MULTI-DIMENSIONAL, MULTI-AGENT PATHFINDING FOR AUTONOMOUS FLIGHT PLANNING WITH AIRSPACE DECONFLICTION

David Russell,[*] Michael Liquori,[†] Chien-Tsung Lu, Ph.D,[‡] Denfeng Sun, Ph.D,[§] and Fred Meyer[**]

Safe, robust and stable flightpaths for Unmanned Aerial Vehicles depend on a combination of pre-flight planning, navigation, object detection, collision avoidance, traffic management tracking, communication and registration. We describe an innovative multi-agent auto-generated flightpath algorithm that can solve for simultaneous, safe, efficient, deconflicted flight paths for hundreds of vehicles operating in a complex, high-density airspace. Cost-function factors may include risk calculus, vehicle profiles, and environmental variables. These can be used to drive the flightpath selection to the most effective and efficient path given various mission planning scenarios. The addition of a virtual "Avoidance Limit" sphere surrounding the vehicle enables Detect-And-Avoid maneuvering without disrupting pre-defined flight plans but requires extending the path search algorithm into as many as six dimensions. Flight Plans are created such that the exact location, trajectory and velocity of every vehicle in the system is known at any given point in time enabling correct-by-construction deconfliction of paths.

INTRODUCTION

Safe and effective operation of unmanned aerial vehicles (UAVs) in dense, low-altitude airspaces depends upon a resilient flightpath search and navigation capability. While today's UAVs have highly sophisticated systems for general navigation and object Detection And Avoidance (DAA), most flight paths today are generated manually and are topologically simple. These paths typically follow either a series of waypoints, simple up/over/down or repetitive line patterns. Developing systems for effective, efficient and safe deconflicted flightpaths using this approach can be challenging because pilots often lack information about the trajectories of other UAVs. The risk of error in plotting paths beyond visible line of site that avoid existing landscape features (e.g., buildings, trees, poles, antennae, etc.) is high. For example, in an uncontrolled airspace without deconflicted flightpaths, the probability of at least one vehicle conflict in an airspace approaches 100% with as few as 5 vehicles / km^2. [1]

Over 80% of the US population currently lives in urban environments. By 2050 that is projected to rise to 89% in the US and 68% worldwide.[2] This high population density in urban environments

[*] CTO, Airio.io and Principal Engineer, R&D Birket Engineering, Winter Garden, FL.
[†] CEO, Airio.io Oakland, CA.
[‡] Ph.D. Professor, Department of Aviation Technology, Purdue University.
[§] Ph.D. Associate Professor, Department of Aviation Technology, Purdue University.
[**] Director, Data Analysis, Phoenix, AZ.

implies that UAV logistics and UAM Human Transport functions will primarily operate in crowded and complex airspaces.

In this paper, we outline and test a concept for generating unique, deconflicted flightpaths for multiple aerial vehicles operating in a dense low-altitude airspace. We hypothesize that a system that automatically generates flightpaths could substantially improve the speed, efficiency, and safety of UAV operations. Such flightpaths reduce risks and improve effective airspace management when combined with Detect and Avoid systems and traffic management systems. The algorithm also supports the ability to optimize flightpaths to a wide variety of operating objectives and constraints by leveraging the power of comprehensive geospatial and operational datasets.

Automated pathfinding algorithms have been around since the Dijkstra in the1950s.[3] Path routers have been used to solve a variety of problems including optimizing vehicle delivery routes, organizing traces on printed circuit boards and integrated circuits, and a variety of geospatial analyses, among others. Pathfinding algorithms typically require source and destination inputs and a raster of rectilinear nodes and edges representing the operational space. Grid search methods use a step wise evaluation from cell-to-cell to isolate and track viable alternatives until a lowest cost solution is identified. Grid search methods include Dijkstra's method, and Lee's algorithm.[4] Line/probe algorithms (e.g., Hightower's algorithm) use line segments instead of grid points to discover paths.[5] Other approaches include Steiner trees, Global routers, and the A* and D* algorithms.[6]

Murray and Chu described a generalized routing strategy for UAVs operating in a delivery context, but their approach did not generate specific flightpaths.[7] Barmpounakis et al. defined some of the core challenges with navigation based primarily on Detect and Avoid strategies, including improvements to vehicle design, localization, navigation, and vehicle-to-vehicle coordination.[8] Vásárhelyi et. al. examined flocking behavior, while Xue and Do showed that complexity is poorly correlated to UAV density and is more strongly correlated to the types of conflicts and associated avoidance maneuvers.[2, 9] Chakrabarty proposed a scheme for multiple vehicles sharing a common reserved airspace in discretely defined volumes by relying on vehicle-to-vehicle communications.[10]

APPROACH

While the mathematics for 2D pathfinding algorithms are well-developed, adopting these to UAV airspaces requires additional dimensions. In fact, precise solutions that fully optimize the common three spatial dimensions, a time dimension, velocity constraint profile, and a safety parameter might set up a 6-Dimensional optimization problem. Optimization under such complexity is considerably more challenging. We simplify the problem by a) combining multiple search algorithm methods into separate agents, b) using a dynamic database architecture, c) tuning heuristics to set the topology and optimizing cost functions, and d) defining the trajectory as continuous mathematical paths which include space and time rather than waypoints. These alterations enable effective flightpaths where the exact location of every vehicle is known at any given point in time from authorization thru to flight completion, enabling discrete deconfliction and multi-agent optimization strategies.

Geospatial and Operational Inputs

We define the airspace using available geospatial data in a projected geographic coordinate system. A base topographical raster is compiled from existing Digital Elevation Models and/or available LIDAR, depending on the resolution requirements of the operational airspace as in Figure 1.[11]

Additional information about objects contained in the operational landscape can be added to the base raster, including buildings, antenna masts, vegetation, roads, rail lines, overpasses, etc.

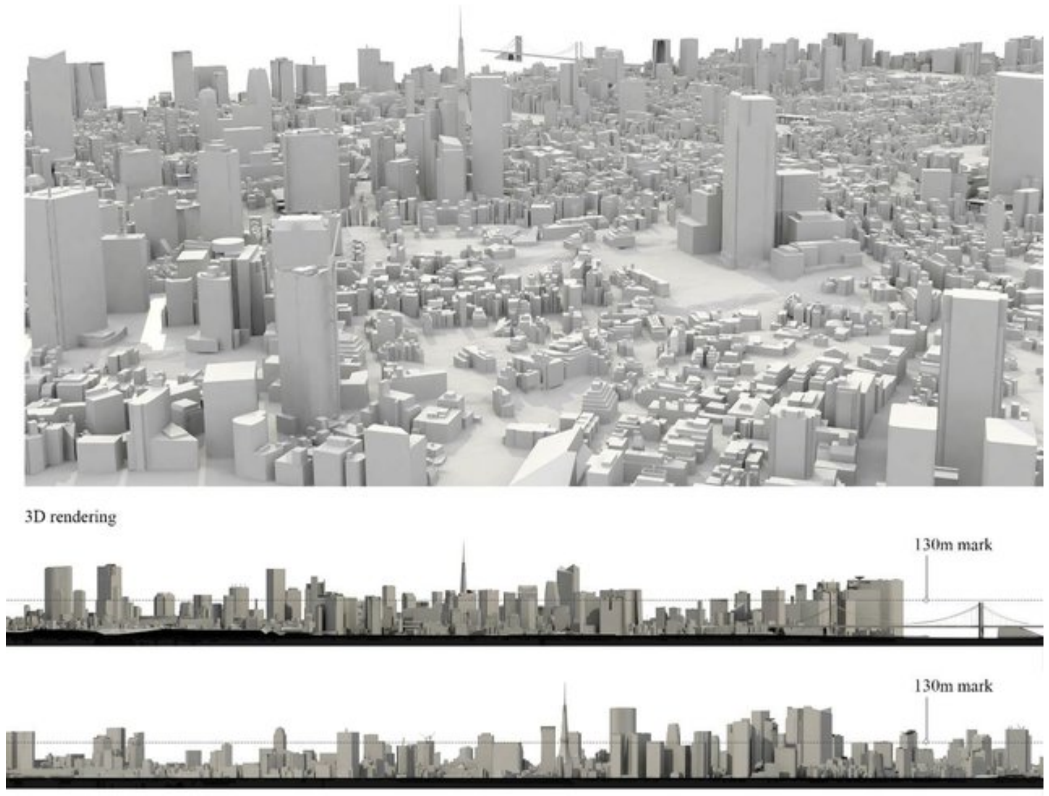


Figure 1. Modeling Urban Airspace.

A cost matrix is developed that encapsulates both the value of the metric being considered and the relative weighting of each metric. The metric may be identified as value, an index, or multidimensional indices into a raster or other associated media while the weighting may be a single value or an index to separate tables and/or combinations of tables of values given for different scenarios, times, or vehicles. The same vehicle and mission plan may generate different results at different times or weather conditions for example. The cost function used for each route is derived from that particular mission plan. The problem is the vehicle really needs this information to navigate an efficient path, but it has neither the storage nor the processing power to make use of this model so the vehicles find their way without it. The alternative is to pre-plan the flightpath when all of this data and the processing power to use it is available and load just what is needed into the vehicle.

The third alternative is cloud-based processing to assist the vehicle during flight, but this drives up the communications requirements for the airspace, which is already problematic.[19]

One of the strengths of this algorithm is its adaptable cost-basis for various use cases. The cost matrix algorithms can be adopted with relevant data and only modest refinements to the cost matrix algorithm. While different use cases may use some of the same metrics, the associated weighting may still differ significantly. In other cases, significantly different metrics may be required. For example, a cost matrix for an urban delivery use case might define costs by minimizing flight time or distance or maximizing range. Alternatively, a search & rescue application might use a cost model based on likely deviations from existing roads and paths, requiring inputs for existing roads, paths, search scenarios, behavioral assumptions, sheltering buildings, water sources, etc.[12] Alternatively an environmental monitoring application using multiple vehicles with different sensor types might determine flightpaths based on vegetation, soils and hydrography datasets. Other data such as regulatory constraints, geofences and other operational parameters can be integrated into the system code or through API calls to existing GIS data resources.

Operational inputs to drive each route require the source and destination points and a time window for the flight plan. A variety of other user-input factors that improve the operational value of the system might include vehicle specifications, payload information, operational commands to trigger onboard sensors, camera angles, communication parameters, etc.

Hybrid Search Techniques

Our approach uses a hybrid of grid/probe techniques. In general, grid path search methods will find a viable solution if it exists but can be very slow due to the high computational demands. By contrast, probe search methods are very fast but may not find an initial solution. Using both helps to improves speed, and enables the path to be influenced by a wide variety of raster-based data, including environmental and operational constraints, that are integrated into the solution using cost functions and heuristics to define a high-quality (if not perfectly optimal) path topology.

Generally, path algorithms which evaluate all potential solutions are classified as NP-Complete. By definition, a search algorithm returns a path which best fits the cost function given as opposed to the globally optimal solution. In a hybrid search, the solution space is fully considered in small complex areas and less consideration is given to areas of lower complexity, seeking a balance of least cost paths versus speed of computation.

Traditional 2D grid-based pathfinding algorithms can be slow and computationally intensive due to the $O(N^2)$ number of iterations required. The addition of a third dimension and the need for high-resolution grids when operating in dense spaces increases the computational demand by several orders of magnitude if not exponentially. Precise solutions become even more complex with the addition of time (4D), velocity, avoidance limits, and other cost function complexities. While velocity is usually simply considered as the derivative of the location dimensions, it may also be given as an input to the system as a velocity target or profile to inform or constrain the cost function and final route characteristics.

Dynamic Database Architecture

In most grid path search algorithms, the grids contain practically all the information about a particular spatial area (actually a matrix in this 3D case) including whether it's an obstruction, cost function values and variables, and the current cost and retrace pointers. Because this data is unique to a given path, no two paths can be searched simultaneously in the same area. As the grid field may represent a large geographic area at high resolution, the traditional grid approach is not really feasible. The static data is therefore arranged in a large structured database keyed by kD-Tree of the 3D space coordinates such that all of the data remains as its native type – weather data, LIDAR data, polygons, etc. at its native resolution. The grid system is built virtually as the grid expansion phase progresses outwards from the source point and contains only the temporary information generated for each particular search path. This virtual grid system allows searches to be based on a multi-agent architecture. Multiple searches are able to traverse the same search areas simultaneously without having to lock out grids with different data searching through the same points. Other multi-agent simulations have been performed such as Ether, developed by Lundell et al. for teams of UAVs.[13] This virtual grid system is also necessary as the size of the grid itself may change between one route and another due to the potential size and capabilities of the vehicle and environmental conditions and therefore of the Avoidance Limit.

Avoiding Geospatial Objects with 3-Dimensional, Least-Cost Path Finding

The search process begins with a probe line drawn from source (S) to destination (D). This is the minimal distance path. The database is queried to provide all geospatial and related data that intersects this line, which may include information on the terrain, elevation, geofences, buildings, weather, feature types, vegetation, risk factors or any other data relevant to the cost function as defined by the mission plan and environmental factors.[14,15,16] Initial analysis of this data (along with other operational inputs) is used to select the initial target cruise altitude for the flight path. If no obstructions are found at the target altitude, this short line path essentially may become a simple up-over-down flightpath if all other requirements of the mission plan are met.

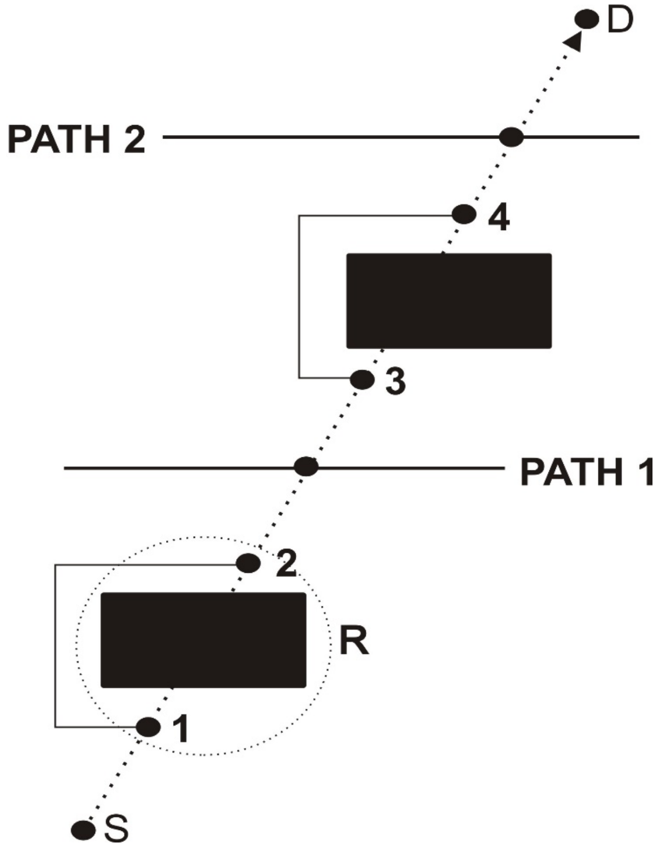


Figure 2. Straight Path Line Broken by Obstacles.

If an obstruction is detected in this path, the path is broken into two segments, and the testing is continued until some number of disjoint line segments are produced from the original shortest path line. In Figure 2, this is shown as breaking the original line path S, D, into discontinuities at locations 1 to 2 and 3 to 4 to avoid obstructions. Path1 and Path2 are existing routes which will intersect this path as well. These could lead to selecting a disjoint altitude or varying timing to deconflict the path.

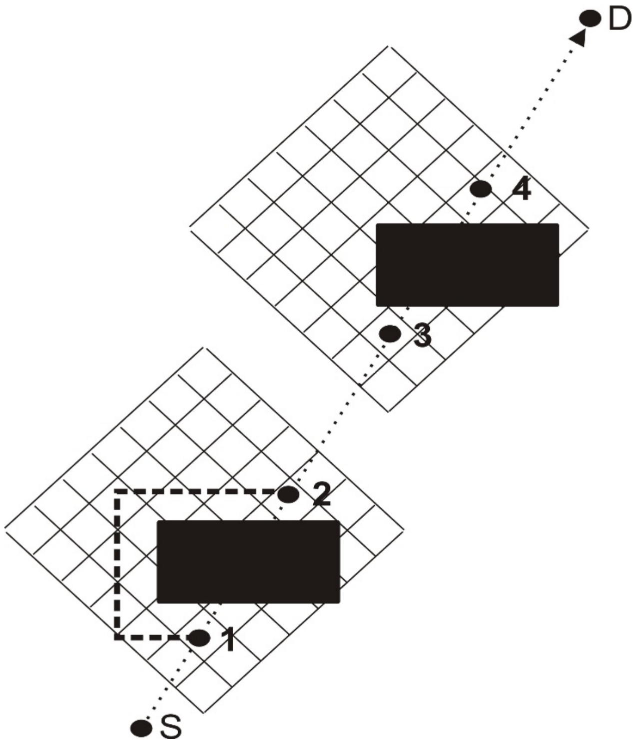


Figure 3. Independent Routing Agents Spawned Around Obstacles.

At this point an agent instantiation of the grid pathfinding algorithm for each disjoint segment is started to find the path around each obstruction using one of multiple possible full search least-cost pathfinding algorithms. The "source" of the route is the new endpoint of the line segment (1) and the "destination" is now any other point on the dotted line (2, 3, 4, D) as shown in Figure 3. It may be topologically costly to reconnect directly on the other side of the obstruction, so the search may find a better (lower cost) path to any other point on the initial probe line. It may also consider any search path in an overlapping agent as a solution. Any intermediate segments are then abandoned. This approach reduces the computational demand by keeping the m x n matrix size of the grid space limited to the size of the obstructions, not the size of the path. As the agents run in parallel, the runtime essentially remains constant for any number of obstructions.

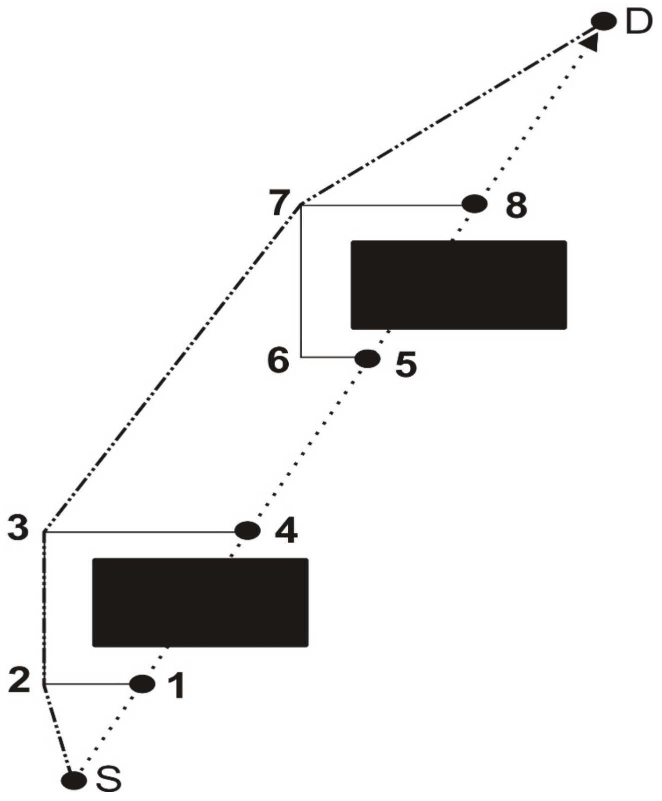


Figure 4. Line Straightening of Reconnected Path.

Once the initial probe path is fully reconnected by the agents, final analysis of the line is performed as shown in Figure 4. The line path segments may be consolidated into fewer lines by a line straightening algorithm, or a long line may be broken into smaller ones to accommodate the velocity profile and timing constraints. The permissible angles of segment changes may be specified by the mission profile and corners may be replaced by curved spline segments at a minimum radius for human transport, for example.

Trajectory Definition

In order to mathematically define the entire trajectory, not just the waypoints, it should be noted that any system of multiple dependent dimensions may be made independent by projecting them onto an arbitrary additional dimension. In this case, we want time to be a component of the trajectory, so instead of Y = f(X) we can redefine the trajectory as Y= f(T) and X = f(T). This can be expanded to any number of dimensions and this allows the flight control system to deal with each dimension independently.

For example, the trajectory of X, Y and Z dimensions are shown as a function of Time in in Figures 5, 6, and 7. The blue circles indicate the original inflection points, or points where the trajectory changes directions in the routing model. The colored lines then represent a spline which provides the mathematical expansion of those points into a fully defined continuous path. Different types of splines may be applied to define slightly different paths. These are calculated as Cubic splines which have more rounded contours. Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) splines provide better straight lines as demonstrated in the added dimension for the Avoidance Limit as shown in Figure 8. Similarly, any number of controls for the vehicle or peripherals could be appended as long as they are expressed as a function of Time.

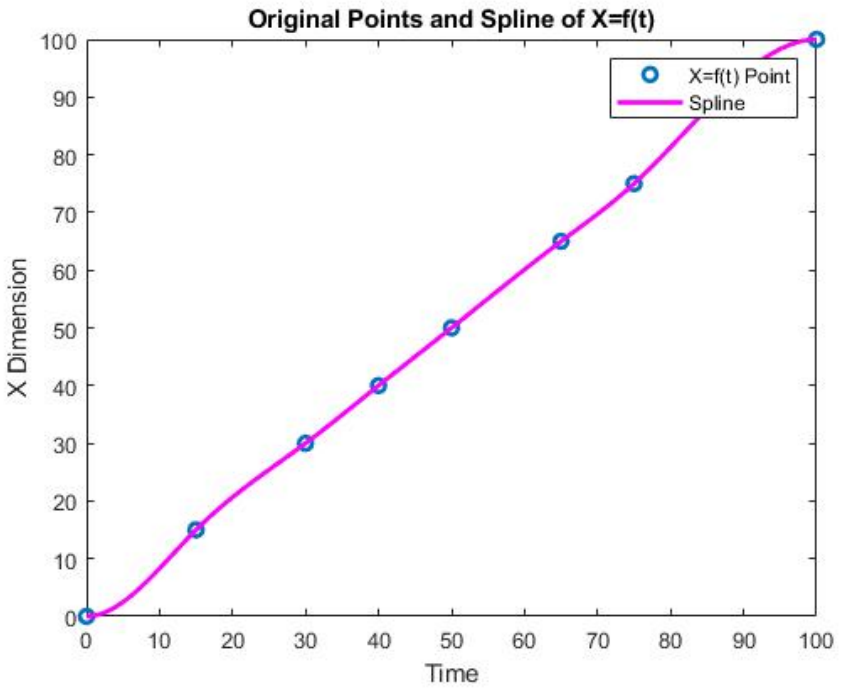


Figure 5. The original path inflection points and cubic spline of X dimension.

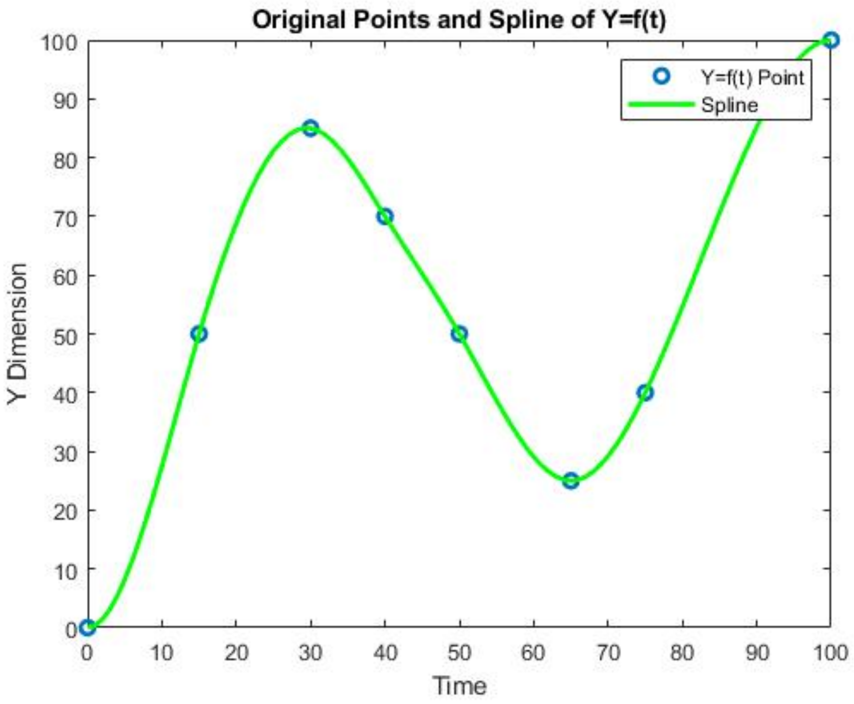


Figure 6. The original path inflection points and cubic spline of Y dimension.

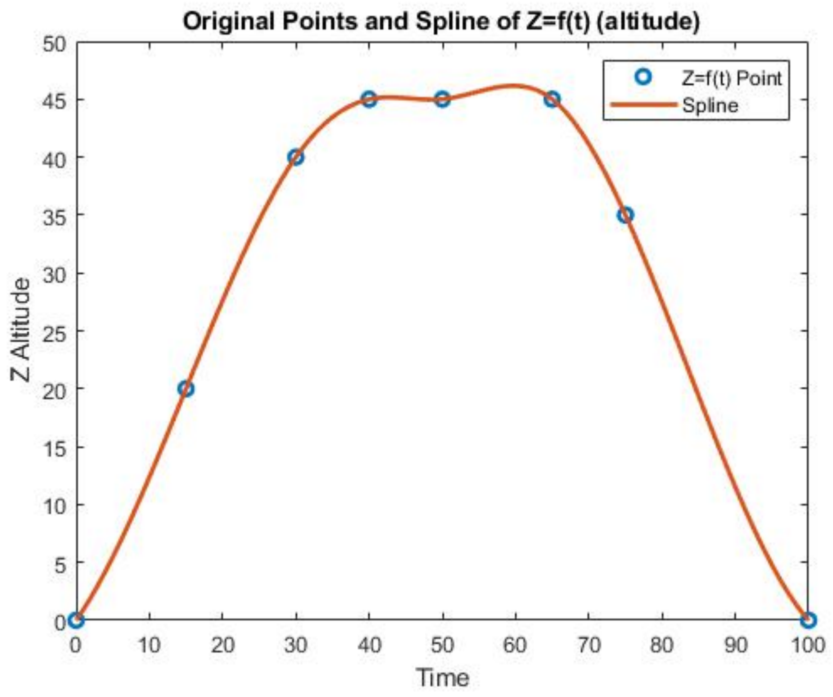


Figure 7. The original path inflection points and cubic spline of Z dimension.

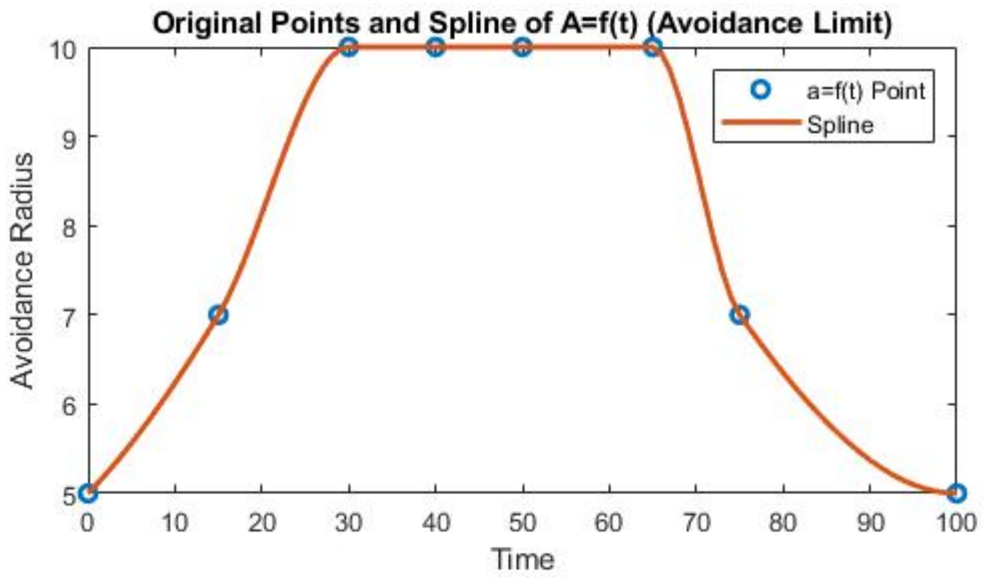


Figure 8. The original path inflection points and PCHIP spline of A dimension.

In the example of Figure 8 where UAM vehicles are set for 10m separation, the Avoidance Limit (AL) begins at a radius of 5m. As the vehicles have a chance to separate, the avoidance limit increases to 10m. When nearing the landing zones, the AL shrinks again to meet the requirements of the landing separation. This could also be coupled with a decrease in velocity if needed as the vehicles would need to maneuver with more precision. These splines allow the system to interpolate the find the exact location for any T plus the radius of the Avoidance Limit. Figure 9 depicts the overall sum of the three coordinates into one plot.

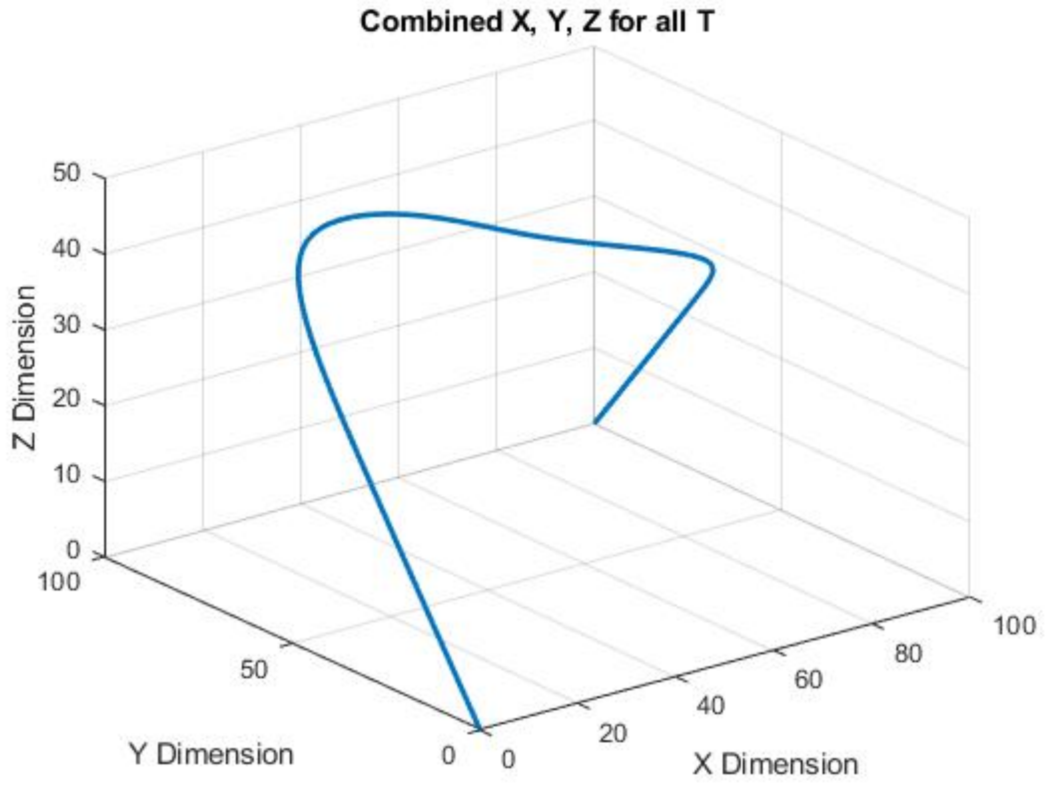


Figure 9. The 3D consolidation of X = f(T), Y = f(T), Z = f(T).

Avoidance Limits

As an additional deconfliction feature, the system generates an "Avoidance Limit" (AL) for each vehicle; a distance-based, spherical buffer around the vehicle such that it triggers specific deconfliction or emergency strategies. The avoidance limit can be thought of as a variable inverse-geofence sphere that exists in the same 4D space and time as the vehicle, but with a radius that is defined by the vehicle type and size, velocity, environmental factors (wind, temperature, etc.), regulatory factors, and operational capacities of the vehicle. For example, a vehicle with poor ability to maintain position would have a larger AL, and piloted vehicles might have very large AL. The distance between any two avoidance spheres is equal to or greater than to the minimum spacing required between the vehicle and all other vehicles and obstructions, as each vehicle could be anywhere within the sphere at any moment in time.

Avoidance Limits extend outwards from the flightpath and can intersect parallel flight paths, just not at the same time the other vehicle is present. This makes most efficient use of the airspace. Similar to the lane on a highway, the UAV can "own" its flightpath while allowing other UAVs to occupy adjacent flightpaths (lanes) as long as the timing of occupancy is beyond their respective Avoidance Limits. This strategy helps to improve deconfliction by giving each operating vehicle a unique flightpath while enabling high traffic “highways” and organic vehicle corridors.

As flightpath deconfliction is calculated, the paths are computed such that no two avoidance limit spheres are closer than minimum values. These spheres serve two purposes: First, they serve as a check that the vehicle has been able to stay on course. If it cannot, then this is the point at which the vehicle is no longer deconflicted, and it may be the point where a backup autopilot is selected. Second, it allows the vehicle to maneuver within this sphere and remain deconflicted. Avoidance Limits enable independent maneuvering using on-board Detect/Avoid algorithms, so the UAV can take emergency deconfliction maneuvers outside of the flightpath without impinging on neighboring vehicle trajectories. In other words, the temporary emergency deviation from the flightpath triggered by DAA can be executed without impacting other adjacent vehicles.

A violation of the AL could trigger local emergency procedures such as descending to layers reserved for unplanned trajectories or landing. Where vehicle-to-vehicle communications are enabled the vehicle may transmit declarations that warn other vehicles that may be outside their normal DAA range while the emergency is cleared or the vehicle must take other emergency measures to clear the controlled airspace.

The resultant solutions provide resilient deconflicted flightpaths for potentially several hundred vehicles/km^2. The precise limits of this method are still being explored. In practice, each authorized flightpath could be reserved for the entire duration of the flight (including any return flight). In theory, with precise launch times and navigational controls, the AL could be reduced to transient bubbles consisting of just the Avoidance Limit sphere. Such a system could maximize use of the airspace.

Flightpath Deconfliction

The search routine simultaneously evaluates other existing flight paths for potential conflicts. It very common for flight paths to cross, but there is a lower probability that the UAVs will cross at the same point at the same time. When the search path (grid or probe) detects that it has crossed or approached another existing flight path, a higher cost is added to the search which has the effect of driving the search algorithm to find a slightly different path, completely avoiding the path conflict. If no lower cost path is found, this path will evaluate potential conflicts based on flight time and position estimates to allow the path because there is no conflict, or it will deconflict the path by altering the Time dimension if there is a confirmed conflict.

If the two paths are colinear, i.e. following the same path, again this is allowed as long as they are not coincident in Time but it does bring up the “following” problem. A faster vehicle overtaking a slower vehicle on the same path needs to decide whether to pass the slower vehicle or match speeds. Both strategies have Pros and Cons, so the details of the Mission Profile may provide the answer. If the colinear path is beginning to form a Self-Regulating Corridor, this might weight the solution towards matching speeds. As the final path will have an accurate prediction either way, the decision tree for this behavior is still being studied.

Deconfliction Strategies

Deconfliction between vehicles occurs in two ways; 1) by maximizing the independence of unique flightpaths (i.e., minimizing the number of flightpath intersections), and 2) by utilizing velocity (and therefore Time) to separate vehicles when approaching intersecting flightpaths (or within the calculated distance of the two Avoidance Limits.)

When using velocity adjustments as the deconfliction method, the timing to avoid a potential conflict can be adjusted either by altering the take-off time or by making alterations to the vehicle's nominal velocity. Because the velocity, trajectory and position of all vehicles are known, the detection limit extends all the way to the time of pre-liftoff authorization, enabling each UAV to maximize its safe path with only minor adjustments to velocity. By contrast, systems that depend on Detect / Avoid strategies must often make dramatic adjustments to both velocity and trajectory because the detection limits are relatively short.

The overall timing of the flightpath may not change, because a slight decrease in velocity to avoid a potential conflict at one point may be compensated for by a subsequent increase in velocity after the conflict point. Multiple conflict points may exist in a single flight path and each may be corrected in a similar manner, but the system attempts to compensate by slowing for one conflict, and then trying to increase velocity for the next. Even if the overall time of the flight plan changes, it is still accurately predicted when the flight plan is published to the regulatory systems and to the user. Most importantly the predicted flight time is very close to the predicted time, lowering costs.

TEST SCENARIOS

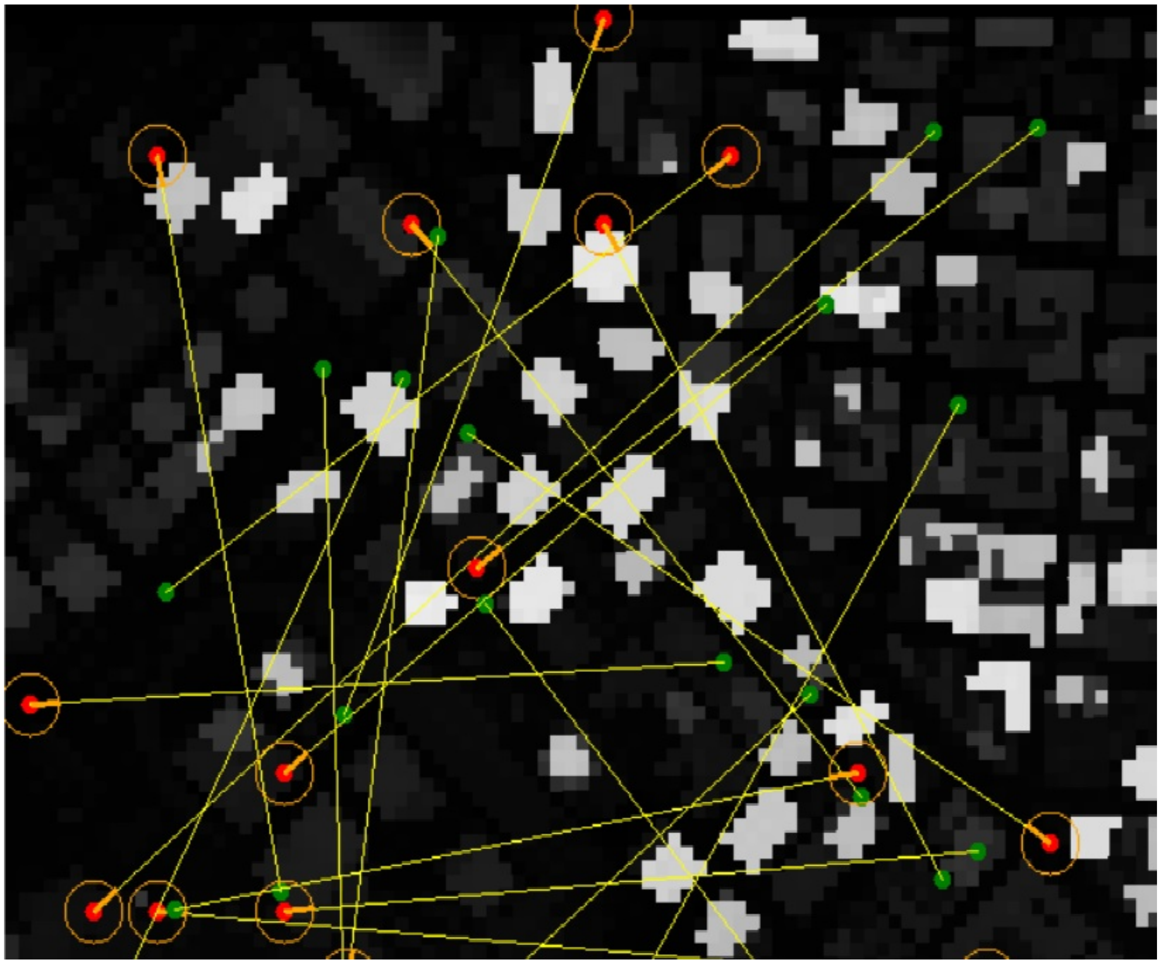


Figure 10. UAV Source and Destinations for DAA Navigation Test.

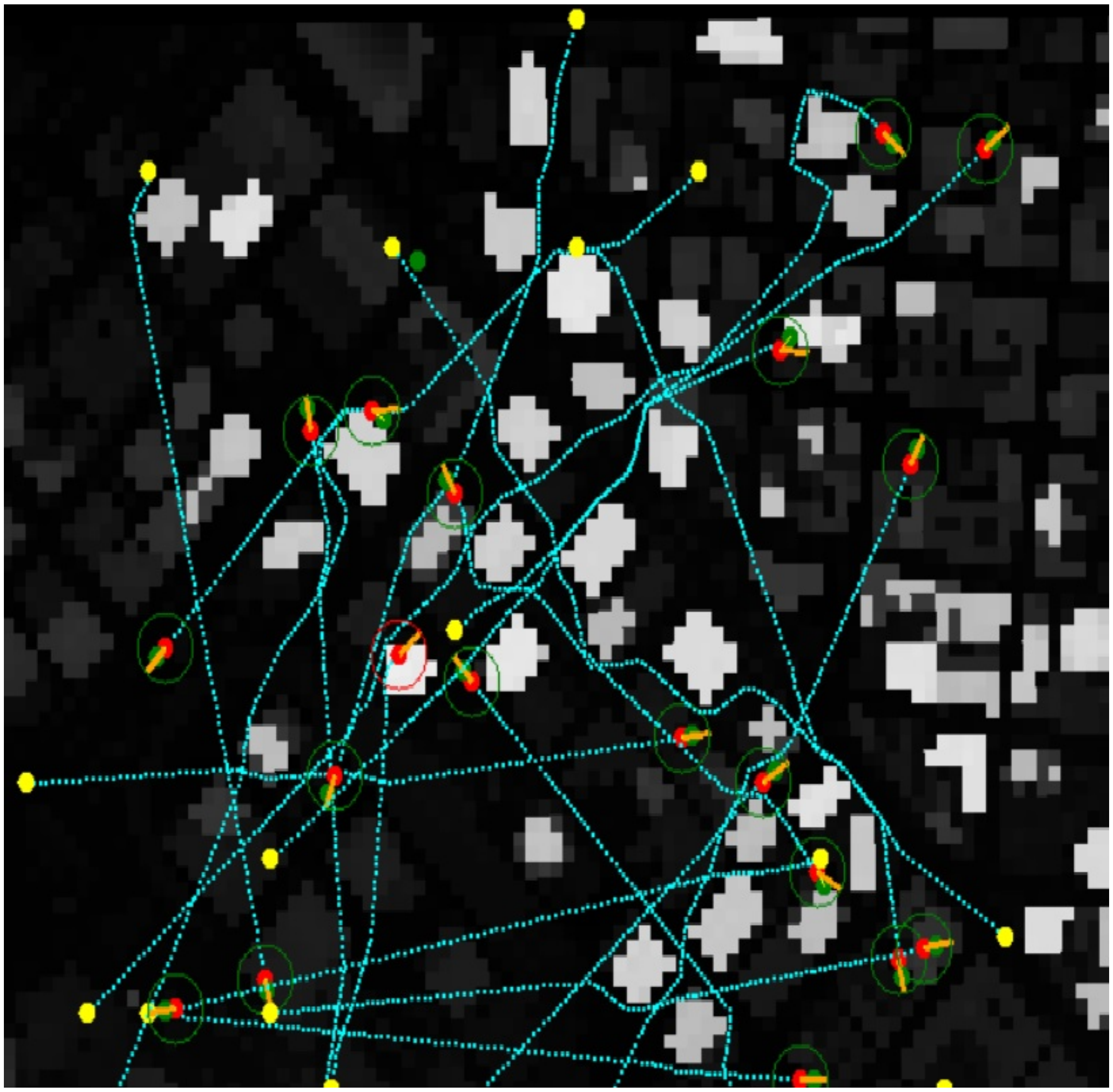


Figure 11. Resultant Paths with DAA Only.

Our initial test was developed as a simulation of the airspace and a variable number of UAVs with a rudimentary but workable flight by DAA only strategy. The objective was to test the hypothesis that utilization of DAA only in flight would lead to less predictable time-of-flight as the number of vehicles in the airspace increased. The initial setup of one simulation run is shown in Figure 10.

This simulation is a 1 km^2 section of San Francisco's Financial District. This is a slice of the 10m DEM model at 100m AGL. In this simulation all the paths are of equal length, but randomly placed across the flight area. The simulation was run averaging the results of 100 runs each for numbers of vehicles ranging from 10 to 100. The resultant paths of an example simulation are shown in Figure 11.

While this simulation was done in 2 dimensions whereas vehicles may avoid in 3 dimensions, this simplification was made with the acknowledgement that at high enough vehicle densities this affect would even out from layer to layer as each layer has approximately the same number of vehicles per km^2.

During the simulation, counters were kept for each simulation cycle of the path (approximately one second of simulation time) at 6 m/S nominal velocity, but slowing by 15% when engaged in Detect and Avoid maneuvering due to an obstruction or detection of another vehicle.

Our hypothesis was that the number of cycles a vehicle was engaged in obstruction (building) DAA would remain relatively constant even as the number of vehicles increased, depending primarily on the placement of the path in relation to the clusters of buildings. The number of interactions between vehicles, however, would be expected to increase.

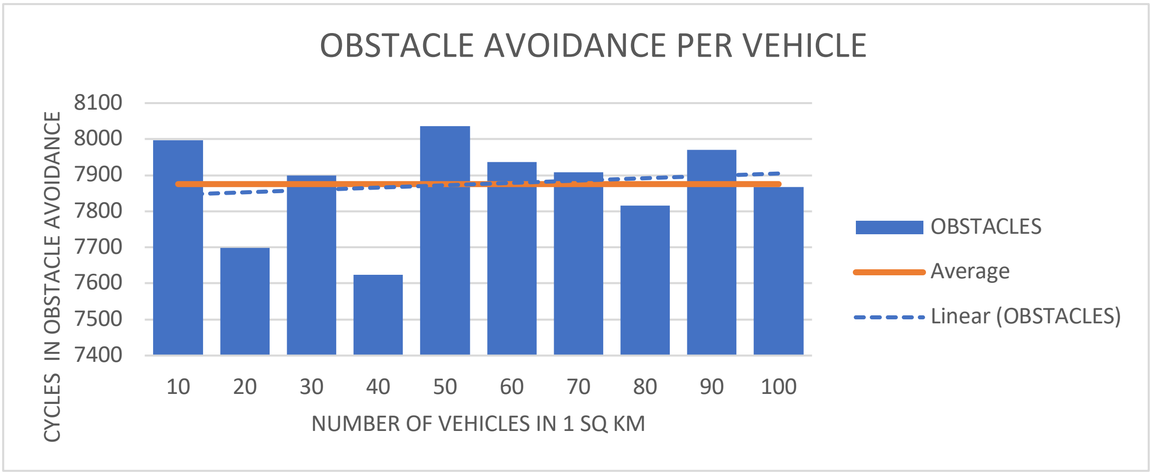


Figure 12. Number of DAA Cycles with Obstacles.

The number of simulation cycles vehicles spent in obstruction DAA maneuvering in relation to the number of vehicles in the airspace is shown in Figure 12. While the linear regression of the overall trend is slightly increasing, it does support the first hypothesis.

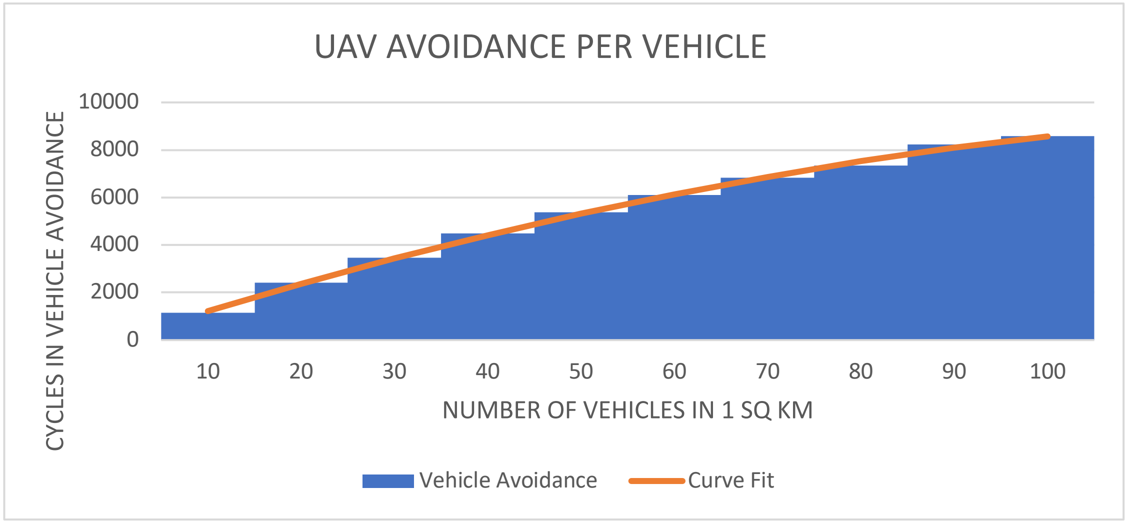


Figure 13. Number of DAA Cycles with Other Vehicles.

The number of cycles in Vehicle to Vehicle (V2V) DAA maneuvering with respect to the increasing number of vehicles is graphed in Figure 13. This does show that the time in avoidance increases, but at more of a logarithmic rather than exponential fashion.

The number of cycles spent in avoidance, however, is not the only contribution to the variability in the total time of flight. The amount of deviation of the flight path from the predictable straight-line path is also a key component, as shown in Figure 14.

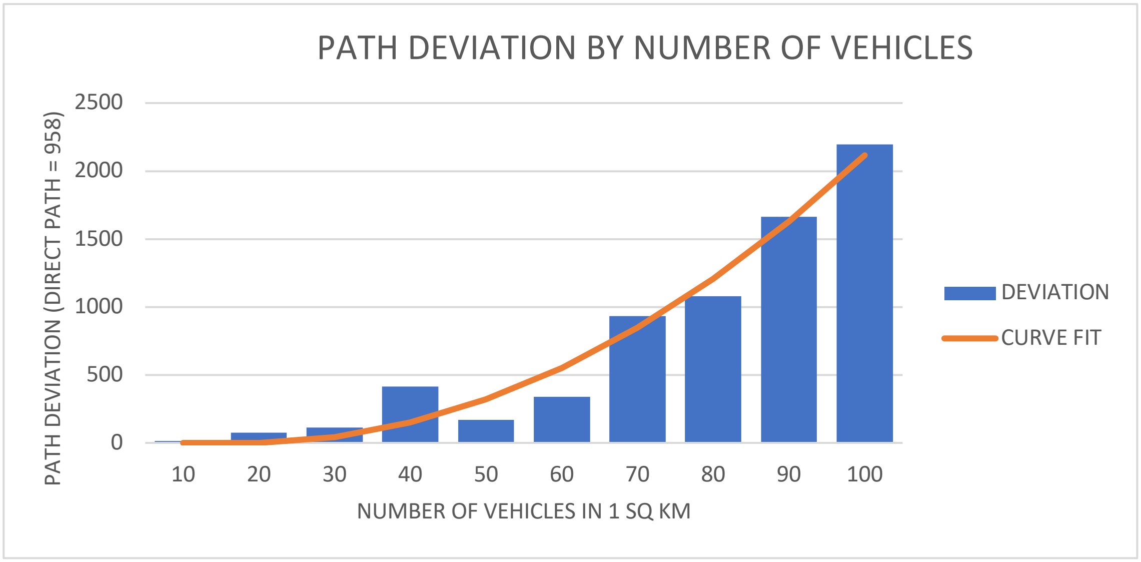


Figure 14. Total Deviation from Straight Line Path.

The length of each flight path directly from source to destination was 958m. This path shows that in addition to the time lost in the DAA process itself, the heading changes can become significant with the increase in vehicles. Additionally, cascade DAA encounters where a vehicle experiences multiple DAA course corrections before it can return towards its destination heading become more commonplace.

4-Dimensional Path Searches

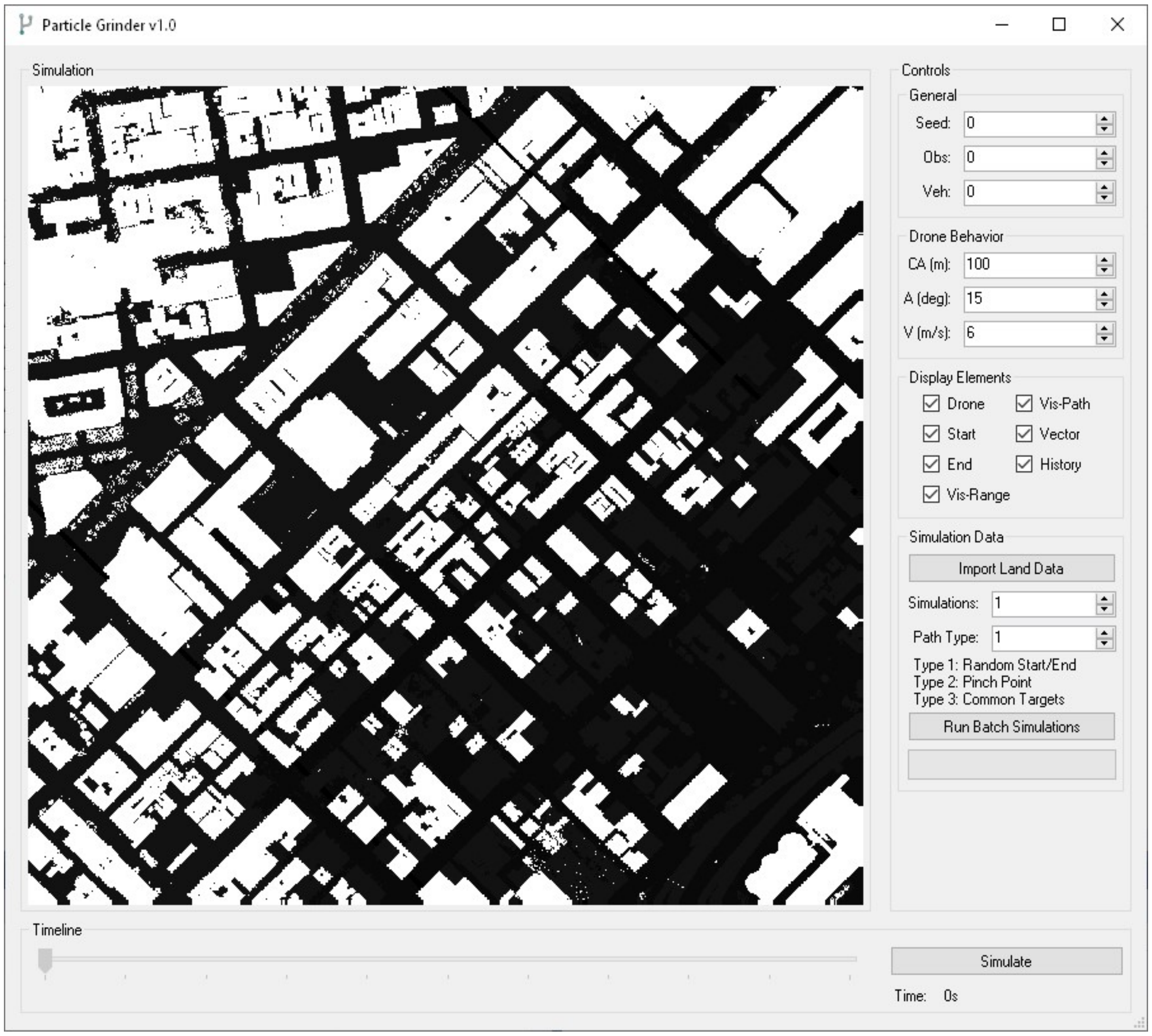
Our initial 4D path routing tests randomly inserted up to 350 vehicles into a dense urban airspace that includes large skyscrapers, roads, population centers and natural topographic variation.

Early versions of the system focused on demonstrating proof-of-concept functionality in core elements such as integration of geospatial inputs, algorithm architecture, database and server network architecture, grid search components, a simple urban use-case cost function, and rudimentary path smoothing.

In the tests shown here we used a 10m DEM over a 1 km^2 landscape with uniformly spaced sources and randomly assigned destinations. The base elevation model was based on 10-year-old data, and thus did not include some of the newly constructed towers. Early runs involved 70, 150 and 350 simultaneously operating vehicles. The relatively poor resolution resulted in a high rate of apparent intersections either with buildings which show in Google Earth but were not in our model, or paths cutting into corners of buildings. These were eliminated with more complete GIS information including the 1m resolution raster data as shown in Figure 15.

Our second model improved the base data with 1m elevation raster that was compiled from 3 data sources; 1) a 2010 LIDAR point cloud dataset that was converted into a 1m elevation raster,

2) a “tall building” database for buildings exceeding 150 feet, and 3) a general building database for the remaining buildings. The latter two databases were provided by the City of San Francisco and were accurate as of January 2020.



Seed: 0
Obs: 0
Veh: 0
CA (m): 100
A (deg): 15
V (m/s): 6
☑ Drone ☑ Vis-Path
☑ Start ☑ Vector
☑ End ☑ History
☑ Vis-Range
Simulations: 1
Path Type: 1
Type 1: Random Start/End
Type 2: Pinch Point
Type 3: Common Targets
Time: 0s

Figure 15. 1m Raster of Obstructions.

The paths generated here were done using only the grid-based autorouter with a grid setting of 10m. This was primarily done to characterize the computational runtime of the grid system using only this mode. As expected, the runtime per line was slow, averaging about 10 minutes per line route. As maze routers in general are $O(N^2)$, this was not a surprise. While the expansion to nearest neighbors was in three dimensions, the depth of the altitude dimension with respect to the size of the overall area is small, rather like an autorouter computing paths in a multi-layer printed circuit board. If the resolution of the grid model were brought down to 1m grids, however, this dimension would become more significant and if considered at $O(N^3)$ the ten-minute run time would be multiplied by 1000 times.

It must be acknowledged, therefore, that to perform searches under these conditions, every solution must be brought to bear to increase the speed of the path calculations. The probe routing and multi-agent systems approach is one tool, switching from a breadth-first search to a depth-first search might be another. Figures 16 and 17 provide a view of the deconflicted paths generated. All paths are initiated at the same time, and one can still see the 45-degree angle turns which are an artifact of the grid router. The line colors are simply to provide more visibility between the routes. Routes which are closer than 10m are deconflicted in time. More routes and the routing area are shown in Figure 17.

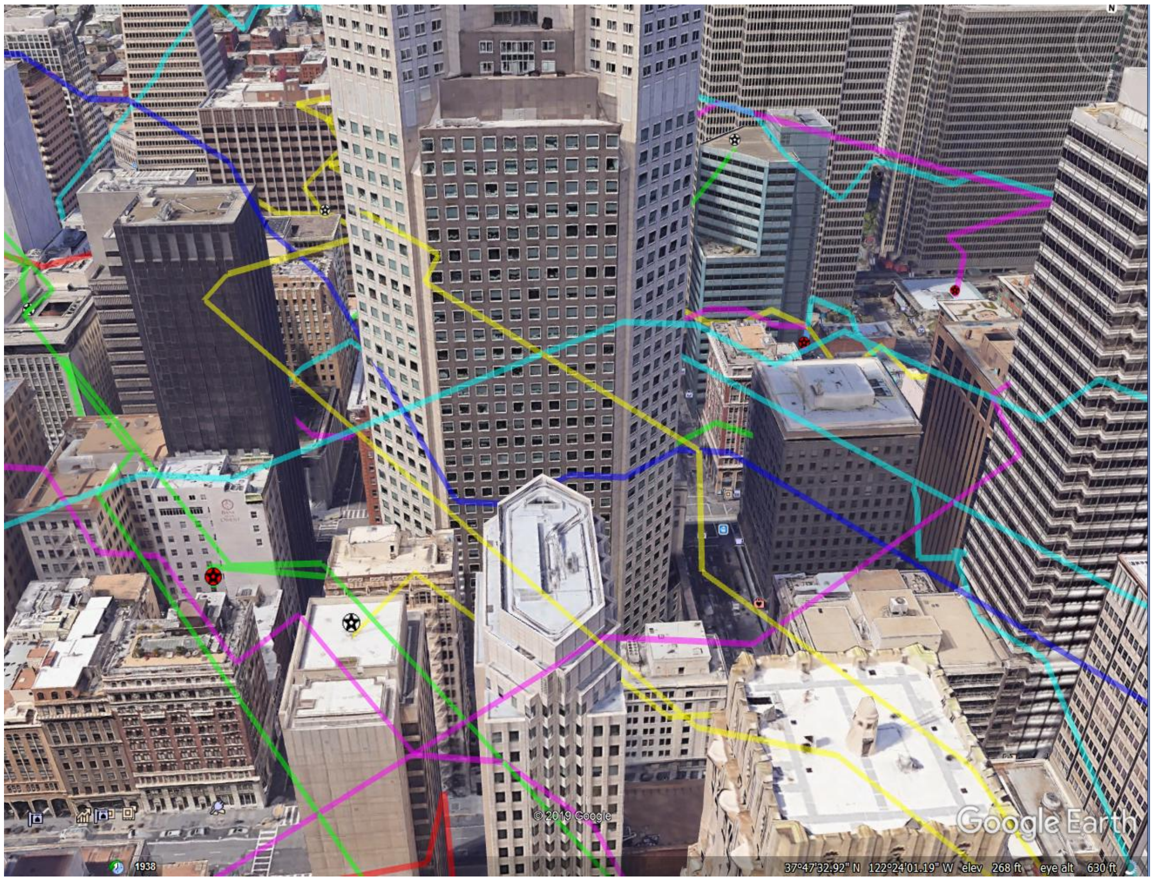


Figure 16. Path Router Results.

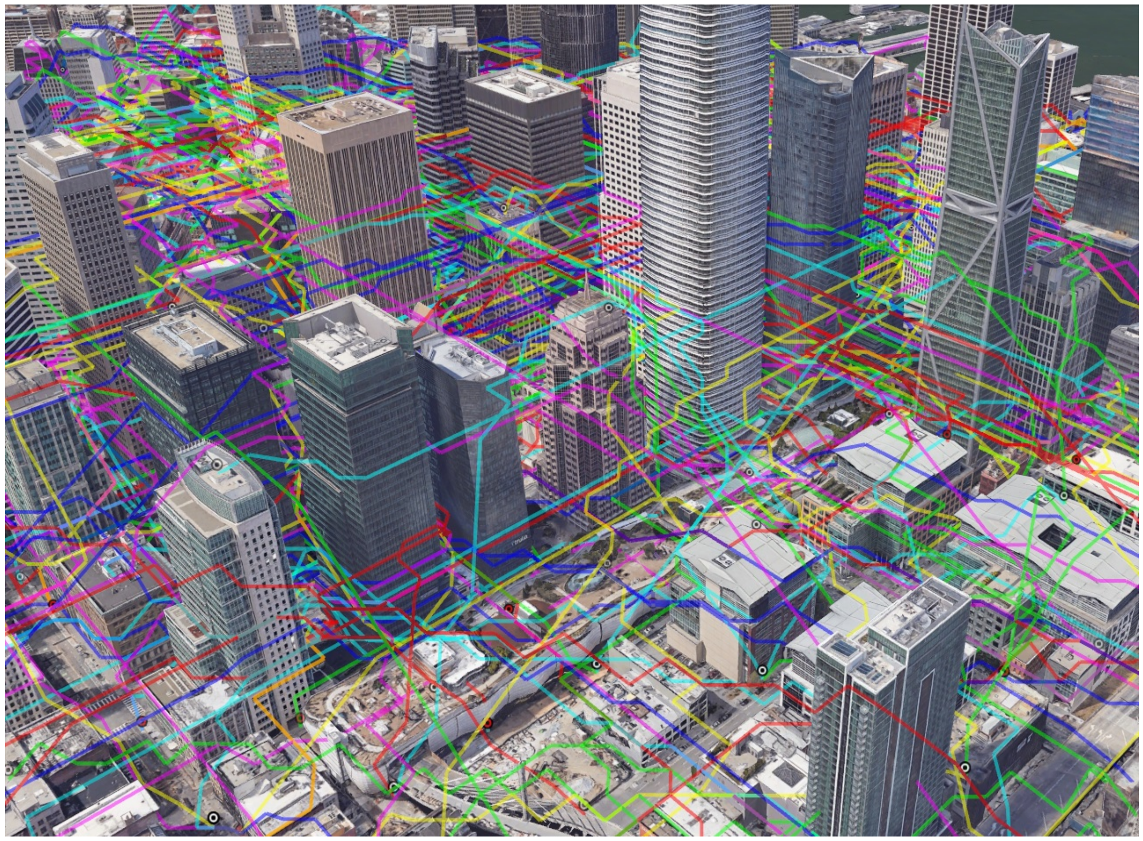


Figure 17. Wider View of Path Router Results.

Eleven simulated UAM vehicles taking off from a set given landing pads at 10m separation is simulated in Figure 18 below. They traverse the airspace and land at other cross-town vertiports 6 minutes later. A closer look at the launch point at the marina district is shown in Figure 19.

Flight Corridor Organization and Communication

Pathiyil et al. proposed simplification of the airspace by creating virtual tunnel and lanes.[17] They also noted that "UAV operations in urban areas could become extremely chaotic." An associated issue is that any static restriction of the airspace may simplify the routing problem, but it inevitably forces higher levels of conflict and inefficiency as the working or useful airspace is now significantly smaller.

Pathiyil also posits that "UTC communicates with the pilot/operator of the relevant UAVs using a robust communications link, and the study of which is necessary." This poses a problem when considered in light of Viragh's observation that any disruption or delay in communications is highly problematic and Mousa's observation that communications requirements rise exponentially with traffic density.[18, 19] While this system does not require any V2V or ground system communications, it can certainly make use of them when available. The vehicle may, for example, be able to broadcast its knowledge of other flight paths to nearby vehicles and/or obstructions, thereby assisting their DAA flight systems while still beyond their DAA detection radius. In short, because the flight plan vehicles know their entire path in advance, they make the entire airspace more intelligent.

Rather than creating fixed corridors, the path management system can create on-the-fly organic or self-organizing corridor configurations as is shown in this example. Because the system knows the overall location and timing of every vehicle, it can track that data over time to intelligently alter the cost functions accordingly. As time goes on and traffic patterns evolve, the organic corridors will evaporate and reform in other areas and directions to maximize the efficiency and throughput of the system.

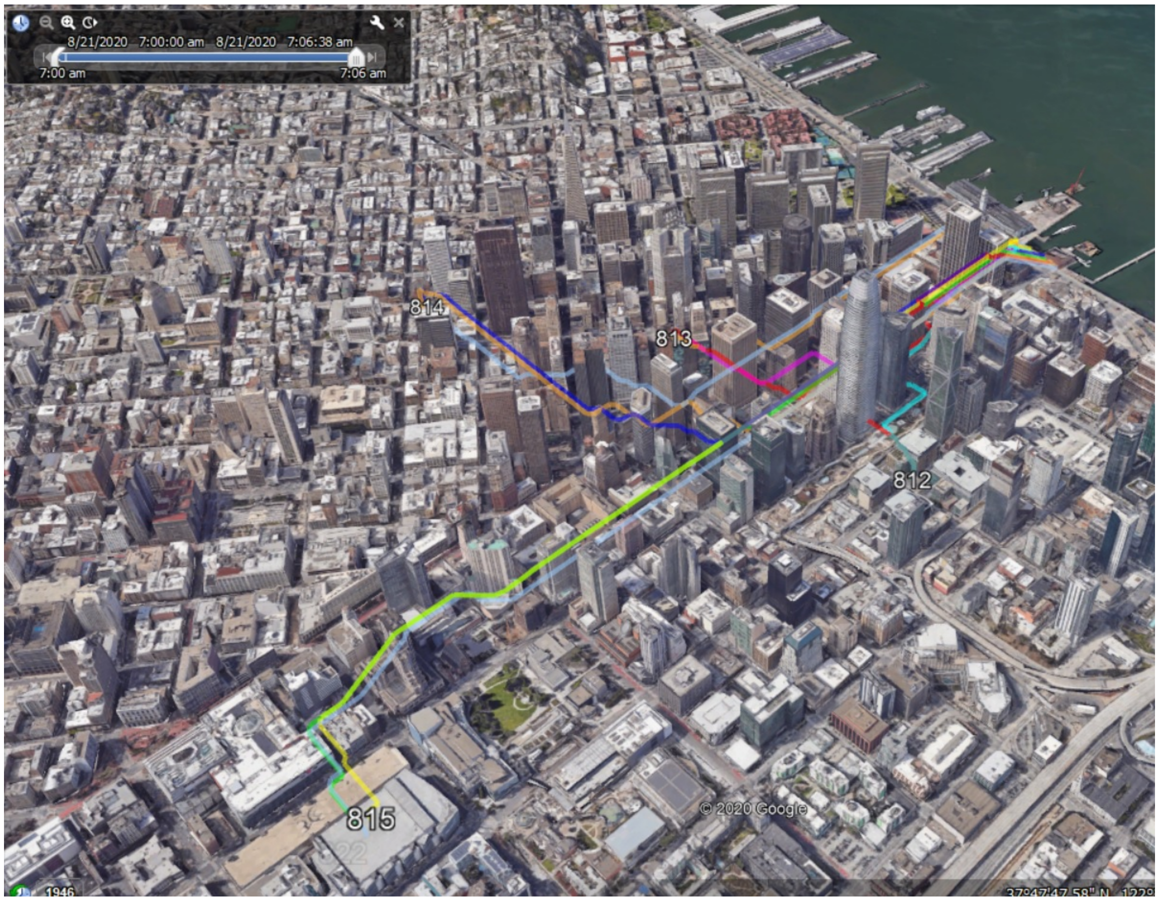


Figure 18. Self-Organizing UAM Corridor.

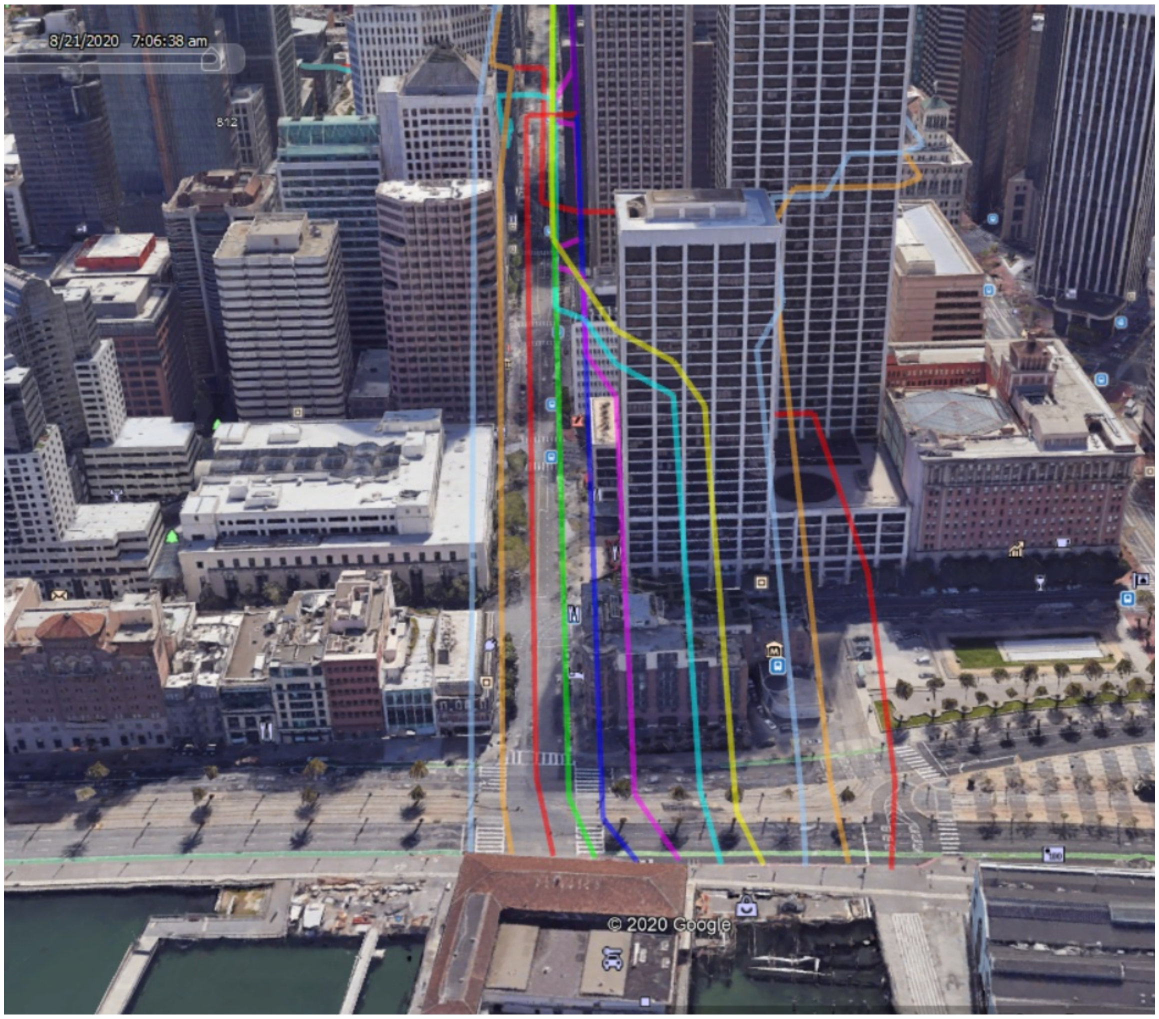


Figure 19. UAM Launch Array.

REFERENCES

[1] Xue, Min, Minh Do (2019) “Scenario Complexity for Unmanned Aircraft System Traffic.” AIAA Aviation 2019 Forum, June 17-21, 2019, Dallas, TX.

[2] United Nations Department of Economic and Social Affairs, 16 May 2018. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

[3] Dijkstra, E. W. (1959). "A note on two problems in connection with graphs". Numerische Mathematik. 1: 269–271.

[4] Lee, “An algorithm for path connection and its application,” IRE Trans. Electronic Computer, EC-10, 1961.

[5] Hightower, “A solution to line-routing problems on the continuous plane” DAC '69: Proceedings of the 6th annual Design Automation Conference

[6] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics. 4 (2): 100–107.

[7] Murray, C.C., Chu, A.G., 2015, “The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery.” Transp. Res. Part C: Emerging Technologies, 54 86-109.

[8] Barmpounakis et al. “Unmanned Aerial Aircraft Systems for transportation engineering: Current practice and future challenges”, International Journal of Transportation Science and Technology 5 (2016) 111-122.

[9] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai , Tamás Nepusz, Agoston E. Eiben and Tamás Vicsek, “Optimized flocking of autonomous drones in confined environments”, *Science Robotics* 18 Jul 2018: Vol. 3, Issue 20, eaat3536 DOI: 10.1126/scirobotics.aat3536

[10] Chakrabarty, Anjan, Vahram Stepanyan, Kalmanje Krishnakumar, Corey A. Ippolito (2019) “Real-Time Path Planning for Multi-copters flying in UTM-TCL4.” AIAA SciTech Forum, January 7–11, 2019, San Diego, CA.

[11] Pazos-Perez, Rafael, Rabuñal, Jose, and Carballal, Adrian, 2017, “Blurring the Boundaries between Real and Artificial in Architecture and Urban Design through the Use of Artificial Intelligence”, ResearchGate publication/322539555 CTBUH Journal 2019 Issue III.

[12] Danser, Raymond A. (2018) “Applying Least Cost Path Analysis to Search & Rescue Data: A Case Study in Yosemite National Park.” Master’s Thesis. University of Southern California. August 2018. 86pp

[13] Lundell M, Tang J, Hogan T, et al. “An agent based heterogenous UAV simulator design” Proceedings of the 5th WSEAS international conference on artificial intelligence, knowledge engineering, and data bases. Madrid, Spain, 15-17 February 2006, p 453-457. World Scientific and Engineering Academy and Society (WSEAS)

[14] Xuejun Zhang, Yang Liu, Yu Zhang, Xiangmin Guan, Daniel Delahaye, Li Tang, “Safety Assessment and Risk Estimated for Unmanned Aerial Vehicles Operating in National Airspace System”, Journal of Advanced Transportation Volume 2018, Article ID 4731585, 11 pages.

[15] Azza Allouch, Anis Koubaa, Mohamed Khalgui, Tarek Abbes, “Qualitative and Quantitative Risk Analysis and Safety Assessment of Unmanned Aerial Vehicles Missions Over the Internet”, IEEE Access April 2019

[16] Primatesta, Stefano, Rizzo, Alessandro, & Anders la Cour-Harbo, “Ground Risk Map for Unmanned Aircraft in Urban Environments.” Journal of Intelligent & Robotic Systems 9 May 2019

[17] Lakshmi Pathiyil, K. H. Low, Boon Hai Soon, Shixin Mao, “Enabling Safe Operations of Unmanned Aircraft Systems in an Urban Environment: A Preliminary Study” Air Traffic Management Research Institute, Nanyang Technological University, Singapore 637460

[18] Csaba Virágh, Máté Nagy, Carlos Gershenson, Gábor Vásárhelyi. “Self-organized UAV traffic in realistic environments.” Conference: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) September 2016

[19] Amjed Al-Mousa, Belal H Sababha, Nailah Al-Madi, Amro Barghouthi, Remah Younisse, “UTSim: A framework and simulator for UAV air traffic integration, control, and communication.” International Journal of Advanced Robotics September 10, 2019.