2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Safe Landing Planning for an Energy-Constrained Multicopter

Isaac J. Olson, Alec J. Ten Harmsel, and Ella M.Atkins,
Department of Aerospace Engineering
University of Michigan
Ann Arbor, Michigan

***Abstract*—As Unmanned Aircraft Systems (UAS) become more prevalent and their commercial application in populated areas becomes technologically feasible, risk mitigation techniques must be developed to minimize the probability of harm to persons and property in the vicinity of the aircraft. This paper presents an algorithm to calculate a safe landing path for a small quadrotor UAS that experiences a low energy condition while flying over a populated area. This approach uses public databases of population distributions, structure locations, and terrain to create a cost map of the operating zone of the UAS offline. The UAS uses this data to quickly identify a safe landing path using an A* search algorithm. Simulation-based case studies are presented of a UAS operating within New York City to illustrate how different cost terms impact optimal path characteristics.**

Keywords—path planning; cost modeling; risk mitigation

I. INTRODUCTION

Commercial use of Unmanned Aircraft Systems (UAS) has the potential for large growth in the near future as the regulatory environment shifts in the coming years. [1] [2] [3] [4] Many commercial applications, such as infrastructure inspection, law enforcement, and even package delivery would require UAS to operate in populated areas. A major challenge to UAS and their operators will be to reduce the risk their actions pose to surrounding persons and property. In the case of an inflight failure, [5] it will be of the utmost importance that the UAS be capable of determining and performing actions to reduce or eliminate the risk it poses to its surroundings. Additionally decision making and acting may all need to occur onboard, as loss of communication link is one of the most common UAS failures.

Another commonly-experienced failure is running low on fuel or battery energy. If unexpected, the UAS must ensure it does not introduce unacceptable risk to people or property when it lands or terminates its flight. In an urban area, the UAS may be flying directly above people or property. The UAS may therefore need to maneuver toward a new location that poses minimum risk to people and property before it needs to land or ditch.

This paper presents an emergency landing planner to calculate a safe path for a small quadrotor UAS that senses unexpectedly low energy reserves while flying over or within a populated urban canyon environment. The real-time planner uses a cost map generated offline from population, structure, and terrain data to identify low-risk landing sites and obstacle-free paths to these sites. By including weighting factors for population, structures, and terrain, the planner trades off operating in areas of dense population, flying near buildings/terrain, and traditional energy and time (path length) cost to find an acceptable landing path while minimizing the risk posed to people and property. Calculating the cost map offline and loading it onto the UAS is the key to managing planning complexity. While the proposed A*-based search algorithm has exponential complexity, preloaded maps, appropriate discretization of the search space, and limited range of the low-energy multicopter enable this approach to be feasible for onboard use. Additionally, pre-computed maps allow an accurate heuristic to be pre-computed which greatly reduces the search space. Once computed, the guidance, navigation, and control system would then use onboard sensors and flight control systems to follow the chosen flight path to the landing location. Energy levels would continue to be monitored prompting replanning if needed. Once the UAS is near the landing site or near the ground in general, a planner similar to the one described in [6] could be used to ensure best local landing site choice.

To evaluate performance of the emergency landing planner in simulation, population, building, and terrain maps were acquired for the midtown Manhattan region of New York City. A series of case studies illustrate how the emergency landing planner can provide a safe landing (ditching) capability for a multicopter UAS flying low over one of the most densely-populated cities in the USA.

This paper is organized as follow: Section II gives an overview of planner models and databases; Section III describes database manipulation for the generation of maps for cost functions; Section IV details the flight path planning algorithm; Section V presents a case study for a quadrotor operating in Manhattan; finally Section VI gives conclusions and considerations for future work.

II. MODELS AND DATABASES

The emergency landing path planner requires models of both the UAS and its environment to safely navigate the populated urban canyon environment. We chose a quadrotor UAS and modeled energy used by the vehicle based on discharge curves of lithium polymer batteries and data taken from the Michigan Autonomous Aerial Vehicles (MAAV)

team, a UAS student team at the University of Michigan.[1] We modeled the environment by gathering data on the population density and structure locations.

A. Quadrotor Energy Usage

As seen in Fig. 1, lithium polymer batteries maintain a fairly constant voltage for the majority of the discharge period followed by a quick drop-off.

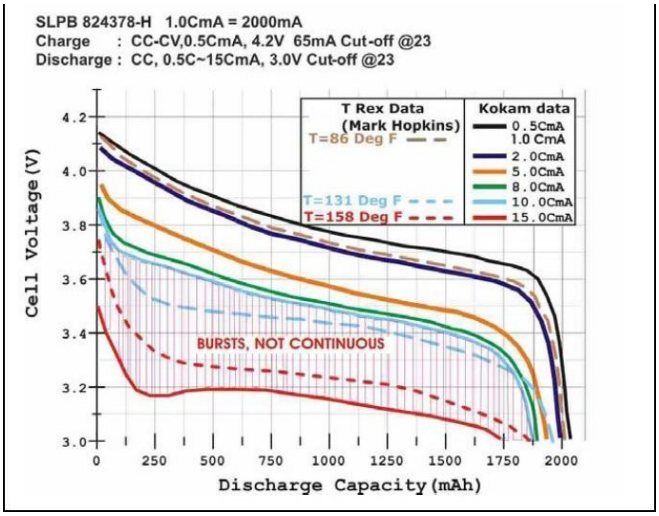

Fig. 1. Lithium Polymer Battery Discharge Curves (http://www.droidforums.net/forum/attachments/smartphone-battery-discussion/50553d1338473331-degradation-battery-discharge_curves_1.jpg)

Note that discharge ratings are in terms of the capacity of the battery 'C' so a battery that has a capacity of 2000 mAh and discharged at a rate of 1.0 C will be outputting 2 amps.

B. Environment

For the MAAV quadrotor, the batteries have a capacity of 4000 mAh and the system draws about 22-26 A during flight, which is a discharge rate of about 6.0 C. At this discharge rate and according to the Fig. 1 graph, the corner point for the cell voltage will be just under 3.4 V/cell and there will be about 500 mAh of capacity left in the battery. Thus, the vehicle will have approximately 80 seconds of power remaining.

Data for the population map of New York City and the surrounding area was acquired from the US Census Bureau for the 2010 census. Using their published latitude, longitude, population, and area values, a population density map was created using discrete data points and then interpolating a smooth surface between them. Because census blocks are rarely simple shapes that can be fit to a grid, this interpolated representation provides a reasonable approximation of the data without having to import a complete census block map, which would increase the complexity of the simulation by a large factor.

[1] The first two authors are members of the MAAV team and the third author is an advisor to the MAAV team.

The Department of City Planning (DCP) of New York City (NYC) has released a dataset called Primary Land Use Tax-lot Output (PLUTO) under an open license that contains information about every taxable lot in NYC. We incorporate PLUTO building data into our simulator that can be easily extended or updated as needed. The following PLUTO dataset fields are used in this model:

- Number of Floors
- Bounding Box
- Vector Array of Lot Boundaries

This data is listed by address for every taxable lot in an ESRI Shapefile. The full PLUTO dataset is separated by region, and the Manhattan region data is used in our simulation. The Shapefile comes with location information (e.g. the Bounding Box and Vector Array) listed in New York State Plane coordinates. The Shapefile is converted to latitude/longitude coordinates for use in our algorithms by the open-source Geographic Information System software QGIS.

III. Database Manipulation

Processing required for path planning can be divided into two parts: Preprocessing of the cost map, and online calculation of the landing path. To compute the cost map, we generated individual population, terrain, and structural cost maps, and then merged them using a weighting system to form the final cost map. The onboard search algorithm uses this cost map to perform A* search to find the path to a safe landing location. For the test cases in this paper, we used an area of the Upper West Side of Manhattan, between Central Park and the Hudson River. Satellite imagery of this area is shown below in Fig. 2.

Fig. 2. Satellite image of the area of Manhattan used in case studies for the path planner (Google maps).

A. Population Map Generation

Because the cost will be based on potential risk to people, the vehicle's impact radius (that is, the radius outside of which people will be presumed safe) must be assessed. For the MAAV quadrotor, a small and lightweight vehicle, we

will consider this radius to be $2\,m$. Therefore population densities above $0.0795\ persons/m^2$; that is, an average of one person within the impact radius, will be ineligible for a safe landing. This constraint may be relaxed when lower-population areas are unable to be reached. Once the visual resolution is sufficient to identify individuals, targets within $2\,m$ of a person will be considered ineligible. Therefore, to calculate the population cost of individual grid spaces, we will use $0.0795\ persons/m^2$ as the normalizing factor on the population density to convert the density grid to a normalized population cost grid for which cost values greater than 1 will be clipped at 1. The population cost map generated for our intended flight area is shown in Fig. 3. In future work database information on population can be supplemented by real-time sensor data (e.g., cameras) to ensure the UAS can account for unexpected people and objects near the emergency landing site.

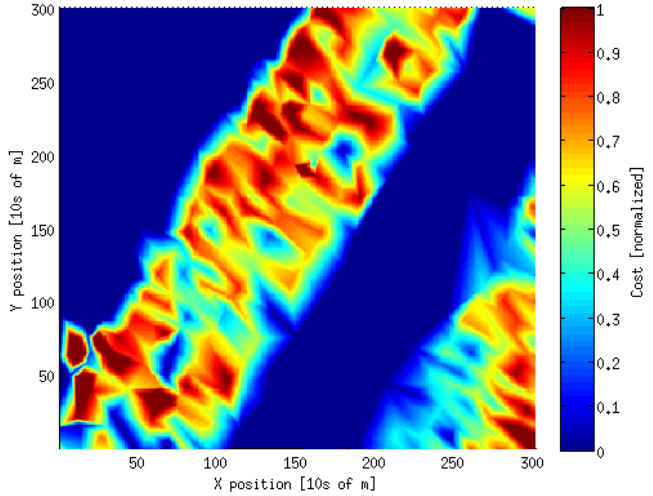

Fig. 3. Population cost map generated with population density normalized by $0.0795\ persons/m^2$.

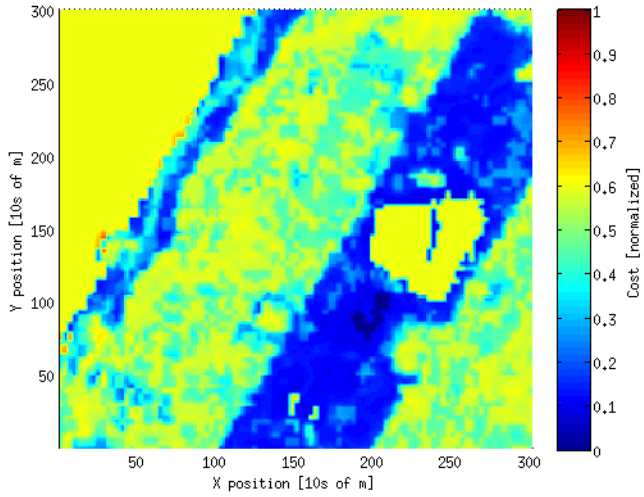

Fig. 4. Terrain cost map prioritizes areas that are flat and undeveloped, such as near the Hudson riverbank and Central Park.

B. Terrain Map Generation

To develop an initial terrain cost, we used elevation, terrain class, and imperviousness percentage data available from the U.S. Geological Survey. An individual cost for each data type was developed and then weighted and summed to calculate the total terrain cost. The elevation data was used to calculate the slope of the ground. The terrain-slope cost is normalized by a slope of 45° (cost = 1.0) to prioritize flat areas (cost = 0.0) as landing zones. The terrain class data distinguishes the types of terrain, e.g., with water given higher cost than land, as will be an important distinction for our case studies. The terrain cost map generated for our intended flight area is shown in Fig. 4.

C. Structural Map Generation

The Shapefile of Manhattan provided by the NYC DCP contains vector arrays for the boundaries of every taxable lot in the city as shown in Fig. 5. To generate the map used in the simulation, the vector information in each entry of the Shapefile is used to create a matrix with the height of the building at each position if one exists as shown in Fig. 6.

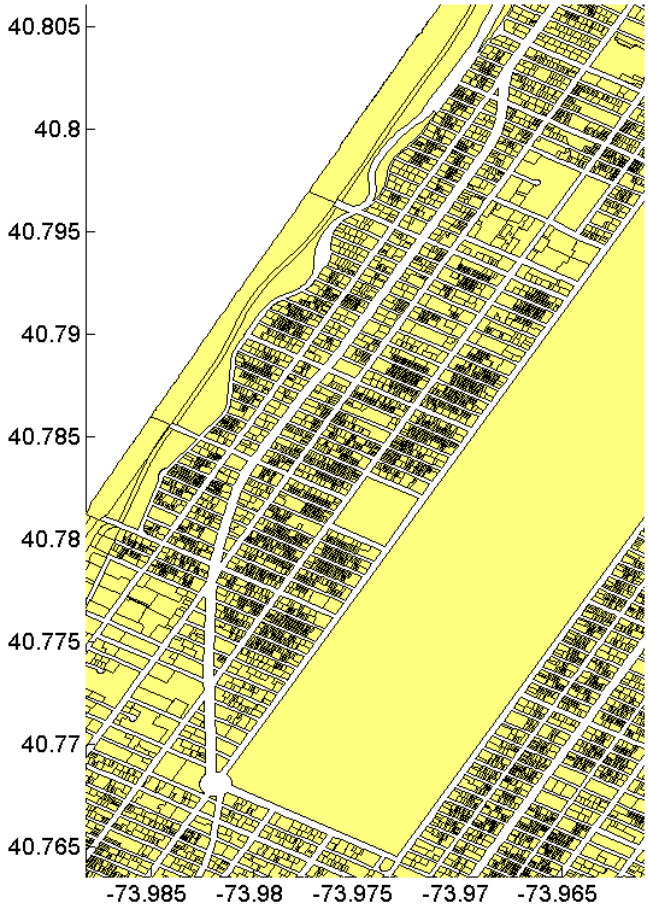

Fig. 5. Matlab representation of the Shapefile.

1) Shape File Usage

Height maps generated by the Shapefiles are limited in resolution by both memory and run-time constraints. This simulation uses a spatial resolution of 10 meters to optimize

resources. The height map is represented as a 3D occupancy grid.

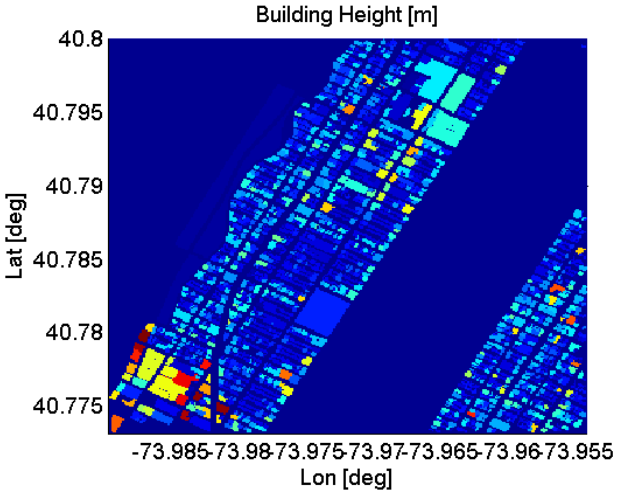

Fig. 6. Height map created from the Shapefile. Dark blue represents no buildings; light blue to yellow to red depict increasing building heights.

2) *Cost Gradient Repulsion Around Obstacles*

To increase safety during landing path following, it is important to prevent the vehicle from flying immediately next to edges of obstacles in order to travel the shortest path. This is accomplished by inflating the cost in the grid cells surrounding grid cells occupied by obstacles. Because the grid cells near obstacles have a higher cost than those slightly farther away, the vehicle will tend to keep a wider safety margin from obstacles while maintaining a reasonably short path length.

The algorithm to create the repulsion fields in a grid map is divided into two parts. The first part calculates the repulsive cost caused by a given obstacle at some grid cell near the obstacle. The second part of the algorithm searches through the grid map and applies the cost function to each cell surrounding the obstacles. Because the cost calculation function is separate from the search function, the behavior of the repulsion field can easily be modified to produce different shaped gradients. One possible repulsion cost function maps a linearly decreasing cost to the area surrounding the obstacle, with obstacle location L_O, target location L_T, repulsion field width W, and location cost function C. The formula for the repulsion cost function C_R is given by Equations (1) and (2).

$$C_R(L_T) = C(L_O) \cdot \left(1 - \frac{\text{norm}(L_T - L_O)}{W}\right) \quad (1)$$

$$C(L_T) = \max\left(C_R(L_T), C(L_T)\right) \quad (2)$$

An example two dimensional repulsion field is shown in Fig. 7. The actual repulsion fields generated in our case study are three dimensional, creating regions of higher cost both along the sides of known structures as well as above them.

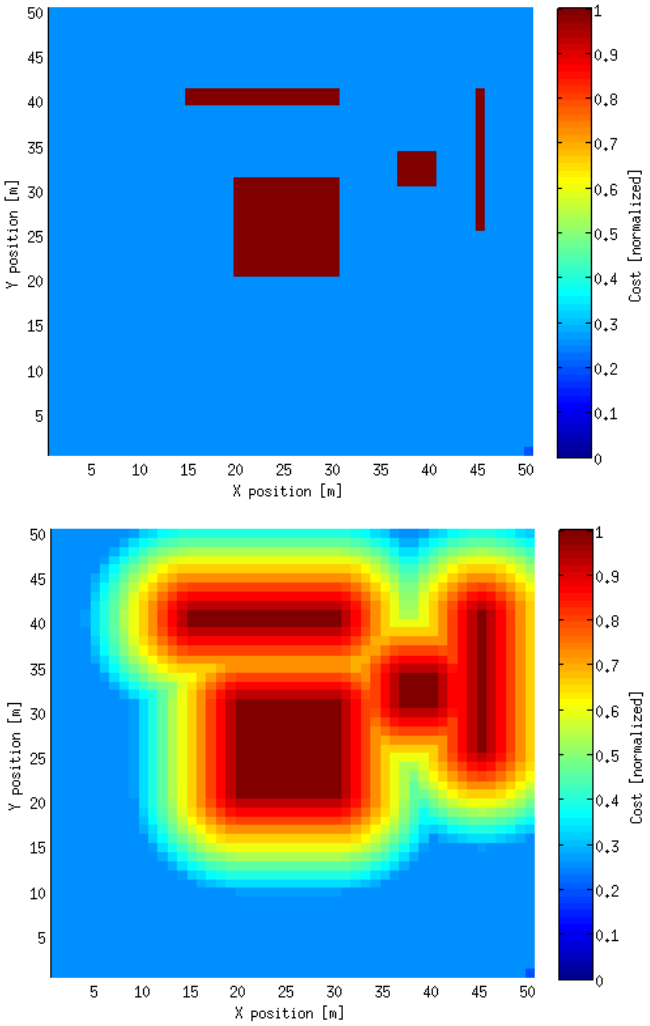

Fig. 7. Cost map for four distinct buildings (top); cost map of the same buildings with repulsion fields (bottom).

D. *UAS Acceleration and Movement Costs*

The onboard search algorithm seeks to produce in real time a suitable trajectory to a safe landing, taking as input the combined cost maps discussed above. It is also critical to properly model remaining battery capacity at each step in the landing path. Battery energy loss is calculated using acceleration and movement cost functions. Acceleration cost is computed from the current vehicle velocity v_{in}, target velocity v_{out}, remaining battery capacity B, maximum battery capacity B_{max}, time step dt, UAS mass m, and acceleration due to gravity g. Function D provides a simple translation of acceleration magnitude, battery capacity, and time step to expected current draw over this time step. An approximate acceleration cost term C_A is then computed according to (3) through (5).

$$dv = v_{out} - v_{in} \quad (3)$$

$$\frac{dB}{dt} = D\left(\text{norm}\left(\left(\frac{dv}{dt} + g\right) \cdot m\right), B, dt\right) \quad (4)$$

$$C_A(v_{in}, v_{out}) = \left(\frac{dB}{dt} \cdot dt\right) / B_{\max} \quad (5)$$

Current draw function D was developed from curve fitting experimental data collected during motor tests for the Michigan Autonomous Aerial Vehicle (MAAV) quadrotor which uses the Axi Gold 2212 motors, Castle Creations Pheonix 25 electronic speed controllers and 9 inch propellers. Tests were conducted when operating at the nominal 3.4 V per cell.

The acceleration cost expression was used in the computation of a movement cost term C_M. Define L_{out} as a goal position and L_{in} as a starting position for movement cost function C_M. Then C_M can be approximated by (6) through (8).

$$n = \frac{\text{norm}(L_{out} - L_{in})}{speed} \quad (6)$$

$$v = \frac{L_{out} - L_{in}}{n} \quad (7)$$

$$C_M(L_{in}, L_{out}) = 2 \cdot C_A(0, v) + (n-2) \cdot C_A(v, v) \quad (8)$$

Movement cost is computed assuming ramped initial acceleration and final deceleration steps between which constant speed (v) is commanded, representing the desired traversal speed. The above formulation presumes acceleration and deceleration have the same cost, an assumption consistent with MAAV experimental data.as the small UAS was not able to complete the maneuver.

Fig. 8 and Fig. 9 show the expected acceleration cost given constant travel distance and constant battery capacity respectively. This information was used to compute the expected maximum number of time steps in a valid solution flight path. This was not an exact limit; however, as the actual accrued acceleration cost was dependent on how the vehicle maneuvered throughout the path.

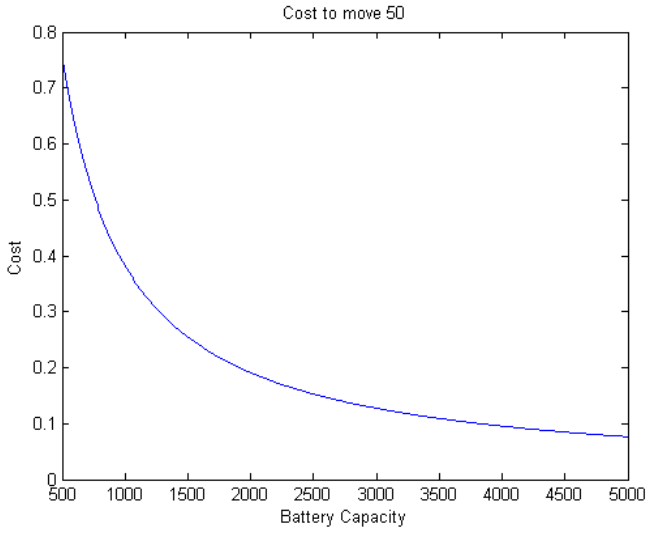

Fig. 8. Cost C_M of moving 50 meters expressed as a percentage of initial battery capacity. .The function exponentially decays as initial capacity increases, consistent with experimentally-derived models of battery performance.

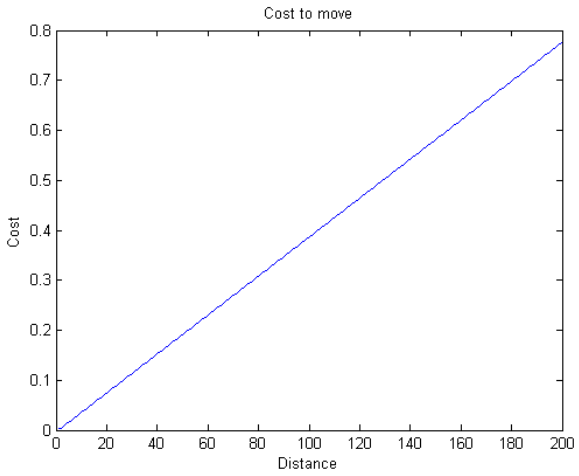

Fig. 9. Movement cost C_M presuming an initial battery capacity of 2000 mAh. This plot shows the expected movement cost is linear in distance traveled given that the battery is not nearing depletion for this case.

IV. SEARCH ALGORITHM

The suite of environmental map and UAS movement costs collectively describe a cost map over a local 3D grid in which a landing path must be planned. A traditional A* search algorithm [7] was applied to this problem. The total cost C_{total} of expanding a node L is given by the movement cost (C_M) from the previously expanded node L_l plus a weighted sum of the population (p), terrain (t), and structure distance (d) costs at that node, shown in (9).

$$C_{total}(L) = C_M(L_l, L) + \begin{bmatrix} w_p & w_t & w_d \end{bmatrix} \begin{bmatrix} C_p(L) \\ C_t(L) \\ C_d(L) \end{bmatrix} \quad (9)$$

Weighting factors w_p, w_t, and w_d are left as configuration variables used to specify tradeoffs between cost terms. Note that since all terms are normalized the weighting terms will directly indicate the relative values of the different cost terms. A* search begins from the current quadrotor state. The quadrotor is commanded to hover during the brief time (see below discussion) required to generate the emergency landing path. A* terminates when it finds a viable solution or until it exhausts all path options within the vehicle's remaining range. A viable solution must have cost less than a specified value, called *goal_max*, which is left as a configuration variable. For this paper, the search algorithm is more appropriately labeled uniform cost or Dijkstra's algorithm because no heuristic estimate of cost has yet been incorporated. A true A* implementation is expected in the near-term using a heuristic expected to substantially reduce average search time as discussed as future work in the final paper section.

A. Algorithm Complexity

The emergency landing planner must execute in real time, likely within a few seconds. Pre-calculation of the map-based costs enables rapid computation of cost for each search node, but the search itself still must be performed. While general

search has exponential complexity, we can bound the search space as a function of the bounded and discretized region the multicopter can reach. The proposed search has complexity contains $O(n^4 \log_2(n))$ where n is the maximum number of cells along any side of the 3D search map. n^3 is the maximum number of nodes to expand and an $O(n \log_2(n))$ sort of a priority queue of nodes is performed at each step.[2] In practice the algorithm typically explores far fewer nodes.

When running on a single core of a 2.50 GHz processor, the searches in our case study averaged around 60 seconds in run time using uncompiled, unoptimized MATLAB code.[3] This algorithm could be implemented in a compiled language onboard a UAS for real time landing zone calculation. Even in the event that the algorithm still requires between 5-10 seconds of execution time, having the vehicle hover is a viable option provided the algorithm is called when sufficient flight time remains for both the computation and execution of the safe landing.

V. MANHATTAN CASE STUDY

For our case study, we used an area of Manhattan bounded by $(40.773°N, 73.990°W)$ and $(40.800°N, 73.954°W)$, which forms a 3 km by 3 km square. The cost map was formed in this region with 10 m x 10 m x 10 m cubes, extending up to 200 m above ground level.

A. Test Matrix

The emergency landing path planner was evaluated with a test matrix constructed over four variables. The test matrix includes starting location L_0, starting battery energy B, weights w_p, w_t, and w_d for cost map generation, and the maximum allowed cost of a goal $max(C_{total}(L_F))$. Below is the test matrix used for this case study that examines the impact of varying the tunable simulation parameters.

TABLE I. SIMULATION TEST MATRIX. EACH VALUE IS GIVEN A LABEL TO FACILITATE REFERENCE. SIMULATIONS WERE PERFORMED OVER ALL COMBINATIONS OF THESE VALUES.

L_0	Easy 1, Easy 2 Hard 1, Hard 2
B	20%, 10%, 5%
Cost Weights (w_d, w_p, w_t)	CM-S: (1,0,0) CM-P: (0,1,0) CM-SPT: (1/3,1/3,1/3) CM-SP: (1/2,1/2,0)
$max(C(L_F))$	0.05, 0.1, 0.2

[2] In the given expression, the worst-case node queue size is n^2 rather than n^3. This reduced value is based on data collected over our case studies; formal derivation of upper bound on queue size is left to future work.

[3] By all authors' experiences, compiling Matlab code and/or converting code to a lower-level [compiled] language reduces run time by one or more orders of magnitude.

There are four starting locations L_0: Easy 1 and 2, and Hard 1 and 2. The Easy 1 and 2 initial positions are the same latitude and longitude that is in a high cost area with multiple low cost areas nearby but different starting altitudes. The lower altitude test (Easy 1) starts 20 meters in the air, and the higher altitude test starts 70 meters in the air. Both Hard 1 and 2 start the UAS in a location of high cost also surrounded by locations of high cost. Note that starting in a location of acceptably-low cost is a degenerate case as the multicopter can then just directly descend to a safe landing. Both Hard 1 and Hard 2 start at 20 meters of altitude. The maximum capacity of the UAS battery is set to 8000 mAh, and the different starting energies specified as a % of the maximum capacity simulate low-energy conditions.

The cost weights for structures, population, and terrain terms are used to generate the cost-map used in search. The structures cost weight controls the weight of the repulsion fields around buildings. Note that even a zero structure distance cost weight does not allow the UAS to fly into a building because hard constraints are placed on building or terrain impact. The cost metrics in the test matrix, CM-*, are in the format (w_d, w_p, w_t).Using CM-S, for example, generates a cost map purely based on repulsion fields as well as the UAS movement cost C_M which always has unity weight. CM-P generates a map based on population density, CM-SP equally weights structure and population costs, while CM-SPT equally weights all three environmental cost terms. Movement cost is always equally balanced with total environment cost for this case study.

Cost $max(C(L_F))$ is examined for values between 0.05 to 0.2. A cost of 0.2 is generally found in large intersections or the bank of the Hudson River, while a cost approaching the low level of 0.05 is only found in very low population density areas, often in bodies of water or the open areas of Central Park. Cost maps and results for the mid-town Manhattan case study are presented below.

B. Results

All combinations of parameter values from the test matrix were run in simulation. The emergency landing planner returned viable solutions for all test cases. Depending on the relative cost weights and remaining energy B, the planner varied path characteristics, selecting paths that either weaved through urban canyon corridors or gained altitude to fly above the canyon. Specific solution characteristics are presented below.

Before examining example landing paths we first examine how the environment cost map changes as a function of altitude. Because population and terrain cost are considered constant at every altitude, only the structures and repulsive cost gradients around them contribute to cost changes at different altitudes. As expected, the map generally has fewer obstacles and high cost areas at higher altitudes because there are fewer buildings. This is a primary driver for selecting paths that climb above building tops, along with the added benefit of taking more direct routes once above all buildings. Fig. 10 shows the environment cost maps generated for mid-town Manhattan (west side) at three different altitudes.

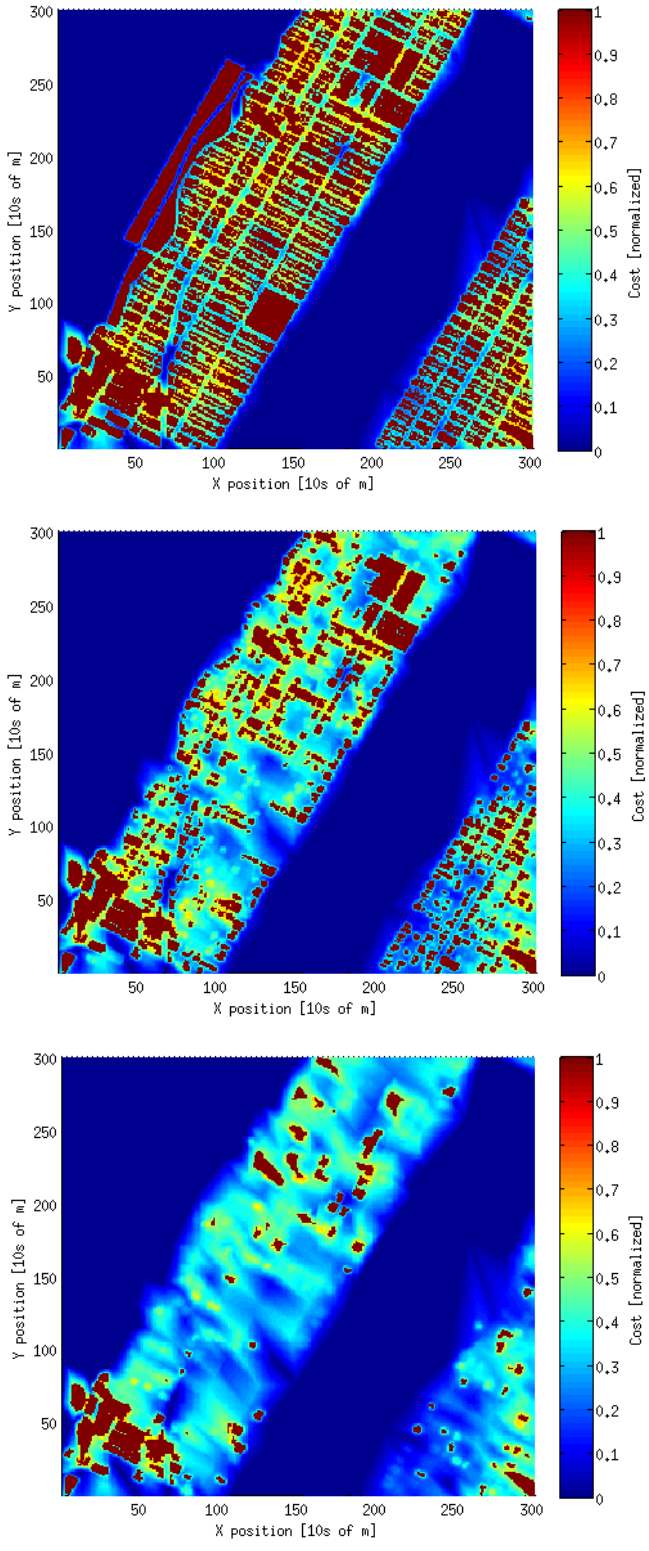

Fig. 10. Cost map generated using equally-weighted population and structure costs at ground level (top), 50 m (middle), and 100 m (bottom) altitudes. Structure based costs trend to 0 at high altitudes while population based costs remain constant.

1) Planning with only Structure Cost

In test cases where the cost map was only weighted by structures and repulsion gradients around structures, the planner can find emergency landing (goal) sites in the middle of wide streets. This is clearly not desirable behavior for an actual planner operating in a populated area and demonstrates the need for additional information besides structural maps. Fig. 11 shows the planner's behavior starting at a low altitude near the center of our mid-town Manhattan test area.

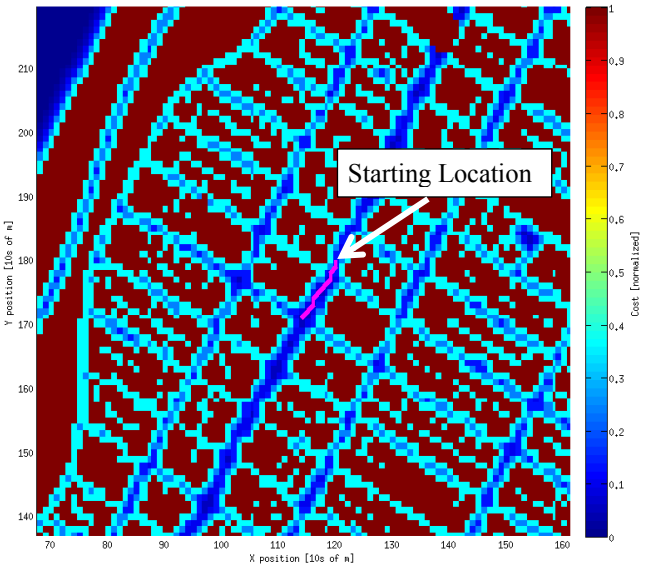

Fig. 11. Environmental cost map with only structural costs considered. With this map the planner considers wide streets feasible for landing. An example path is shown as a purple line near the center of the plot; the quadrotor traverses a short distance and lands in an intersection.

2) Navigating Urban Canyons

Our tests placed the initial condition of the vehicle inside an urban canyon with the option to either navigate the canyon on the way to a goal destination or to gain altitude and fly above the buildings. The choice between these two modes of operation was heavily influenced by the relative environmental map cost weights.

In cases when the structure cost weight was zero the vehicle did not have a significant incentive to fly above the buildings so the search typically generated paths navigating through the urban canyon. This behavior is shown in Fig. 12.

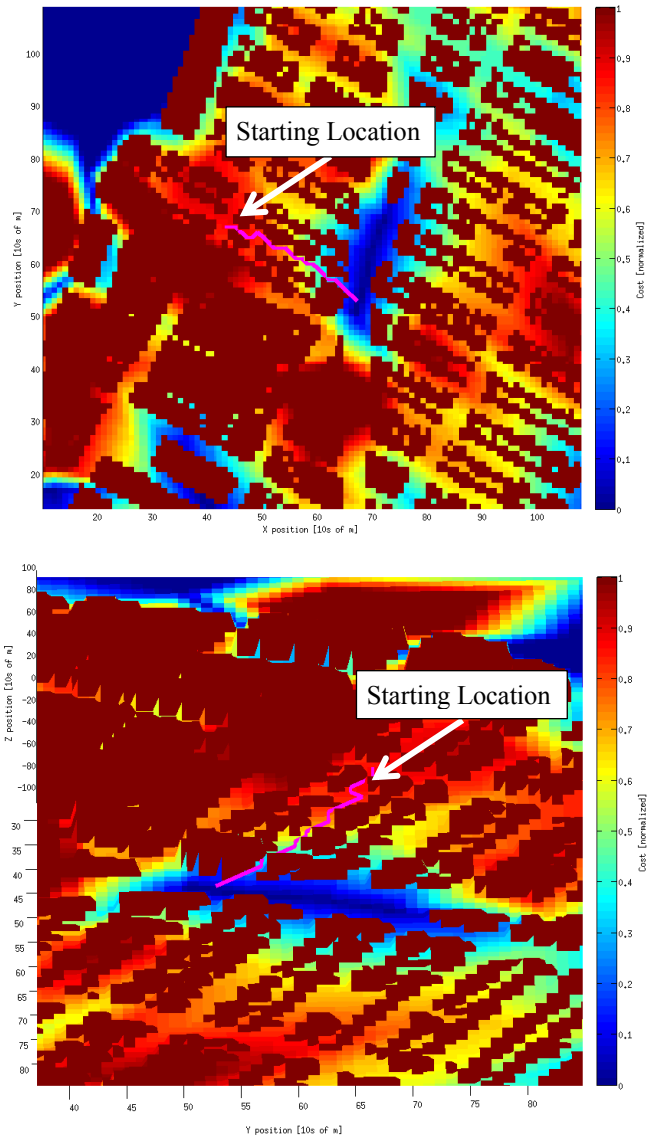

Fig. 12. With only population cost, the vehicle navigates down a street to a landing location with low population as seen from an overhead view (above) and an orthographic view (below). The example landing path is shown in purple, landing in the low-cost (blue) region.

Because there was no repulsion cost around structures, the cost in some wide streets and intersections was still sufficiently low for these areas to be classified as a viable landing locations.

An example with only population cost (Fig. 13) illustrates a situation in which cost is sufficient to drive the landing site away from population, in this case toward the Hudson River. The path for this case primarily follows a street but travels over the roofs of shorter nearby buildings. This case shows that although in general neglecting the structural repulsion cost gradients can cause the planner to land in the middle of intersections or similarly undesireable locations, in many cases it behaves similarly to the planner that uses both structure and population costs.

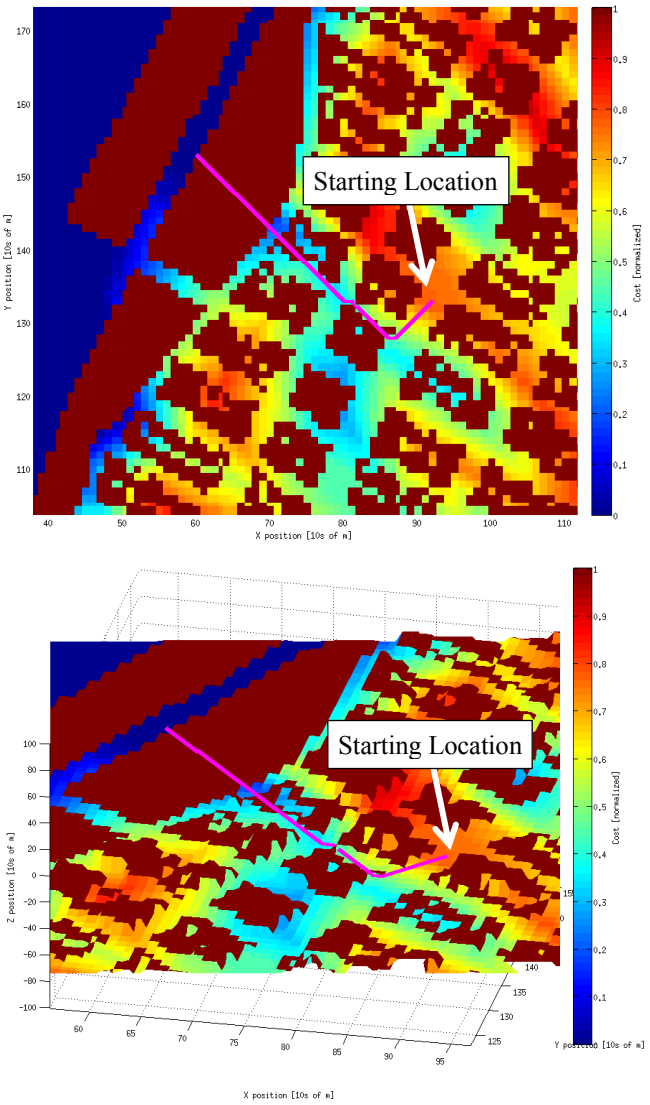

Fig. 13. Case in which only population cost has non-zero weight. The landing path flies near obstacles but escapes populated areas by landing near the Hudson River.

3) Flight Around Tall Buildings

With a low initial state altitude, the planner typically creates a path which flies over shorter buildings but around tall ones. This behavior represents a compromise between the added cost of gaining altitude and the reduced path cost from a more direct route to the goal. Fig. 14 and Fig. 15 show paths that exhibit this type of maneuver. The former uses only the population cost plus the hard obstacle constraints always considered. In the latter, the cost map is equally weighted between population, terrain, and structure repulsion costs. This causes landing goal to be selected nearer the shore of the river as well as causing the path to give buildings a wider berth.

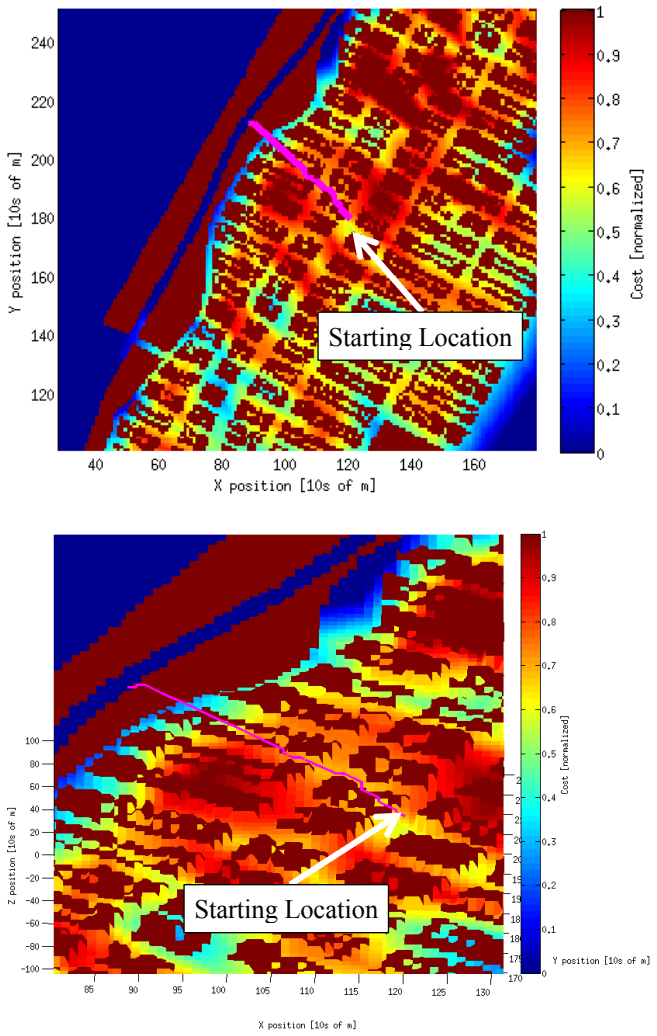

Fig. 14. A typical path with a population-only environmental cost map. The vehicle maneuvers to fly over streets and building that have a low reported population density in comparison to their surroundings. Additionally, because the structural cost is disabled, the path is not penalized for passing very near to structures while maneuvering through streets.

As demonstrated in Fig. 15, an additional benefit of using this cost map is that the addition of the terrain cost allows the planner to differentiate between locations on the riverbank and inside Central Park. Because population and structure costs are near zero for both of these zones, even a small terrain cost biases landing site choice to the most appropriate (low and level) option.

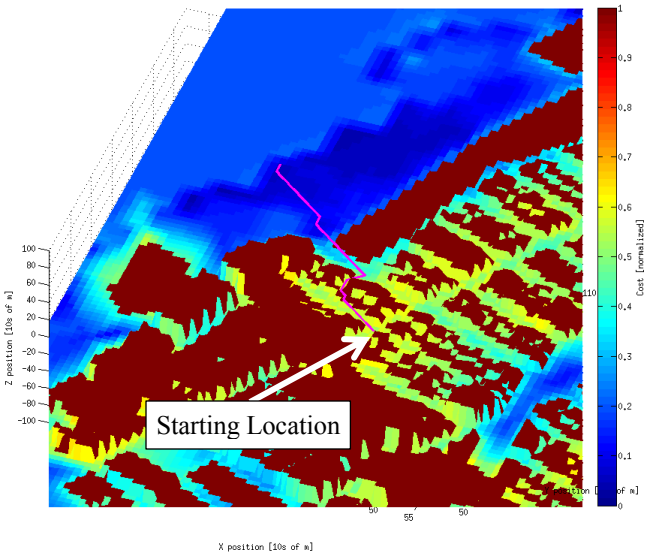

Fig. 15. Example solution with equally-weighted population, terrain, and structure costs. The selected path and landing site are modified relative to previous solutions based on variations in terrain. These variations are small for Manhattan but still influential with respect to landing site selection.

4) Flight Over Buildings

In some situations, such as when the energy reserves of the vehicle are relatively high (e.g., 20%) and structure costs are included, the optimal path may gain altitude and fly directly over all the buildings. Fig. 16 shows an example of this behavior.

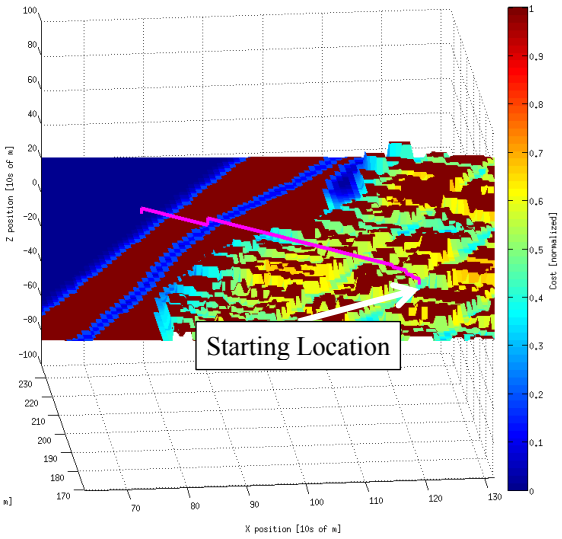

Fig. 16. When the vehicle has substantial remaining energy reserves, it can gain altitude and fly over surrounding structures.

For a case with the same initial conditions as those shown in Fig. 16 but with the cost map including terrain cost, similar behavior is observed in that the path flies to the landing site

over all buildings as shown in Fig. 17. However, in contrast to the prior test, when terrain costs are added, the planner finds a different goal landing site: a solution in Central Park instead of by the Hudson River, which will provide a safer landing for the vehicle based on terrain characteristics.

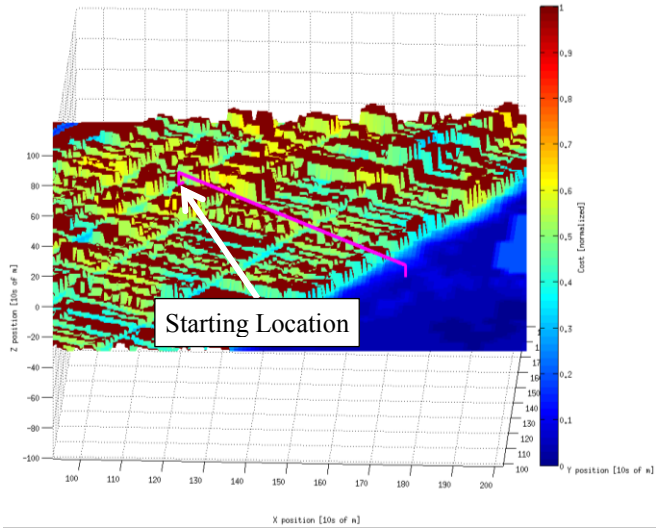

Fig. 17. High initial battery energy case with all environmental cost terms included. Because Central Park has a lower terrain cost than the Hudson River the quadrotor climbs above buildings and flies into Central Park to land.

5) Descending from High Altitude

In another test case, we initialized the vehicle at a high altitude in the vicinity of multiple goal states but starting over a very high cost location (the Easy 2 case series). For this case study and nonzero weights to all environmental cost terms, the planner first navigates to a zone with significantly reduced population cost then descends to land. This result is shown in Fig. 18.

6) Landing in an Urban Canyon Area

We examined a very low energy (5%) case in which the maximum goal cost is higher than usual to allow for the necessity to land in the city rather than Central Park or the Hudson River.[4]

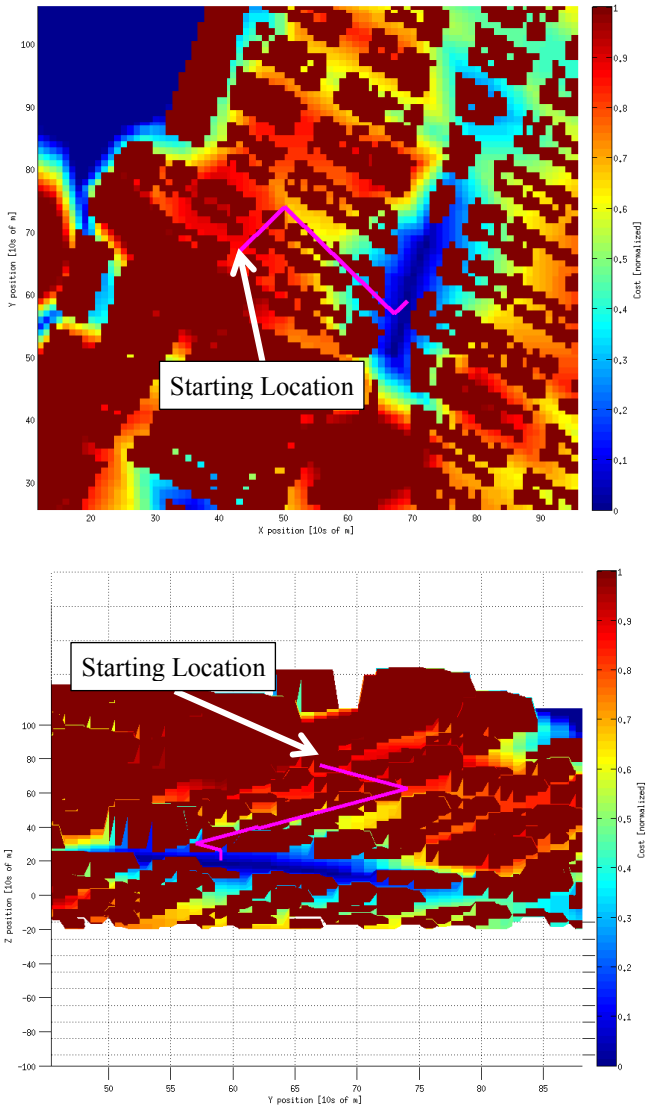

Fig. 18. Test case with nearby low-cost landing site availability. The solution path avoids highly populated areas to minimize the risk posed to persons overflown.

This would be the contingency path if the vehicle did not have sufficient energy to fly to a more desirable landing zone. As shown in Fig. 19, the path stays within the urban canyon while navigating to a reachable nearby lower-cost urban area for landing.

[4] Automatic relaxation of the goal cost constraint would be required in practice as the planner computes or determines that no landing site is possible given remaining energy and default goal cost constraints. This procedure is left to future work.

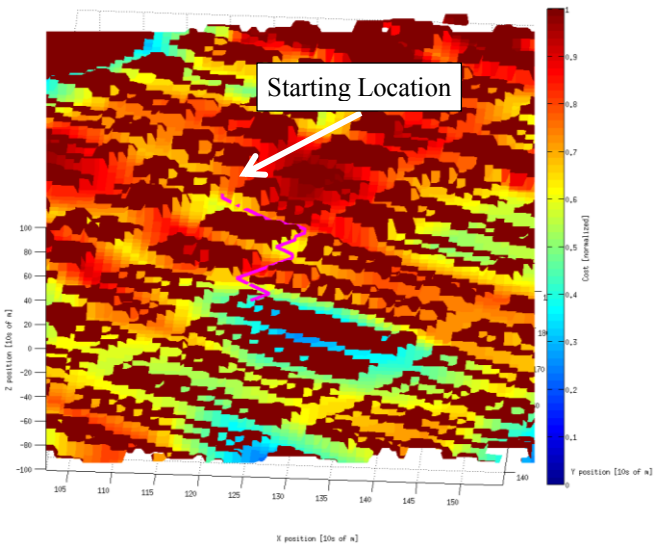

Fig. 19. Case with high maximum cost and low initial battery energy; the vehicle weaves through streets to a suitable landing zone.

VI. Conclusions and Future Work

This paper has presented an emergency landing planner applied to a small multicopter operating in an urban environment that must land in the near-term due to low energy reserves. Simulations illustrated tradeoffs between population, terrain, and structure clearance costs. Database preprocessing enabled expensive computations to performed offline enabling this algorithm to ultimately be implemented for online use by a small UAS.

A number of future improvements to our path planning algorithm will produce a more robust system. Improvement to the UAS model will improve simulation fidelity. Additionally, the planner must handle cases where no viable landing site is within range of the vehicle, more comprehensively trading off risks associated with a non-ideal landing location with the probability of not having enough energy to reach the ground before losing power.

To increase the processing speed of the search algorithm, more information could be precomputed then loaded onto the UAS. For example, an optimal policy over the entire cost map could be generated a priori without considering acceleration cost, significantly improving the A* heuristic function. Thought this would not reduce worst-case search complexity, it could greatly reduce average-case search times. Additional search information could be incorporated into the search. For example, locations of streets as well as expected traffic density information based on time-of-day and day-of-the-week could prevent the UAS from designating the middle of a busy intersection as an acceptable landing zone. Similarly, if the planner was given data classifying the roofs of buildings in the region by slope and other pertinent structural features, it might find that rooftops would actually serve as ideal emergency landing sites.

VII. Acknowledgements

This research was supported in part by NASA contract NNX11AO78A. Special thanks to Michigan Autonomous Aerial Vehicles (MAAV) team for sharing flight data from their quadrotor.

References

[1] A. Watts, V. Ambrosia and E. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use," *Remote Sensing,* vol. 4, no. 6, pp. 1671-1692, 2012.

[2] K. Hayhurst, J. Maddalon, P. Miner, M. DeWalt and F. McCormick, "Unmanned aircraft hazards and their implications for regulation," in *IEEE/AIAA 25th Digital Avionics Systems Conference*, 2006.

[3] E. M. Atkins, "Risk identification and management for safe UAS operation," in *IEEE Intl. Symp. on Systems and Control in Aeronautics and Astronautics (ISSCAA)*, 2010.

[4] K. Ro, J.-S. Oh and L. Dong, "Lessons learned: Application of small UAV for urban highway traffic monitoring," in *AIAA Aerospace Sciences Meeting*, 2007.

[5] I. Olson and E. Atkins, "Qualitative Failure Analyis for a Small Quadrotor Unmanned Aircraft System," in *Guidance, Navigation, and Control Conference*, Boston, MA, 2013.

[6] M. Warren, L. Mejias, X. Yang, B. Arain, F. Gonzalez and B. Upcroft, "Enabling aircraft emergency landing using active visual site detection," *Field and Service Robotics,* vol. 9th, pp. 9-11, 2013.

[7] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Englewood Cliffs, 2005.