study completion time.

Simulation run time increases rapidly in comparison to the total events in a simulation as can be seen in Figure 15. This is due to the overhead associated with collecting and filtering perceptions. The more PAVs in a scenario, the more sensor chances between those players, which results in more overhead. The more overhead, the fewer events per second the longer the overall runtime. We look at thousands of events per second (KEPS) to measure the
speed of a simulation. Our studies averaged about 2.3 KEPS. The smaller (400-3,500 PAV) scenarios netted a higher KEPS rating and conversely, the larger simulations netted fewer. It is interesting to note that 1.6Ghz Intel M chipsets were some of the fastest processors used in the NASA PAV study. 4.1KEPS were clocked on some of the larger simulations. This is interesting considering that the 1.6Ghz Intel M processors were running almost double the study’s average KEPS and are older, less expensive hardware more suitable for the creation of a simulation work farm than Hyper-threaded 3.2Ghz Intel P4 processors that are clocked at about the same speed (4.4KEPS) that are more expensive.

The Simajin executable, associated Java servers and databases require a total of about 900Mb of disk space and simulations up to 350 million events could be run on 512Mb of RAM. After
about 350 million events, runtime would increase dramatically as the server would start to use hard disk swap for more memory. A memory size of 700Mb would be optimal unless large amounts of data were being collected from the larger simulations. Depending on the data collected, several gigabytes more hard disk space may be required.

6.0 Applicability to Future Work
This study provides a simulation foundation in the form of database components and data processing scripts and techniques. These components would allow the New York scenario to be quickly adapted to another geographic location because much of the data would not change, and data that is specific to a different area could be adapted quickly using the automated techniques developed for the NYC area.

6.1 Scalability
The largest scenario we have run to date comprised of 85,000 PAVs in the same 200x200 mile area. This was approaching the limit of feasibility because it took several days to run on a 2.0GHz G5 processor and used about 1.5GB of ram. Scenarios with this many PAVs would run faster if the PAVs were distributed over a larger area. Based on the limits of 32bit computer architecture, the upper limit on the number of total PAVs is about 100k. Runtime for such a scenario would depend on the number of potential interactions between the PAVs, and would thus be lower (run faster) if the PAVs are less concentrated in a geographical sense, and vice versa. If needed, Simajin could be adapted to a 64bit architecture fairly quickly (2-3 months) to accommodate a larger number of players. Scenarios requiring this feature would likely be limited only by runtime then.

Figure 15 500’ Incident detection range study event average
One other limitation that we were close to reaching with the 85K PAV scenario was the 2GB limit on output file size with the standard C libraries. Limiting the output slightly by only recording PAV incidents allowed this scenario to run to completion. If needed, this limitation could recording PAV incidents allowed this scenario to run to completion. If needed, this limitation could be worked around with some relatively straightforward code changes.

Scenarios spanning geographical areas larger than a hemisphere would be limited by the earth projection used by Simajin. Again, if needed, this limitation could be worked around with some relatively straightforward code changes.

6.2 Reusable Components
The databases built for the New York City scenario have a large number of reusable components. Nearly all of the pattern database would directly apply to a simulation of PAVs operating in other geographic areas or a more developed simulation of the same area. The episode database would change significantly for a new area, but the automation and scripting tools used to generate the majority of the NYC episode database would be applicable to similar data from another locale inside or outside of the United States. An abroad scenario would require reworking of these tools to the extent that the format of the available data differs from what we used. Most of the exercise database would be reusable, and at a minimum would provide the foundation for extracting different data from the same scenario. The Simanj database tags are simple to remove by generating the Simajin database, but they provide the instrumentation for running study matrices and also a more polished Graphical User Interface to allow non-analysts easier access to the simulation parameters.

While most database components can be reused or regenerated from new data, an example of time consuming database work is the theoretical PAV ports that were placed in downtown New York. There was no data existing for where these were, so we manually picked locations that seemed realistically possible: barges/piers along the Hudson river, Long Island Sound etc. If a different scenario requires a lot of manual input of data because the data doesn’t exist or is otherwise unavailable, this will slow the database development. Other examples of tedious database work would include development of geographically specific ATC tactics and procedures, and tuning of databases with results from parametric analysis.

The United States Geological Survey (USGS) provides terrain data in DEM format for the United States. Simajin uses terrain in DTED format, so a Simajin conversion utility program was used to convert DEM format data into DTED. Geographic Information Systems (GIS) software can usually read and convert most terrain data types (e.g. ArcView), but they are an expensive solution to simply converting file formats. If terrain data in a format other than DEM, DTED, or ASCII grid needs to be used, it will have to be converted with available software or with a custom conversion program.

The commercial flights were taken from the Global Commercial Flight database provided by NASA LARC, and pruned to encompass only flights going to or coming from airports in the New York City area of interest. It would be simple to adapt these scripts to prune this global database for a different set of airports. Data in a different format will require modifications to the scripts.

General Aviation (GA) airport locations and associated runway specifications were obtained from Federal Aviation Administration (FAA) National Aeronautical Charting Office (NACO) publications. Again, the tools used to extract and adapt this data for our New York City scenario would also apply directly to a different region of the US, and possibly elsewhere.

Some custom post processing tools used for extracting paths, computing path complexity, reading and parsing XML, XSLT, Excel, etc., are based on the Simajin output format and would also apply to similar data generated for other scenarios.