UAV Path Planning using Aerially Obtained Point Clouds

Auburn REU on SMART UAVs 2022

Pugh, Alec
University of North Carolina at Chapel Hill
alecjp@unc.edu

Bower, Luke
The University of Alabama in Huntsville
lcb0035@uah.edu

Chapman, Richard
Auburn University
chapmro@auburn.edu

Biaz, Saad
Auburn University
biazsaa@auburn.edu

June 14, 2022

Abstract

With the growing use of unmanned aerial vehicles (**UAVs**) for commercial and military operations, path efficiency remains an utmost concern for battery and time preservation. This paper presents a method for three dimensional (**3D**) path planning using point clouds obtained from the USGS 3DEP (**United States Geological Survey 3D Elevation Program**) dataset via Open Topography. The path itself is obtained using the *A* algorithm*, with additional modifications implemented to account for path smoothing, UAV size, and energy consumption. We also introduce a collision avoidance method using the precomputed data to account for unforeseen obstacles not rendered within the point cloud. The method presented is designed specifically for point clouds obtained via LiDAR (**Light Detection and Ranging**) scans from aircraft, where cavities may be present underneath the surface layer. Simulations and physical testing using way point transmission show the validity of this method.

1 Introduction

The importance of research towards efficient and reliable path finding algorithms has steadily increased with the growing popularity and use of autonomous unmanned aerial vehicles (UAVs). There are many benefits around the development and use of UAV path finding, and each process developed mainly strives to satisfy each of the following requirements: time preservation, path optimality, and danger avoidance. The consequences of these requirements are the most apparent in military and commercial operations: provide a reduction in the cost spent towards training UAV pilots, a reduced amount of human manpower needed to pilot the UAVs, and a reduction of accidents due to human error or incorrect decision making. Technological developments have allowed humans to power cars with batteries for hundreds of miles at a time. While such batteries are viable for larger vehicles, versatile UAVs are compact and cannot carry large batteries or payloads due to their size and thrust output. As such, a typical commercially viable UAV has an average flight time of approximately 15-25 minutes [7]. With flight times this short, being able to accomplish more during a mission without recharging is critical to success. This correlates directly with the time spent travelling between points of interest and energy consumption of maneuvers, which path finding methods help reduce.

The USGS 3D Elevation Program is a developing program that aims to provide topographical data for the entire United States. Beginning production in 2016, 84% of the nation has topographical data that has been obtained from aircraft LiDAR scans. These LiDAR scans are displayed in the form of a point cloud, which is a set of 3D points (containing X, Y, and Z coordinates, among other types of data) in space relating to their physical GPS location. The data available through USGS 3DEP can be seen as undoubtedly instrumental in providing accurate path finding for UAV operations, especially for locations that may be inaccessible to humans or ordinary UAV flight operation due to dangerous conditions, FAA (**Federal Aviation Administration**) regulations, or various other impractical reasons. Furthermore, if this information is confined to a specific operating environment and is obtained prior to a mission launch, optimal paths can be found with less computational intensity and can be flown without prior mapping.

To this end, the main purpose of this paper hopes to leverage the vast amount of LiDAR point cloud data obtained through USGS 3DEP and show a concise method to provide path finding using this data. In addition, a secondary objective is to present a method of collision avoidance that also leverages the precomputed data, creating an all-inclusive program. This paper is outlined as follows: Section 2 displays the purpose of the research presented. Section 3 presents a literature review for the previous research done in this field. We outline our proposed method for UAV path planning using aerially obtained point clouds in Section 4. Real-word testing is performed and documented in Section **??**. Our results follow in Section **??** with a brief summary and outline for future work in Section **??**.

2 Problem Statement and Motivation

The advantages that UAVs withhold over manned aerial vehicles are apparent; however, there are still many limitations with the currently available technology. One main disadvantage comes from limited battery life that UAVs posses, so research into battery improvements for UAVs has been vast. While various methods for increasing battery life has been tested, such as unlimited endurance (laser-beam in-flight recharging, tethering, and swapping), along with alternative fuel sources (such as hydrogen, methanol, and hydrocarbons) [1], few of these methods or alternatives have become common practice for use in commercial and consumer UAVs as of today. This is often due to the limited weight tolerance that UAVs allow, which naturally prevents the trivial solution of equipping a UAV with additional batteries. Therefore, off-board optimization techniques can be viewed as a solution to help optimize the use of the limited battery life available while also preventing weight additions on UAVs with already limited payload tolerances.

As stated previously, obtaining point cloud scans of a given area can be difficult or impossible due to lack of proper equipment, state/governmental regulations, or safety concerns. By taking advantage of the datasets already available, this eliminates the need for expensive equipment and the inaccessibility of obtaining topographical information otherwise. However, the accuracy of path planning using point cloud data is proportional to the quality of a given point cloud scan. With USGS 3DEP datasets, LiDAR scan quality is standardized through QL (quality level) specifications which are detailed enough to be trusted for use for path finding operations. In addition, USGS has plans to continue to increase the overall scan quality overtime through dataset updates [23]. This shows that the accuracy of the methodology presented will increase linearly with the progression of future processes and technologies used to obtain these scans. Therefore, a streamlined 3D path planning approach using this data will be useful for not only

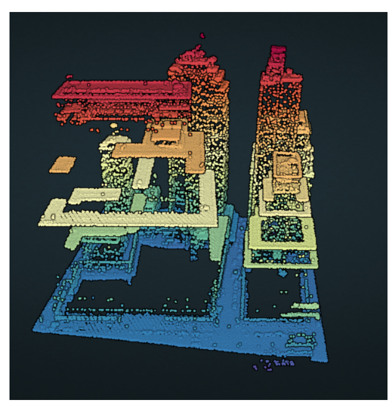

Figure 1: Example of point cloud with cavities present. (Potree)

present UAV operations, but also the future.

The primary objective of our research is to create a reliable and efficient 3D path that works seamlessly with 3DEP scans. Due to the nature of these scans being obtained through aircraft flying above an area, only the surface points of an environment may be scanned accurately. Thus, features that exist directly perpendicular to a surface (such as building faces underneath a roof, or steep cliffs) may be represented as empty space in a 3DEP scan, as shown in Figure 1. The process presented accounts for this problem. Our research uses a multi-rotor UAV for its quick movement and directional freedom. Due to the path being generated off board from the UAV, on-board equipment can be limited to a 2D LiDAR sensor, GPS, and transmitter. A secondary objective of this research is to have systems in place to provide collision avoidance procedures. By using the precomputed LiDAR data for collision avoidance, we once again do not require additional onboard equipment, which provides greater flexibility for deployment applications. The final objective is to incorporate API integration to create a seamless path generation process from input to output.

3 Literary Analysis

There has been many significant contributions to the field of UAV research over the years. To provide a top-down approach in topic discussion, this section will first provide a classification for the different types of mapping systems within UAV research. Then, we provide definitions and related work with UAV and LiDAR, path planning, and collision avoidance techniques with respect to the type of mapping that reflects this paper and process.

3.1 Mapping Classifications

In general, there are three main types of mappings used in autonomous UAV navigation and path planning: map-building, map-less, and map-based systems [15]. *Map-building* systems use a UAV to simultaneously build a map of the environment while flying within it. This is referred to as the "simultaneous localization and mapping" (**SLAM**) computational problem. Work in this area commonly uses onboard sensors (such as LiDAR) to perform both the mapping and localization. Algorithms used for SLAM operate with probabilistic quantities (a dimensionless set of values that build the environment), with two major implementations being the extended Kalman filter (**EKF**) SLAM and GraphSLAM [24]. The benefits of SLAM can be clearly seen for use in environments with unreliable GPS navigation [9], environment reconstruction [21], or as a means to perform autonomous navigation as a whole.

A *map-less* system seeks to perform navigation without using any maps. This is often accomplished using computer vision and reinforcement learning techniques [11], in which information about an environment is constantly being given to a UAV and decisions are made using that data. One method uses a computer vision algorithm to identify the features of obstacles in a process known as *feature tracking*, and the position of a UAV within its environment is calculated based on such features [18]. This method of path planning is great for unknown locations; however, due to the increasing amount of LiDAR data collected from USGS that covers the majority of the United States, there are less computer intensive methods of path planning.

Finally, the *map-based* system, which is the type of system that this paper is based upon, uses a predefined map layout and allows a UAV to navigate based on that information. One way obstacles are defined in a map-based system is through an obstacle grid, which is a 2D grid that stores information on whether a cell is an obstacle or not by giving either a value of 0 or 1, with one representing an obstacle and zero representing free space. Dryanovski *et al.* developed a method for producing multi-volume occupancy grids, which store three-dimensional information about about both free and occupied space of an environment and continuously corrects a map when given updated sensor information [5]. Another common method of mapping within a map-based system uses *octrees*. With octrees, a node of space is represented as a 3D cube, called a voxel. Through recursion, each voxel is further subdivided into eight children until a minimum size is obtained, which is predefined and also dictates the resolution of the octree. Because octrees are represented as a tree data structure, previous (lower) resolutions can be easily accessed at any time [10]. However, octrees by nature can only provide approximations of an environment, as voxels are represented as cubes. As such, an approximation of an object with surface curvature (i.e. a sphere or dome) will never be perfectly represented regardless of octree depth.

3.2 LiDAR and UAVs

LiDAR is a remote sensing method that uses pulsed lasers to emit light waves towards an environment. The time it takes for these pulses to bounce off of objects in the environment and return to the sensor is measured, and that distance is used to determine the height of the objects. LiDAR points are initially returned as spherical coordinates, but can be converted into cartesian using mathematical formulae, which are displayed below with radius ρ, polar angle θ, and azimuth φ.

$$
\begin{aligned}
x &= \rho \sin\theta \cos\varphi \\
y &= \rho \sin\theta \sin\varphi \\
z &= \rho \cos\theta
\end{aligned}
$$

Alternatively, an onboard Inertial Processing Unit (**IMU**), often equipped on aircraft and used to report the orientation of the vessel, can be used to automatically convert LiDAR data from spherical to cartesian coordinates.

The applications of LiDAR scanners often mesh well with the operational capabilities of UAVs, making their usage together well documented in research. Specifically, there have been many applications resulting from equipping a UAV with a LiDAR scanner for mapping, inspection, or measuring purposes. The general process for direct mapping and some inspection applications is to obtain a point cloud via a LiDAR-equipped UAV, and then perform a feature extracting technique to remove noise and enhance the data related to the problem at hand. G. E. Teng *et al.* used this process to inspect the features and security of a power line [22]. Measuring applications follow a similar process: develop post-processing algorithms that interpret the obtained LiDAR data for desired information. Chisholm *et al.* have recently used a LiDAR equipped UAV to estimate the diameter of trees below canopies [4], which shows a promising future for remote sensing using LiDAR in GPS-denied environments. It should be mentioned that most applications with LiDAR and UAVs involve obtaining a scan using the UAV, and then afterwards (or simultaneously in the case of SLAM) performing some processing on the data. This paper focuses on performing UAV flight operations on LiDAR scans obtained via another source (from aircraft).

3.3 UAV Path Planning

Path planning techniques all exist to solve the same problem: given an environment with obstacles and a start/goal node, how can a path be constructed between these two nodes while avoiding all obstacles? As described in [15], there are two main types of UAV path planning: global and local. *Global path planning* techniques find a path from the start to the end node within a global (static) map environment. *Local path planning* works to dynamically find subsets of a global path based on UAV state and updated stimuli. It can be seen how global and local path planning are related. A global path remains constant, but can be adjusted with intermediate local paths based on new information obtained from sensors (i.e. appearance of an obstacle, environment that differs from expected, etc.). There are many different algorithms that achieve these goals, with each belonging to a certain paradigm that describes its process at a categorical level [25]. This review will mostly cover research involving UAV pathfinding for node and sample-based algorithms with a focus on real-word environments.

Sampling-based algorithms, such as the Rapidly-exploring Random Tree (**RRT**) and RRT* algorithms operate by *sampling* points randomly in some $\mathbb{R}^n$ space. RRT then uses these points as seeds to grow trees from, and connects the grown trees together to form a path between a start and goal point. Many variations have been made to the base RRT algorithm for UAV path planning purposes, such as introducing a bidirectional variant [16] or reducing the search space by growing trees from the start and end points instead of just the initial point [27] to reduce computation time. 3D UAV navigation using sampling-based algorithms and point clouds has also been documented [28], however these algorithms may fail if cavities are present in the point cloud scans.

Node-based algorithms, such as the *A* algorithm* have been widely used for UAV path planning purposes as well. This is due to its ability to be easily modified to fit the subjects of specific papers [6]. It works by calculating the total cost $F(n)$ of each node using two components, a metric for node distance travelled so far $g(n)$ and a heuristic function $h(n)$. The heuristic function chosen can vary the resulting path greatly, and is itself a subject of many papers. By processing the nodes with the lowest cost first using a priority queue, the algorithm prevents unnecessary exploration of too costly nodes, which reduces the computation time. The formula to calculate a node's cost value is shown below.

$$F(n) = g(n) + h(n)$$

The A* algorithm has been shown to generate almost optimal paths for 3D UAV path planning, with shorter lengths and less computation time compared to the popular sampling-based algorithm, RRT/RRT* [26, 2].

3.4 Collision Avoidance

Collision avoidance is necessary for safe and fully-functional autonomous flight, and the different systems used to accomplish this has undoubtedly contributed to the vast amount of research that has been done in this area. The two key concepts that make up collision avoidance systems are sensing and detection, and maneuver approaches [20]. Sensing and detection, as can be implied from the name, constitute devices or methods that allow a UAV to position itself relative to other obstacles. One technique is optic flow, where motion is detected from brightness patterns of an image and used to calculate an optical flow field, consisting of vectors for each pixel [3, 17, 8]. Stereo-vision techniques use the triangulation of two sensors to obtain depth maps, which is used to perform an avoidance [19]. Laser sensor techniques use sensors such as LiDAR scanners to perform detection and avoidance, and can be seen in [14, 12].

Maneuver approaches are more varied, as the optimality of an approach would depend on the sensing method used and environment that the UAV is operating in. Many different techniques have been developed, and are covered by [20].

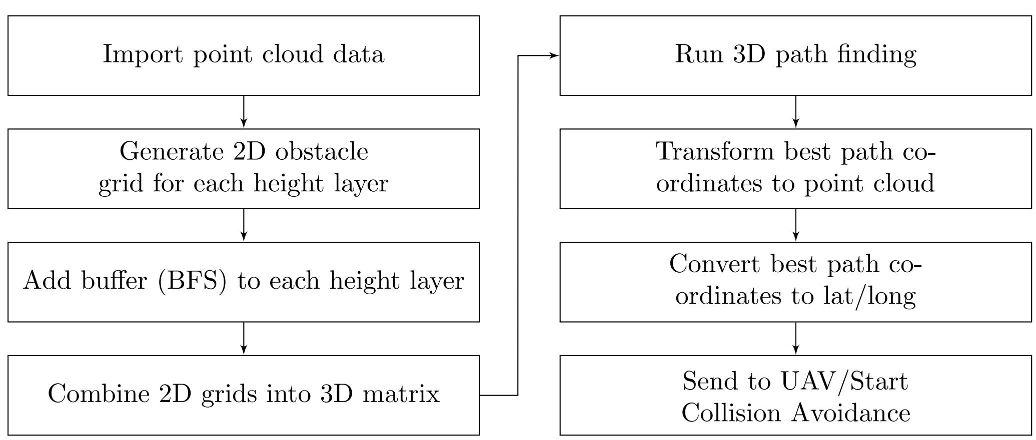

Figure 2: Overview of process from importing data to sending way points to UAV.

4 Approach

4.1 Software and Equipment Used

Software/Equipment	Reasoning
OpenTopography	Used to access all point cloud datasets for testing purposes. OpenTopography API was used to access some datasets remotely.
UTM	Python library used to convert WGS84 encoded coordinates to latitude and longitude to send to the UAV.
LASzip	Used initially to unzip .laz files to .las and ASCII formats.
Open3D	Python visualization library used due to its available functionality for point cloud operations.
ARDU Pilot	Used for communication from computer to UAV and for simulated UAV testing purposes.
Mission Planner	Simulation and transmission software that can control UAVs for realistic simulation testing.
MAVProxy	In-between software that allows for ARDU Pilot and Mission Planner to communicate and link over network IP addresses.
2D LiDAR Scanner	Equipped to the UAV for collision detection and avoidance testing.

4.2 Process

Our process for computing path planning and collision avoidance for aerially obtained point clouds is shown in Figure 2 and outlined in detail below. The process has been subdivided into multiple sections for better comprehension. Each part will discuss the methodology used to accomplish the goal outlined in the title of the section.

4.2.1 Matrix Generation

To generate the 3D matrix M that is used for path planning, we first must import the .las file that contains the desired point cloud. To do this, we use the *laspy* library in Python. Additionally, we use *laspy* commands to extract the maximum and minimum point cloud points from the data. The difference between the maximum and minimum Z point (Z_{max} and Z_{min}) defines the depth of the matrix. Similarly, the difference between the maximum and minimum Y (Y_{max} and Y_{min}) and X points (X_{max} and X_{min}) define the rows and columns, respectively.

$$M_x = X_{max} - X_{min}$$
$$M_y = Y_{max} - Y_{min}$$
$$M_z = Z_{max} - Z_{min}$$

An elevation grid is then created from the imported .las file. In this 2D grid, the X and Y coordinates represent the relative point cloud location, and the value within that cell represents the height of that section in meters (1 cubic meter per cell by default). This is done using code modified from [13], and contains an interpolation gradient to account for data loss from the point cloud data.

To convert the 2D grid into a 3D occupancy grid, we loop using height values that start from the minimum Z point to the maximum Z point in the point cloud, creating a temporary 2D grid and assigning a value of 1 or 0 depending on whether or not the cell is an obstacle or not, respectively. This is determined by subtracting the current height value from the height found in the elevation grid at the specified pixel. A threshold value is used to create a *vertical* buffer for object classification within the matrix. The algorithm is displayed as Algorithm 1. A *horizontal* buffer is created using an implementation of the Breadth First Search (**BFS**) algorithm. Obstacles groupings are identified and expanded outwards by a predefined amount of meters. The inclusion of the vertical and horizontal buffers were for three main reasons: to help prevent

Figure 3: Two cross-sectional 2D grids displaying different heights generated using Algorithm 1.

potential collisions due to noise within the point cloud, to account for the size of the UAV itself, and to provide a safe distance between the UAV and the obstacle for path finding and smoothing purposes.

Algorithm 1 create3DMatrix($height\_values$, $buffer\_size$, $threshold$)

$M \leftarrow matrix$
$height\_values \leftarrow H$
for $cur\_height = Z_{min}$ to Z_{max} **do**
  $G \leftarrow grid$
  for $x, y \in G$ **do**
    if $cur\_height - H[x, y] \leq threshold$ **then**
      $G[x, y] \leftarrow obstacle$
    else
      $G[x, y] \leftarrow free$
    end if
  end for
  $G \leftarrow bfs(G, buffer\_size)$
  $M \leftarrow append(G)$
end for
return M

4.2.2 Path Planning and Conversions

To develop the path planning portion of this project, we needed something easily modifiable that would work smoothly with the matrix generation technique discussed in the previous section. For these reasons, the A* path finding algorithm was selected and used as a base for further modification. To account for all directions in a cubic grid, our A* will evaluate 26 neighbors at every current node. This accounts for the four cardinal directions and diagonals on the same Z plane, as well as the same for one Z above and below, and straight up/straight down. The path obtained from A* consists of 3D points in space in relation to the matrix, with the origin at $(0, 0, 0)$. This path cannot be directly related to the point cloud, as the coordinates of the point cloud are encoded using some reference system (such as WGS84 Web Mercator), so it must first be transformed. Algorithm 2 presents the method for transforming the path coordinates to point cloud space using the best path and the maximum/minimum point values from the .las file.

With the best path now encoded in WGS84 to coincide the point cloud data, we can apply path smoothing using Bézier curves, the formula of which is presented below as a function of time t.

$$\mathbf{P}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$

Bézier curves produce smooth paths through n control points. This defines the degree of the curve (1 point for linear, 2 for quadratic, etc.). By using an n-order Bézier curve, we create a smoothed line using each

Algorithm 2 transformToPointCloud(*best_path*, *las_extremes*)

for $(x, y, z) \in best\_path$ **do**
  $x \leftarrow x + X_{min}$
  $y \leftarrow Y_{max} - y$
  $z \leftarrow Z_{min} + z$
end for
return *best_path*

point from the best path as a control point. This provides for the most accurate representation of the non-smoothed path, which is shown in Figure 4.

Because the focus of this project is for real-life operation, additional conversions must be made to the best path coordinates. Namely, the previously transformed path must be converted to latitude and longitude so that the UAV may receive the path using an onboard GPS. To do this, the Python library *utm* is used. The conversion function must be supplied with the UTM zone of the physical location containing the point cloud scan and whether it exists in the southern or northern hemisphere. The final path F is a 2D list that contains tuples with the latitude and longitude coordinates, and the unconverted height (Z value) for each point in the smoothed path. This path is now able to be sent to a UAV, and the accuracy of the conversions from point cloud to real world GPS way points can be shown in Figure 5.

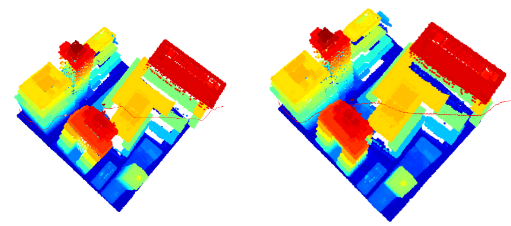

Figure 4: Comparison of non-smoothed and smoothed path.

4.2.3 Collision Avoidance

The collision avoidance procedure is currently simulated by measuring the distance between the UAV and an obstacle using euclidean distance, the formula of which is given below. A 2D LiDAR scanner will be used for real-world testing in Section **??**.

$$\mathbf{d}(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$$

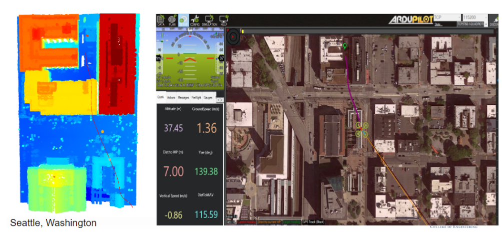

Figure 5: Simulated path in point cloud vs. satellite view using ARDU Pilot.

After the best path has been found using the methodology discussed in Section 4.2.2, the start of an avoidance path is calculated for each way point in the best path. This is done by evaluating the 3D obstacle matrix in a set of predefined directions by continuously extending a line until an obstacle space is met, or a threshold is reached. This is done for each direction and for each point, and the longest line is saved as the path with the most freedom and stored in an array. The idea behind this precomputation is to reduce the computation time during the event of collision avoidance. If an obstacle is met, the general flow of operation is: move UAV to precomputed avoidance point, recalculate A* from that avoidance point to a point on the best path that is beyond the obstacle, and then continue on the main path. Algorithm 3 shows how the avoidance path calculation is done.

After the avoidance paths have been generated, they are ran through the same transformations and conversions discussed in Section 4.2.2. Therefore, the avoidance path can be sent to the drone with no issues.

Algorithm 3 findAvoidances(*matrix*, *best_path*, *threshold*)

```
A ← all_avoidance
B ← best_avoidance
for p ∈ best_path do
    for d ∈ directions do
        while in_bounds(d, matrix) and threshold > 0 do
            p_x ← p_x + d_x
            p_y ← p_y + d_y
            p_z ← p_z + d_z
            threshold ← threshold − 1
        end while
        B ← max_dist(B, [p_x, p_y, p_z])
    end for
    A ← append(B)
end for
return A
```

4.2.4 API Integration

A goal of this research is to create a seamless process from the input of GPS coordinates to the UAV flying the generated path. Thus, we needed to utilize an API to allow for remote file downloading. OpenTopography has a limited API for developers, which allows for two GPS coordinates broken down into the maximum and minimum latitude and longitude to create a search polygon. This polygon is then filtered on the bases of metadata, output format, and if federal databases like USGS should be searched alongside databases from other origins (community submitted or OpenTopography). The API then searches with the filtered data to produce a .json file, which gives vital information about the databases that are inside the search area. By passing this information into the GeoAPI naming convention and searching the servers that host the .laz files, it can be downloaded remotely. The .laz file produced is an culmination of the point cloud points for the searched area using a particular Web Mercator conversion factor, followed by four digits. (in the format EPSG ####). The conversion factor associated with the point cloud is extracted from the .json file and allows for the point cloud to then be converted to GPS coordinates.

The z-coordinate is directly related to the mean height above sea level from the NAVD88 standard based on a tide station in Rimouski, Quebec. These conversion factors convert EPSG 3857 (WGS 84) to EPSG 4326 (latitude/ longitude) including the height of those points related to the z-coordinate. These conversions allow for us to set the starting point of the path generation to the input GPS coordinate saving time from additional input and allowing for the final path to be converted to GPS coordinates once the program has finished computing.

5 Future Work

In the second half of this REU, we hope to begin real-life testing using one of the UAVs available and sending path way points to the GPS transmitter. We will also work more on collision detection, improving the algorithms developed so far and testing them in simulations before trying them out in real life as well. We will experiment with additional variables to improve energy consumption and other factors for more UAV-specific optimization. Our paper will include a Testing section with data from various test runs in simulations and real-life, experimenting with different heuristics and potentially other path finding algorithms and comparing them based on time, distance, and other factors. We will also work more on the API integration aiming to have a fully functioning process from start to end. Finally, we will include a Results and Conclusion section detailing all of our findings and ways future research can be added to our proposed method.

References

[1] "A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects". In: *Applied Energy* 255 (2019), p. 113823. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2019.113823.

[2] João Braun et al. "A Comparison of A* and RRT* Algorithms with Dynamic and Real Time Constraint Scenarios for Mobile Robots". In: Jan. 2019, pp. 398–405. DOI: 10.5220/0008118803980405.

[3] Haiyang Chao, Yu Gu, and Marcello Napolitano. "A survey of optical flow techniques for UAV navigation applications". In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2013, pp. 710–716. DOI: 10.1109/ICUAS.2013.6564752.

[4] Ryan A. Chisholm et al. "UAV LiDAR for below-canopy forest surveys". In: *Journal of Unmanned Vehicle Systems* 01.01 (2013), pp. 61–68. DOI: 10.1139/juvs-2013-0017. eprint: https://doi.org/10.1139/juvs-2013-0017. URL: https://doi.org/10.1139/juvs-2013-0017.

[5] Ivan Dryanovski, William Morris, and Jizhong Xiao. "Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 1553–1559. DOI: 10.1109/IROS.2010.5652494.

[6] Daniel Foead et al. "A Systematic Literature Review of A* Pathfinding". In: *Procedia Computer Science* 179 (2021). 5th International Conference on Computer Science and Computational Intelligence 2020, pp. 507–514. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.01.034. URL: https://www.sciencedirect.com/science/article/pii/S1877050921000399.

[7] Boris Galkin, Jacek Kibilda, and Luiz A. DaSilva. "UAVs as Mobile Infrastructure: Addressing Battery Lifetime". In: *IEEE Communications Magazine* 57.6 (2019), pp. 132–137. DOI: 10.1109/MCOM.2019.1800545.

[8] Zdzislaw Gosiewski, Jakub Ciesluk, and Leszek Ambroziak. "Vision-based obstacle avoidance for unmanned aerial vehicles". In: *2011 4th International Congress on Image and Signal Processing*. Vol. 4. 2011, pp. 2020–2025. DOI: 10.1109/CISP.2011.6100621.

[9] Sebastian Hening et al. "3D LiDAR SLAM Integration with GPS/INS for UAVs in Urban GPS-Degraded Environments". In: *AIAA Information Systems-AIAA Infotech @ Aerospace*. DOI: 10.2514/6.2017-0448. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2017-0448. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2017-0448.

[10] Armin Hornung et al. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34 (2013), pp. 189–206.

[11] Sunggoo Jung and David Hyunchul Shim. *Mapless Navigation: Learning UAVs Motion forExploration of Unknown Environments*. 2021. DOI: 10.48550/ARXIV.2110.01747. URL: https://arxiv.org/abs/2110.01747.

[12] Cezary Kownacki. "A concept of laser scanner designed to realize 3D obstacle avoidance for a fixed-wing UAV". In: *Robotica* 34.2 (2016), pp. 243–257.

[13] Joel Lawhead. *Learning Geospatial Analysis with Python*. Birmingham, United Kingdom: Packt Publishing, 2019.

[14] Liang Lu et al. "Laser-based Collision Avoidance and Reactive Navigation using RRT and Signed Distance Field for Multirotor UAVs". In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 1209–1217. DOI: 10.1109/ICUAS.2019.8798124.

[15] Yuncheng Lu et al. "A survey on vision-based UAV navigation". In: *Geo-spatial Information Science* 21.1 (2018), pp. 21–32. DOI: 10.1080/10095020.2017.1420509. eprint: https://doi.org/10.1080/10095020.2017.1420509. URL: https://doi.org/10.1080/10095020.2017.1420509.

[16] Li Meng, Song Qing, and Zhao Qin Jun. "UAV path re-planning based on improved bidirectional RRT algorithm in dynamic environment". In: *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. 2017, pp. 658–661. DOI: 10.1109/ICCAR.2017.7942779.

[17] Paul Clark Merrell, Dah-Jye Lee, and Randal W Beard. "Obstacle avoidance for unmanned air vehicles using optical flow probability distributions". In: *Mobile Robots XVII*. Vol. 5609. International Society for Optics and Photonics. 2004, pp. 13–22.

[18] Ivan Fernando Mondragon et al. "Visual Model Feature Tracking For UAV Control". In: *2007 IEEE International Symposium on Intelligent Signal Processing*. 2007, pp. 1–6. DOI: 10.1109/WISP.2007.4447629.

[19] Jongho Park and Youdan Kim. "Stereo vision based collision avoidance of quadrotor UAV". In: *2012 12th International Conference on Control, Automation and Systems*. 2012, pp. 173–178.

[20] Hung Pham et al. "A survey on unmanned aerial vehicle collision avoidance systems". In: *CoRR* abs/1508.07723 (2015). arXiv: 1508.07723. URL: http://arxiv.org/abs/1508.07723.

[21] Zhexiong Shang and Zhigang Shen. "Real-time 3D Reconstruction on Construction Site using Visual SLAM and UAV". In: *CoRR* abs/1712.07122 (2017). arXiv: 1712.07122. URL: http://arxiv.org/abs/1712.07122.

[22] GE Teng et al. "Mini-UAV LiDAR for power line inspection". In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* (2017), pp. 297–300.

[23] *USGS 3DEP Topographic Data Quality Levels (QLs)*. URL: https://www.usgs.gov/3d-elevation-program/topographic-data-quality-levels-qls.

[24] Wikipedia. *Simultaneous localization and mapping — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Simultaneous%20localization%20and%20mapping&oldid=1092128572. [Online; accessed 08-June-2022]. 2022.

[25] Liang Yang et al. "A literature review of UAV 3D path planning". In: *Proceeding of the 11th World Congress on Intelligent Control and Automation*. 2014, pp. 2376–2381. DOI: 10.1109/WCICA.2014.7053093.

[26] Christian Zammit and Erik-Jan Van Kampen. "Comparison between A* and RRT algorithms for UAV path planning". In: *2018 AIAA guidance, navigation, and control conference*. 2018, p. 1846.

[27] Denggui Zhang, Yong Xu, and Xingting Yao. "An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect". In: *2018 37th Chinese Control Conference (CCC)*. 2018, pp. 4854–4858. DOI: 10.23919/ChiCC.2018.8483405.

[28] Zhaoliang Zheng, Thomas R. Bewley, and Falko Kuester. "Point Cloud-Based Target-Oriented 3D Path Planning for UAVs". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020, pp. 790–798. DOI: 10.1109/ICUAS48674.2020.9213894.