Unmanned Aerial Vehicle Path Planning in JMP

Landon Wright and John Salmon

Brigham Young University, Provo, Utah

Introduction

As electronics become lighter, cheaper, and more accessible, Unmanned Aerial Vehicle (UAV) technology has become ever more prevalent in modern society. With this increase in prevalence, additional suggestions of using drones for more mundane tasks in society such as delivering packages, returning library books, supervising children as they walk home from school, traffic monitoring, and other airborne tasks, is becoming a real possibility. While the benefits of these proposed activities are clear, the risks and challenges associated with them are less well understood. One of the key challenges is understanding how a UAV will traverse a crowded city environment filled with skyscrapers and other obstacles. In this paper, we present a simulation environment in JMP that randomly generates a theoretical city grid populated with buildings of varying heights, calculates a possible flight path through the simulated city, and simulates a UAV following the calculated flight path through the city. This simulation provides an accurate physics-based model of the UAV and allows the user to tune several of the parameters that influence its flight characteristics. We then present some results from an exploration of the design space and Monte Carlo simulations demonstrating the utility of the application.

Background and Motivation

Current regulatory constraints prevent the full exploitation of the benefits that UAVs could provide in society. Understandably there is considerable concern as to how UAVs will be able to safely traverse a city environment. Due to the highly dimensional nature of this problem it is difficult predict with a high degree of certainty the exact path that a UAV might follow. Simulation can significantly aid in solving these problems.

There are many factors that influence not only the path that a UAV system will plan to traverse a city, but also the flight characteristics of the UAV, and the likelihood in causing damage to another person or property. Using this analysis and visualization tool we are able to perform Monte Carlo simulations to determine the probabilistic nature of how and where the UAV is most likely to fly within the simulated city. This information can then be used to shape how path planning is performed, to define control laws and algorithms, and even assist in UAV design.

Method

As shown in Figure 1, the flight simulator consists of five modules: a path planner, path manager, path follower, autopilot, and aircraft simulator. These modules work together to simulate the flight path of an autonomous UAV and are outlined in the reference book "Small Unmanned Aircraft: Theory and Practice [2]". The path planner requires information about the size of the map or city, obstacles or buildings to be avoided, and the final destination of the UAV. Using this information the path planner uses the rapidly-exploring random tree(RRT) algorithm to randomly suggest possible branches for a flight path until a viable solution path is found [8].

Once the path planner has found a viable solution path it creates a series of waypoints that are passed into the path manager. In this module, the current state of the aircraft is compared with the waypoint path that was generated in the path manager. Using this comparison a path definition is computed. The path

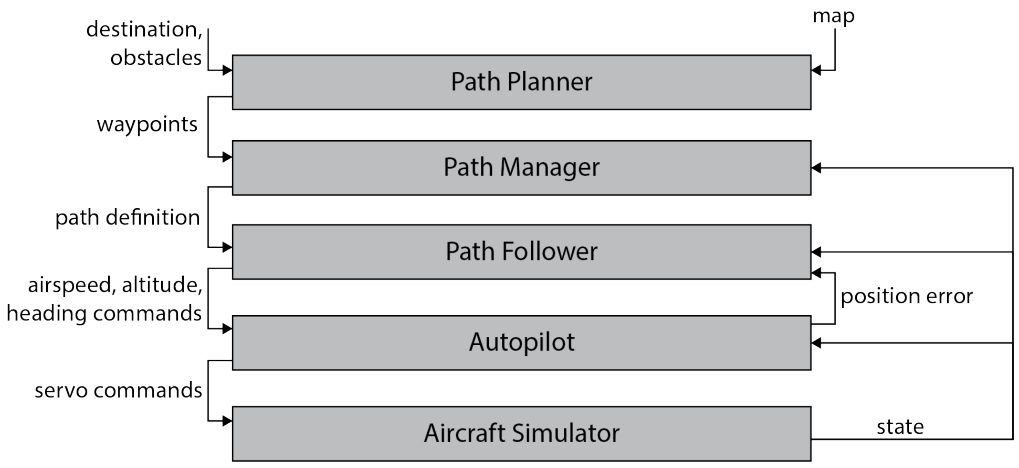


Figure 1: Overview of the interaction of the various parts of the system

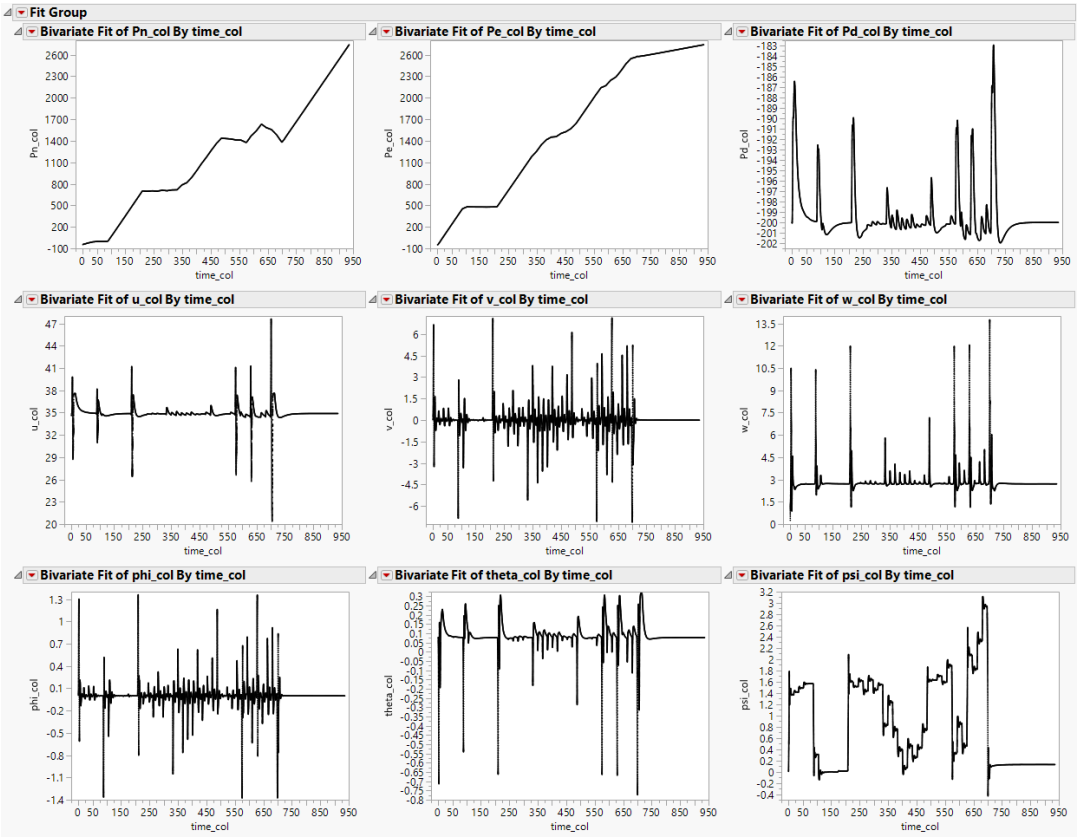


Figure 2: Time series graphs of 9 of the 12 states as calculated at each time step of the simulation

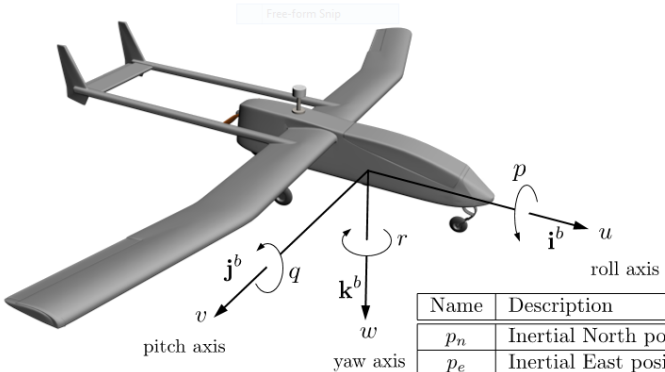
definition contains the current line or arc trajectory that the aircraft needs to follow in order to maintain the course calculated in the path planner. This path definition is then passed into the path follower.

In the path follower the critical airspeed, altitude, and heading commands are calculated. This is done by creating a vector field around the path definition; far away from the defined path the vector field points nearly perpendicular to the path. However, near the path the vector field becomes nearly co-linear with the path as presented in [2]. The vector in the field that coincides with the location of the aircraft is then used to define the commanded heading of the aircraft. Using this technique we are able to ensure that the aircraft rapidly approaches and follows the defined line without excessive overshoot and oscillation. The

commanded height of the aircraft is determined by the path manager and is included in the path definition. This value is simply passed through the path follower into the autopilot. The airspeed is defined to be a constant, but could also be calculated from the path definition. Once the commanded airspeed, altitude, and heading are calculated and collected in the path follower, they are passed into the autopilot where they are used to calculate the servo commands.

The autopilot uses basic control theory to calculate the servo inputs that should be used to control the aircraft. The course angle is controlled using the ailerons of the aircraft whose commanded values are calculated using a PID control algorithm [1] and successive loop closure on the course and roll transfer functions [3]. This calculates the commanded aileron input by examining the difference between commanded heading and the current heading as well as examining the change in heading through time. The height and velocity of the aircraft are used to calculate the servo commands for the elevator and throttle of the aircraft. This is done through the use of a total energy control system [7] which examines the total energy (kinetic and potential) that the aircraft contains at any moment and compares that value with the theoretical energy that the aircraft would have at the desired airspeed and heading. This information is then used to calculate the elevator and throttle commands that should be applied to achieve the desired altitude and airspeed.

The aircraft simulator then uses the equations derived in [2] to determine the forces and moments that would be acting on the aircraft given the servo commands that were calculated in the autopilot. These forces and moments are then used in the six-degree-of-freedom, 12-state model for the flight kinematics as can be found in many textbooks to calculate the state of the UAV at the next time step [4, 9, 10, 11, 12]. This propagation is done through the classical Runge-Kutta method using a step size of 1/10th of the time step of the simulation [5, 6]. This state is then fed back into the path manager, path follower, and autopilot in order to facilitate the next step of the calculation. Figure 2 presents nine of the 12 states over the simulation time, including the position in a North-East-Down coordinate frame, the roll-pitch-yaw angles, and the speed of the aircraft within the vehicle frame (as summarized in Figure 3).



Name	Description
p_n	Inertial North position of MAV expressed along $\mathbf{i}^i$ in $\mathcal{F}^i$.
p_e	Inertial East position of MAV expressed along $\mathbf{j}^i$ in $\mathcal{F}^i$.
p_d	Inertial Down position of MAV expressed along $\mathbf{k}^i$ in $\mathcal{F}^i$.
u	Ground velocity expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
v	Ground velocity expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
w	Ground velocity expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.
ϕ	Roll angle defined with respect to $\mathcal{F}^{v2}$.
θ	Pitch angle defined with respect to $\mathcal{F}^{v1}$.
ψ	Heading (yaw) angle defined with respect to $\mathcal{F}^v$.
p	Body angular (roll) rate expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
q	Body angular (pitch) rate expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
r	Body angular (yaw) rate expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.

Figure 3: Aircraft state variables (reproduced from [2])

Dashboard Overview

The application interface is broken into three major sections. As seen in Figure 4, the left hand side contains the various parameters which the user can adjust. At the top left are the variables associated with the scenario or city definition. Within this block the average height of the buildings, the range of heights, the roof heights, number of streets, and the street gap between the buildings are designated.

Within the middle section the 3D representation of the city, the flight path, and aircraft maneuvers are displayed or animated. Various controls are provided to the user to set the view angles, the size of the scene, and zoom level.

Various output graphs are found on the far right including the birds-eye view of the calculated flight path, the RRT exploration paths, and an additional z-height viewing control. The bottom right contains a variety of buttons executing various functions and scripts within the work flow.

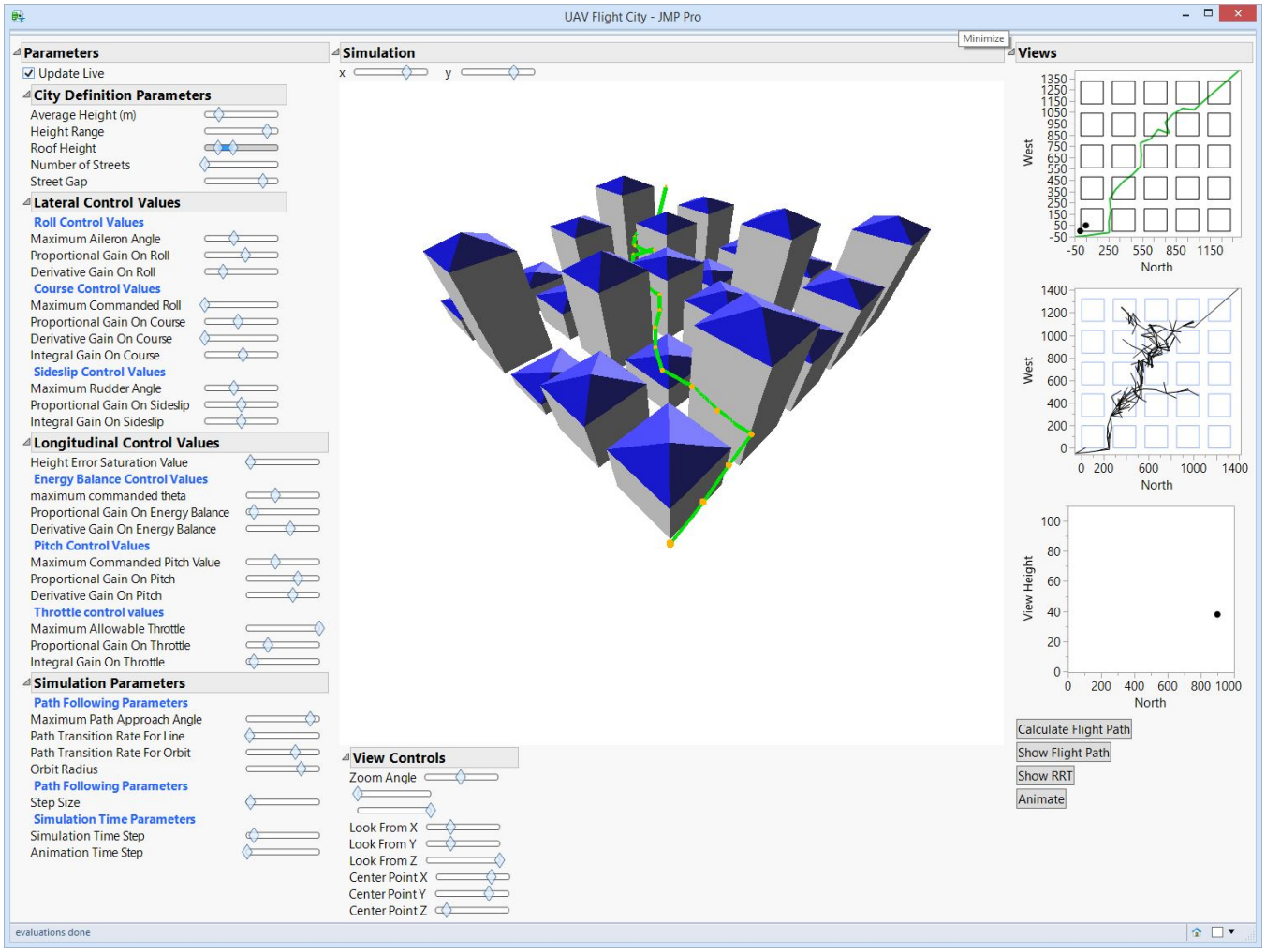


Figure 4: UAV Flight Path Analysis Tool

City Definitions

The examples of the types of cities available within the dashboard are presented in Figure 5. In the top left, a city with five blocks in both the North and West direction is shown. The distance between the buildings, called the Street Gap, is widened in the subfigure in the top right (with respect to the figure in the top left). The two figures on the bottom show a significantly larger city with a lower variance in building height on the left and higher variance in building height on the right. The various combinations of cities provide a diverse set of scenarios upon which to test the path planning algorithm previously described and evaluate the UAV flight characteristics.

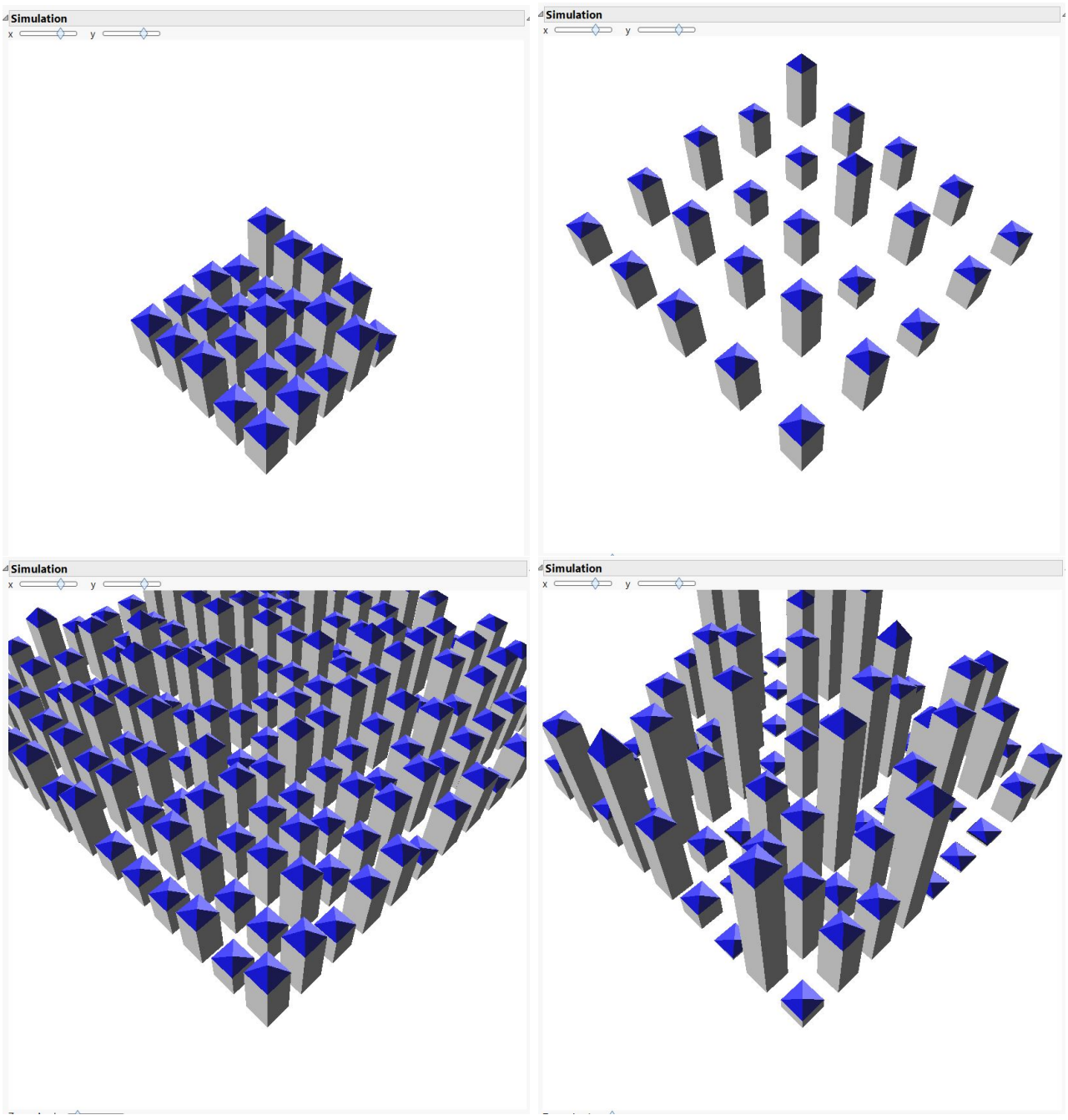


Figure 5: Examples of Parameter Effects within the City Definition Design Space: (top left) narrow streets, (top right) wide streets, (bottom left) larger city, (bottom right) larger building height variance.

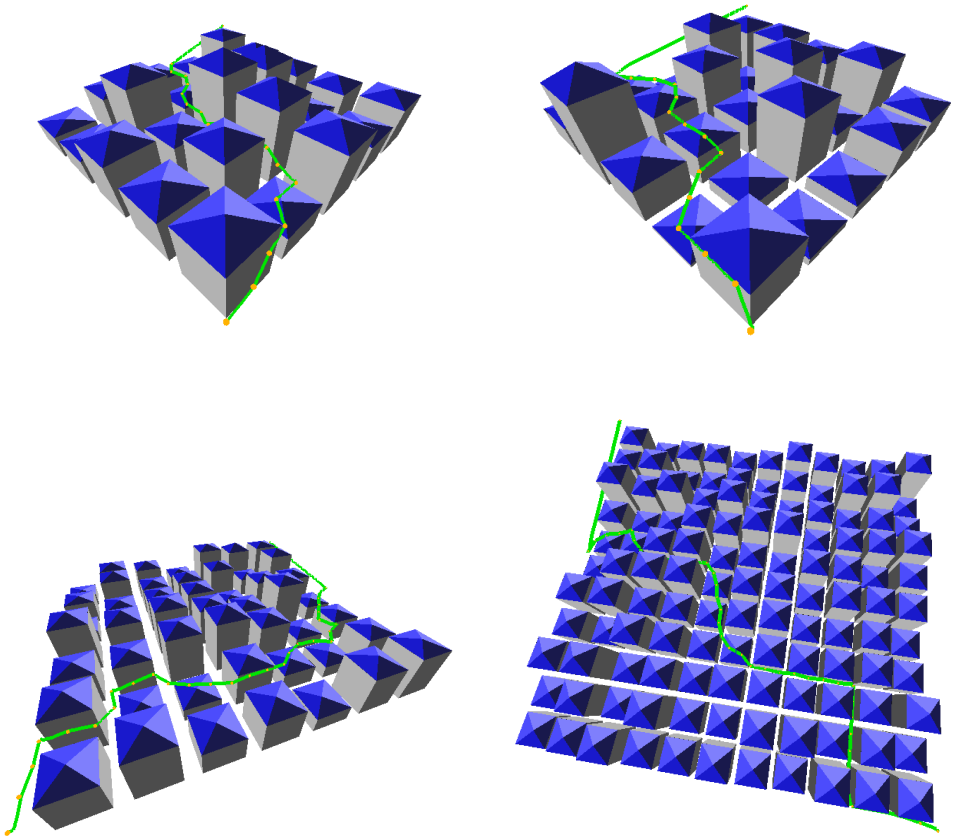


Figure 6: Examples of the calculated flight path as simulated through various cities of differing size and complexity

Results and Discussion

The results of the aircraft simulation are shown in Figure 6. The top left shows a relatively small city with a simple flight path that was quickly found and simulated. The top right image shows a slightly more complex city with a correspondingly complex flight path. On the bottom row we see two cities that are significantly more complex than the ones shown in the top row of the figure. Of particular interest is the path that was followed in the bottom left image. It can be seen in this image that the aircraft initially travelled north and west. However, when confronted with building obstacles that were too tall to be flown over the path planning algorithm was forced to backtrack by travelling east to circumvent these taller buildings. Lastly in the bottom right image we see a very large and complex city. This subfigure demonstrates that the simulation is capable of scaling to very large and complex problems with minimal restriction.

In Figure 7 the RRT exploration paths corresponding to the flight paths in Figure 6 are shown. In the top left the RRT exploration path for the simple city is correspondingly simple and converged quickly to the end point with relatively few branches. The exploration paths in the top right and bottom right are interesting in that they both contain two significant branches to the exploration trees. The upper left image shows a split in the tree at approximately 400m North and 400m West. From this split a branch that continues North and a separate branch exploring the West is observed. Ultimately the branch to the west is the first to find a valid path to the end point and as such it is the branch that is followed in the flight path shown in Figure 6. A similar situation is observed in the bottom left graph where a split occurs and ultimately one branch is the first to find a valid path to the end point and, as a result, that branch

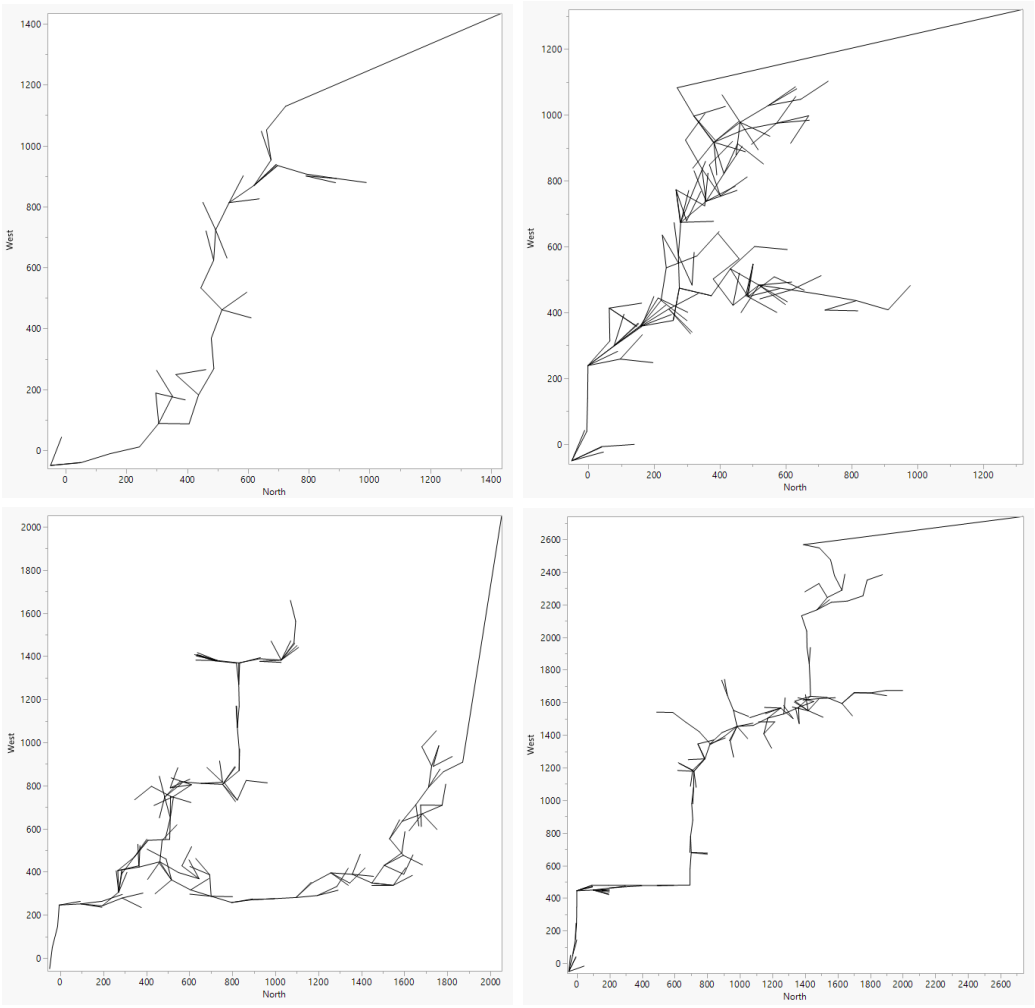


Figure 7: Examples of the computed RRT exploration paths for the flight paths shown in 6, some paths converge quickly to the destination node while others take many iterations to converge.

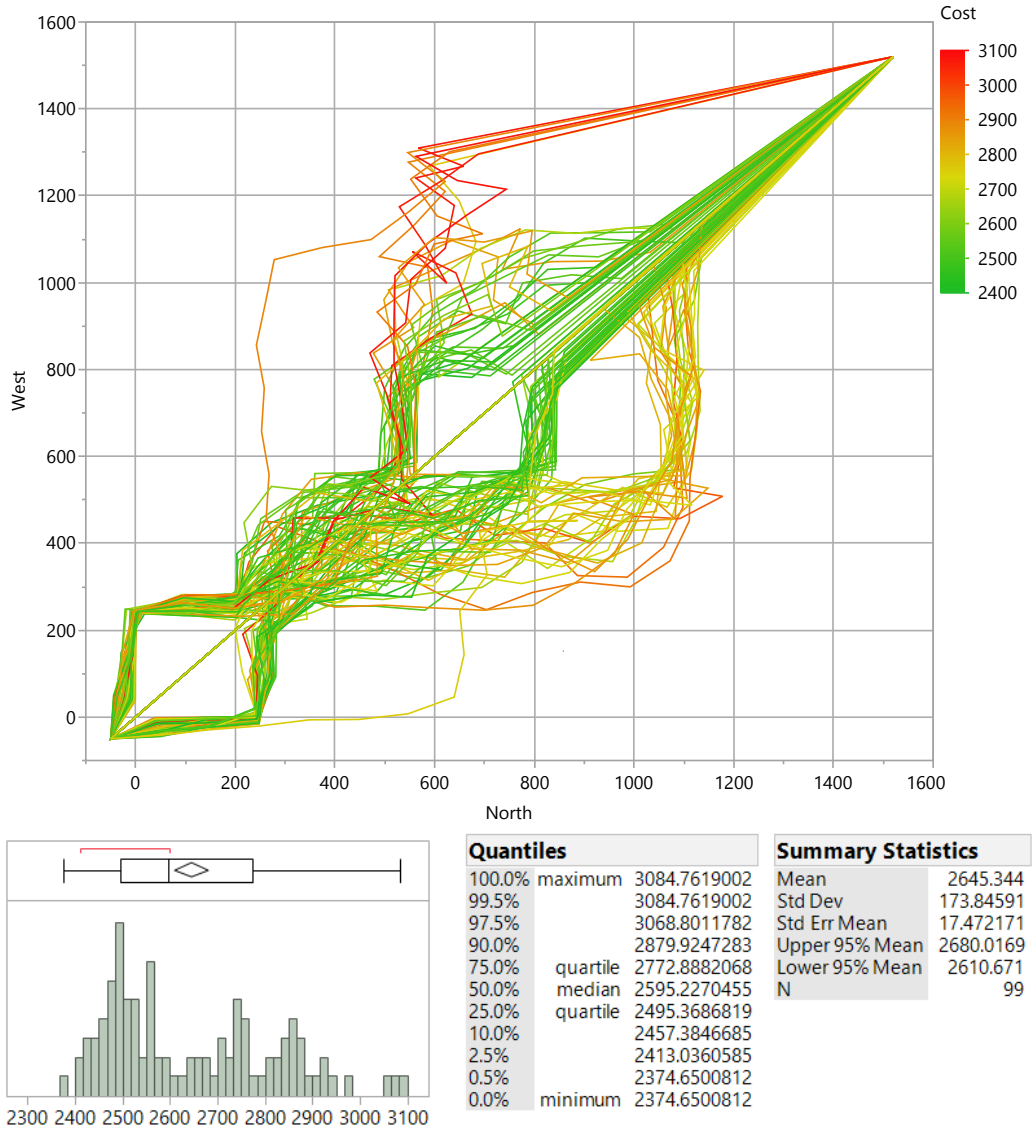
becomes one followed in the flight path. The last image on the bottom right corresponds to the largest and most complex city. The RRT exploration paths show some branching, but all of the branches are relatively short and do not seem to extend far beyond the main branch. This can be attributed to the fact that the city corresponding to this graph had a high variability in building height, meaning that there were many building that were short and the aircraft was able to simply fly over these building. As a result, the process of finding a path to fly through this city was relatively simple and did not result in a highly complex RRT exploration path.

Monte Carlo Exploration

Once the model and RRT algorithm were sufficiently tested, 100 Monte Carlo simulations were executed to extract trends and identify the cost associated with each path. The cost is calculated as the total distance travelled from the origination at one corner of the city to the destination at the other. Therefore, in Figure 8, the flight paths that follow a near direct path along the diagonal across the city are lower in cost (and colored green) compared to the paths that deviate away from this diagonal (colored in shades of yellow and red). These figures are also used to validate the simulations as evidence for avoided square regions representing buildings above the 50m flight altitude are observed.

A histogram of the same cost data is presented in the bottom part of Figure 8 (with one run removed

due to missing data). Of note is the multiple modal shape of the histogram representing the low, medium, and high cost groups, with green, yellow, and red color respectively from the top of Figure 8. Among the MC simulations, the lowest has a cost value of 2374 and the highest cost with a value of 3084, for a full range of 710, and almost 30% more costly than the lowest value suggesting that significant advantages can be realized from running additional RRT attempts.



Quantiles		
100.0%	maximum	3084.7619002
99.5%		3084.7619002
97.5%		3068.8011782
90.0%		2879.9247283
75.0%	quartile	2772.8882068
50.0%	median	2595.2270455
25.0%	quartile	2495.3686819
10.0%		2457.3846685
2.5%		2413.0360585
0.5%		2374.6500812
0.0%	minimum	2374.6500812

Summary Statistics	
Mean	2645.344
Std Dev	173.84591
Std Err Mean	17.472171
Upper 95% Mean	2680.0169
Lower 95% Mean	2610.671
N	99

Figure 8: 100 Monte Carlo Simulation of RRT algorithm with associated cost histogram

Conclusion

JMP offers considerable features to analyze and explore UAV flight path planning algorithms through its visualization tools and statistical processes. Future efforts will develop and contrast additional path planning algorithms to support UAV activities within cities and other locations with high-rise buildings, narrow streets, and complicated environments.

References

[1] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Isa Research Triangle Park, NC, 1995.

[2] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[3] Martin C Berg, Naftali Amit, and J David Powell. Multirate digital control system design. *IEEE Transactions on Automatic Control*, 33(12):1139–1150, 1988.

[4] L Stevens Brian and L Lewis Frank. Aircraft control and simulation. *John Wiley & Sons, Inc., Hoboken, New Jersey*, 2003.

[5] John Charles Butcher. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[6] Steven C Chapra and Raymond Canale. Numerical methods for engineers. 2005.

[7] Anthony A Lambregts. Vertical flight path and speed control autopilot design using total energy principles. *AIAA paper*, 83, 1983.

[8] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[9] Robert C Nelson. *Flight stability and automatic control*, volume 2. WCB/McGraw Hill New York, 1998.

[10] Jan Roskam. *Airplane flight dynamics and automatic flight controls*. DARcorporation, 1998.

[11] Robert F Stengel. *Flight dynamics*. Princeton University Press, 2015.

[12] Thomas R Yechout. *Introduction to aircraft flight mechanics*. Aiaa, 2003.