Landon Smith

Dr. Navid Sabbaghi

OPS 803-02

22March 2023

**Drone Obstacle Avoidance Database Creation:**

**Utilizing Bayesian Regression to Achieve Probabilistic Obstacle Determination**

While the advent of autonomous drone delivery services may seem inevitable when evaluating the pace at which funding is pouring into the development of drone technology, there are still major technological hurdles that need to be conquered to bring automated drone flight mainstream. Among these hurdles, ensuring that a drone can protect the payload it is carrying as well as its own integrity by effectively evading obstacles is the most crucial to solve. In an attempt to reduce the likelihood of drone aviation accidents, NASA’s Resilient Autonomy team worked alongside the DoD and FAA to create a drone obstacle avoidance system called the Expandable Variable Autonomy Architecture, or EVAA. Unfortunately, the occurrence of the Covid-19 pandemic led to the inability of the team to complete flight tests that utilized the technology and caused the project to be discontinued. For the sake of continuity, this project seeks to pick up where NASA’s Resilient Autonomy team left off and create a drone obstacle avoidance database that can be communally utilized by proprietary flight path planning algorithms to enhance aviation safety.

The structure of this paper will consist of five major subsections. The first, second, and third sub-sections will be the overarching project goal, the benefits of using Bayesian Analysis for drone obstacle avoidance detection, and methodology used to achieve the outcome of the project. Following these explanatory sub-sections, the fourth and fifth sub-sections, explain the technologies that enabled this project to be conducted and potential future work that can be done to improve upon the results of this project.

**Project Goal: Achieve a Probabilistic Determination of Obstacle Height Values**

Through implementing a Bayesian Regression Model on cross-validated, overlapping LIDAR point cloud data, we can construct a probabilistic determination of whether or not an obstacle exists in a specific location rather than a simplistic binary assumption. This can be done by first creating a “cell” for every potential geographical location an obstacle could be situated. The sizing of these cells would change based off of the desired accuracy of the prediction model. For example, we could allow the size of the cells to be relatively large in rural areas where obstacles are extremely sparse but constrict the size of the cells in urban areas where the risk of an obstacle collision is much higher due to the higher density of obstacles. We will let each cell be represented by a normal probability distribution, where the mean is the height of the obstacle, and the standard deviation of the distribution is our confidence about the height of the obstacle.

**Benefits of Bayesian Regression for Obstacles Identification**

The use of Bayesian Regression Analysis to generate this probability distribution allows path planning algorithms to utilize much more information than a binary representation of obstacles would enable. For example, instead of only being able to ask whether or not there is an obstacle present in a certain location at a height of 10 feet, we can ask what the probability is that there is an obstacle in that certain location at a height of ten feet. This probabilistic dimension of the data enables path planning algorithms to adjust their flight paths more precisely to the needs of the mission by evaluating the cumulative likelihood of success of each potential flight path. For example, if the drone operator wishes to adopt a conservative approach to obstacle avoidance, they will have a lower threshold for the cumulative probability of cell’s containing an obstacle. This could result in cells without obstacles in them being avoided, making the desired path a longer distance to traverse but safer. However, if a drone operator decides to utilize a risk-tolerant approach, they will have a higher threshold for the cumulative probability of cell’s containing an obstacle. This could cause some obstacles to be overlooked and potentially cause the drone to collide with obstacles on its flight path, but this risk would likely be offset by a decrease in flight duration. The appropriate cumulative probability threshold to go with ultimately depends on the characteristics of the mission.

**Outline of Project Methodology**

The workflow that was utilized to create a drone obstacle avoidance database which leverages probabilistic data is as follows. First, the x, y, and z values from geographically overlapping raw LIDAR datasets was extracted from the raw .ply files the LIDAR dataset are stored as. Next, each unique longitude & latitude combination was allowed to serve as a cell on a grid representing the landscape, and comparisons between the height values reported within each dataset were recorded in a tabular format. These cross-validated height values were then fed into a Bayesian Regression model in order to add the desired probabilistic dimension to the data. In the next step, the output of the Bayesian Regression model was used to determine what cells contain obstacles according to a user-defined drone flight height and acceptable probability threshold. Lastly, the resulting probabilistic obstacle data from the Bayesian Regression model was inserted into a database where the results can be queried by anyone seeking to use the data for their own proprietary flight path planning algorithms.

The point cloud data utilized in this project is sourced from overlapping LIDAR scans of a sample plot of land located in Emeryville, California with the following geographical bounding box:

minX, minY (-13614349.164060349, 4557303.475503025)

maxX, maxY (-13613408.779604943, 4558076.336715533)

The selected plot of land captures many distinct topographic features such as buildings, vegetation, as well as the San Francisco Bay. The three LIDAR datasets used for cross-validation were all obtained on the OpenTopography website, a website which facilitates access to “high-resolution, Earth science-oriented topography data and related tools and resources.” The dataset’s were titled as follows:

CA AlamedaCo 2 2021

USGS LPC CA NoCAL Wildfires B5b 2018

ARRA-CA SanFranCoast 2010

**Data Cleaning: Making Raw LIDAR Data Useable for Bayesian Regression**

In order to obtain the longitude, latitude, and height values for each dataset that was found to be geographically overlapping each other, a Python script was created to extract only the x, y, and z values of each row of the raw LIDAR data. The longitude and latitude values were rounded to the nearest integer to be repurposed as labels for cells of equal size on our grid. The height values were also rounded to the nearest integer in order to make cross-validation between the height values reported in each dataset more efficient.

A byproduct of this rounding process was that duplicate rows with the same longitude and latitude were created, displaying redundant information. These duplicates were dropped from the table to ensure the uniqueness of the longitude and latitude values being used to label each of the cells on our grid. However, dropping all of the duplicate rows where we had redundant x and y values didn’t remove all duplicates because certain rows contained different reported z values. In an incidence where the longitude and latitude of two particular points rounded up or down to the same combination, the z values rounding in different directions would cause the presence of two duplicate rows to remain even after dropping the duplicates. Due to the necessity to remove the remaining duplicate rows, the table was condensed by taking the average of each duplicate row with the same unique x and y value combination and imputing that value into the dataframe as the z value for that particular cell. The output of these data cleaning processes left us with a dataframe of fully unique x and y value combinations with their corresponding height readings that will be utilized as cells later on.

Once the aforementioned processes were completed for all three LIDAR dataset, the dataframes were then merged together. The number of rows in the resulting table was greater than the row count of any of the individual tables because certain tables had unique x and y combinations that were not reflected in the other datasets. The row count for the merged table was 452,788 rows. However, by computing our minimum and maximum x and y values, we know our total grid should be 749 meters by 614 meters, which would total 459,886 cells on the grid. This means we have 7,098 rows not represented that need to be to be present to ensure grid continuity for an obstacle avoidance use case. This gap represents about 1.54% of the total necessary grid size. To remedy the issue, the x and y value combinations that should have been present in our merged table but failed to be represented were subsequently added to the dataset. After this step was conducted an interpolation technique was used to replace the null values in the dataframe and create our finished dataset.

**Bayesian Analysis: Gaining a Probabilistic Dimension to the Data**

The first step toward implementing a Bayesian Regression model to add a probabilistic dimension to the data was to import the interpolated dataframe from Python into R. Within R, a Bayesian Regression Model was created using height\_2021 as the dependent variable, along with height\_2018 and height \_2010 as independent variables.

Before evaluating the models results, we must ensure that the models assumptions were met. In the following two depictions, the combination of the Rhat values all being 1 and the fuzzy graphs that we see when inspecting our b\_intercept, b\_height\_2018, b\_height\_2010, and sigma plots suggests that model successfully converged. Despite our usage of Bayesian Analysis being for the addition of a probabilistic dimension to the data and not for prediction purposes, we can see that the 2021 height values are much more similar to 2018’s height values than 2010’s.

Text

Description automatically generated with medium confidence

Diagram

Description automatically generated

After running our Bayesian Regression model as a prerequisite, we can set user defined thresholds and begin to derive value from our model. The obstacle height threshold defines any cell with a height of over the selected threshold as an obstacle for the drone. The obstacle height threshold used in this example is 10 meters. The acceptable probability threshold creates a threshold for the probability that there is an obstacle of the certain height situated in the cell. A value of 0.95 here would mean that if there is a 95% chance there is an obstacle 10 units tall, then we should count it as an obstacle. In this particular example, a value of 0.20, or 20% was selected.

Now that we have run our Bayesian Regression model, we can utilize the results to calculate the posterior probability of its height for each cell on the map. Once we have the posterior distribution of heights for each square available, we can calculate the probability of a cell’s height exceeding our defined obstacle\_height. This is the same as computing the likelihood of an obstacle being present the cell. Next, we must check if the probability of an obstacle being present in the cell is more than the acceptable probability that we defined earlier. If the height probability ends up being greater than the acceptable probably, it will be defined as an obstacle, denoted by 1. The inverse will occur if the opposite is true. For this example, this methodology returns 166,946 cells with obstacles out of the 459,886 total cells our grid contains.

**Visualization of Probabilistic Obstacle Detections**

The following graph displays a visualization of our expected height values for each cell on our grid. The legend indicates that the elevation of more lightly colored areas is higher than that of lower colored areas.

A screenshot of a computer

Description automatically generated with medium confidence

The following graph shows us a visualization of the 0 and 1 values that were computed based on our obstacle height threshold and our acceptable probability threshold. In this case, these thresholds were set to 10 meters and 20% respectively. A light blue color indicates that there is an obstacle in that area, while a dark blue color signifies the drone can fly over the area with a chance lower than 20% it will run into an obstacle at a flight altitude of 10 meters.

A picture containing text, monitor, electronics, display

Description automatically generated

As a proof of concept, adjusting the obstacle height threshold to 20 meters with an acceptable probability threshold of 20% greatly reduces the number of obstacles identified as observed below. At the typical flight height of a drone at 30 meters, there are virtually no obstacles, allowing a drone to fly over this plot of land unimpeded.

A screen shot of a computer

Description automatically generated with low confidence

**Communal PostgreSQL Database: Integration of Probabilistic Obstacle Data**

Now that we have probabilistic obstacle data resulting from our utilization of the Bayesian Regression model, we now need to construct a communal database in order to make this information accessible to people would like to use it for a drone obstacle avoidance use case. This PostgreSQL database was created on the Google Cloud platform and in order to make it compatible with the RPostgreSQL library utilized in RStudio, the cloud PostgreSQL database was set to version 9.6. The database consists of and consists of two tables called “Bounding Box” and “Bounding Box Data.” The logical and physical data models are displayed below:

**Logical Data Model**

Diagram

Description automatically generated

**Physical Data Model**

Diagram

Description automatically generated

In the “Bounding Box” table, the individual columns titled id, xmin, ymin, xmax, ymax, obstacle threshold, and probability\_threshold store the necessary information about each individual bounding box. All of the columns are highlighted in orange to signify that the combination of all of them will serve as the primary key for the table. This is because the combination of all of these values must always be unique to avoid duplicate Bounding Box data being stored within the database. The id column being set as a serial integer value in this table allows us to avoid vertical replication of a bounding box title that would have been stored a string otherwise. Now that we have reduced the amount of string vertical replication in the database, we will be able to query much faster than previously would have been possible. The following is the result of querying all of the rows that currently exist in the “Bounding Box” table:

Graphical user interface, text

Description automatically generated

In the “Bounding Box Data” table, the longitude and latitude serve as the primary keys because the combination of these two columns must be unique to signify each individual cell. The bounding\_box\_id column is colored purple to signify that it is a foreign key that references the id created in the “Bounding Box” table. The “Bounding Box Data” table also includes columns to record the height\_2021, height\_2018, and height\_2010, as well as the binary obstacle column. The following is the result of querying all of the rows that currently exist in the “Bounding Box Data” table while setting a limit at 5 rows:

Text

Description automatically generated

**Tech Stack**

Many different technologies were used to create the goal of a communal drone obstacle avoidance database which contains probabilistic determinations of height values. The data cleaning portion of the project was done in Python, with the help of the pandasql and pandas libraries. In order to conduct our Bayesian Regression, the output CSV file was exported from our Python script and read into RStudio. Many libraries were utilized within RStudio, such as brms for running our Bayesian Regression, ggplot2 and igraph to help create visualizations of our grid, and RPostegreSQL to connect to our SQL server and upload our results directly. Lastly, as the name of the prior RStudio library would imply, a PostgreSQL database was created on the Google Cloud platform in order to house the outputs of our Bayesian Regression.

Table

Description automatically generated

**Current Limitations & Potential Future Work**

Through the implementation of a Bayesian Regression Model on cross-validated, overlapping LIDAR point cloud data, this project has shown the benefits of utilizing a probabilistic approach in drone obstacle detection and avoidance. However, there are some limitations to the approach that was taken in this report as well. When the raw point cloud data was extracted from the .ply files to create a dataset containing the height values for each cell in the years 2021, 2018 and 2010 respectively, instances of null values were plentiful. To remedy this, an interpolation technique was used to impute values into the dataset based on neighboring values. While this removed the null values, imputing values risks corrupting the underlying data. Another limitation that presents itself is the scalability of the model itself. This model was run on a plot of land that was 749 by 614 meters, an extremely small-scale application of Bayesian Regression. In order to apply this model to the entire continental United States, we would need magnitudes more computing power. The concept of cross-validation using LIDAR heights from multiple years is also in jeopardy when applied to a scale as large as the continental United States. There simply may not be enough overlapping LIDAR data in certain remote parts of the country to use cross-validation in combination with a Bayesian Regression model for an obstacle detection use case.

Despite the current limitations, future work can still be done to move the project further towards our goal of creating a communal store of quality data able to be utilized for aviation safety. The standardization of longitude and latitude coordinate values would allow querying based on geographical data utilizing PostGIS to calculate distances between points, which would be critical information for drone flight path planning algorithms. Another potential route could be to automate the process used to obtain probabilistic obstacle determinations. In other words, instead of having to manually preform each step, we could create a data pipeline that would perform all of the necessary tasks automatically. A hypothetical example of this data pipeline could start with a user visiting a website where they would query based on a specific geographic bounding box, obstacle height threshold, and acceptable probability threshold. The website would then access OpenTopography’s API, request all of the overlapping datasets in based on the user’s bounding box input, run all of the associated scripts automatically, and finally upload the results to the cloud PostgreSQL database the user can then access.

Another possible approach that could potentially be added onto the methodology explored in this project is to create Bayesian priors for each individual cells probability distribution by creating a “confidence score” through a similar type of cross-validation. Despite the current limitations, the future work described can help move the project further towards our goal of creating a communal store of quality data able to be utilized for aviation safety and help this emerging technology reach maturation.