# Soaring Over Terrain

Danielle Bolthausen, Animesh Chaudhry, Trenton Jansen, Robert Rivas, San Tran

***Abstract*— Computational Mathematical systems like Matlab are used for a variety of applications. With the advancements of technology and a vast amount of data, computers can simulate environments. The purpose of the application described in this paper is to provide an extension to the Matlab mapping toolbox; A way to simulate and survey terrain from the perspective of a flying drone.**

## I. INTRODUCTION

### A. Our Application

The main objective of the application that we have created is a simulation of terrain survey from a Drone's perspective. This could potentially be used for real world applications such as topographical surveying, military surveillance, or space exploration. We will explore a real world example in the next subsection. We are trying to simulate a drone flying over terrain, so that we could know what a drone would be filming before we take it into the real world to test it.

The impacts of our study/analysis are that we would be able to check optimal terrain for what we want to accomplish or even use it for virtual reality locations. This could also be used to study other planets. A person would only need to know the latitude, longitude, and altitude data in order to generate terrain data. Once the data is generated it can be used to study the surface of the planet and where it would be a good idea to land a vehicle. They would want to know that the vehicle would not be damaged and that the area around location is safe for the vehicle to move around in. It could also be used to find a location that we would like to explore. With this information we would be able to select a location and have a clear idea of what the terrain at the location would look like.

We were able to come up with three models and translate all of them into an implementation of a "camera" in Matlab. Our models are not considered "advanced", but we were able create a basic implementation of a Drone Flightpath using linear approximations. The affect that these models/implementations have on each other will be discussed in the upcoming sections.

### B. Impact/Real World Analysis

Surveying and GIS are one of prime real world example of our application. Terrain survey from a Drone's perspective offers an enormous potential for surveyors and GIS professionals. Carrying out topographic surveys with a drone provides you with the same accuracy as measurements collected by traditional topographic surveys, but in a fraction of the time (1). This leads to a substantial cost and workload reduction for specialists working in this field (1). Specifically, sub applications of Surveying and GIS include Land surveying/cartography, land management and development, precise measurements, and Urban planning (1).

Survey drones are able to genrate high-resolution orthomosaics (aerial image of an area, composed of multiple photographs stitched together using photogrammetry, which has been scaled and geographically corrected for accuracy) and detailed 3D models of areas where low-quality, outdated or even no data, are available (1). Land management and planning require simplification of topographic surveys and drones greatly simplify that process. Specific examples here are allotment planning and design, site scouting, and final construction of roads, buildings and utilities. All of this mainly leads towards the advancement of development in increasingly dense and complex urban areas since these projects require intensive planning and therefore time-consuming and expensive data collection (1).

## II. MATHEMATICAL MODEL

Our mathematical model describes what it means for a drone to follow a flight path while observing a specific location or target during a passage of time. There are three separate modules that make up this model; manipulating the drone's camera, generating a path the drone will follow and how lighting affects the observed terrain over the passage of time.

The model was derived from researching methods for camera manipulation and utilizing Matlab's built-in toolbox to construct a class of functions that creates a simulated flyover of a drone above a 3D map.

The primary parameters of our camera manipulation model are longitude, latitude and altitude. The altitude was adjusted in order to position the camera above the virtual terrain to a degree that was optimal for viewing. Using these parameters as inputs, and applying a linear approximation for determining a prespecified path, our simulated drone is able to fly over a virtual terrain.

The main source of all of our parameters lie in the usgs24kdem file that our code will process. The usgs24kdem however contains 3 important pieces of data:

1) A matrix that contains all latitude values for a given area

2) A matrix that contains all longitude values for a given area
3) A matrix that contains all altitude values for a given area

It is important to note that the dimensions of all of these matrices are the same as defined by the usgs24kdem. This is due to the fact that the usgs24kdem stores their geographic data as a square-raster. If you were to observe the model of a terrain with the matlab lighting setting flat this square-raster format can be visually observed, in the fact that each square "chunk" of the terrain would be rendered with different light values, resulting in the terrain looking more like a mosaic, than a continuous visualization of lighting over terrain.

### A. Camera Manipulation

There are two components to manipulating the view of a surface in matlab:

- Camera Position
- Camera Target

We approached the topic of each of these separately, but with the same approach. We will use linear approximation to generate a linear function that will represent the position/-target, moving from one location to another. For the camera position, we are using a set of data points (coordinates) to represent the camera position throughout the environment. We generate a set of linear equations for each "drone" position we want the camera to be at. Using the Linear Approximation method described above we can seamlessly transition the camera from position, to position, over one path, or many paths. Camera target is also quite similar since we are using a set of data points (coordinates) to represent the camera target throughout the environment. We then, again, generate a set of linear equations for each target position we want to point the camera at.

### B. Drone Path Generation

We use linear approximation to generate a bunch of paths for the drone to fly along. The exact implementation and proof of this model is presented in our verifying values section. We have a starting point and a destination point and we move on that line in intervals. We can repeat that iteration over and over again.

### C. Lighting

The model for the lighting part of the code was able to measure the height of a light source relative to the XY-Plane of a surface. Using this data, and some arbitrary light values, we are able to adjust the color of the light in order to simulate lighting environments like sunrise and high noon.

## III. Solution Process

### A. Describe the proposed work

In order to manipulate the objects in Matlab to achieve a virtual reality or drone perspective, we needed to understand how the camera, lighting, and terrain data worked in Matlab. We first needed to figure out how to correctly load in the terrain data and project it onto a 3-D surface. To understand the camera, we had to figure out how to manipulate both the camera position and the camera target to get a good view of the terrain we wanted to use. Along with understanding how the camera worked, we wanted to create paths for the camera to travel along and separate paths for the camera target. Afterwards, we had to understand and manipulate the lighting in Matlab to simulate that of Earth, since we were using Earth for our examples. Finally, we needed to put all of these ideas together to create visualizations of what it would look like if we were to send a drone to a specific location and film a video. Along with that, the code we created would be able to be applied to any terrain data, so that it could be more versatile.

### B. Assumptions before development

When we were first developing this project we assumed that the camera system in matlab worked on a position, pitch, yaw, and roll system. In that the camera control basically took 6 different parameters when determining the orientation of the camera:

1) X position in the space
2) Y position in the space
3) Z position in the space
4) Degree of pitch
5) Degree of yaw
6) Degree of roll

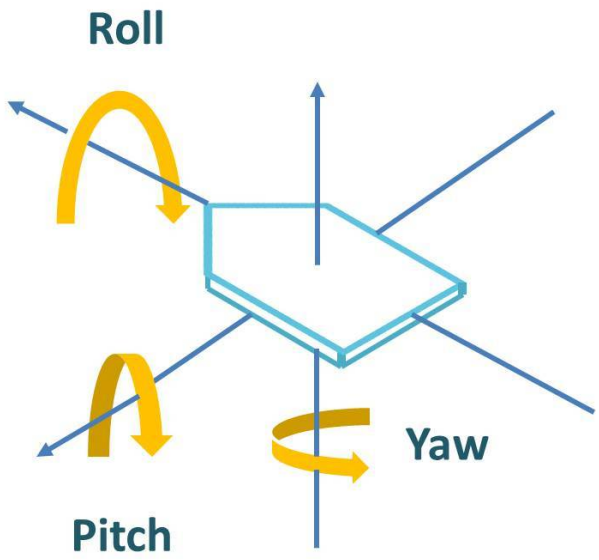


Fig. 1: Visualisation of camera control assumptions

In addition to the assumptions about the camera, when we were developing the terrain loading module we thought that geotiff files were the ideal files to use. We thought that the geotiff files had the Altitude Data corresponding to a latitude and longitude value. That way each data point has an altitude and can accurately represent a terrain in a "virtual" sense.

*C. Address the software/code used in the simulation. - How to use the code we wrote.*

Our code is made up of many functions, each of which can be run either simultaneously or alone by calling each function the user wishes to use. The objective of the code is not to observe or verify any particular model, but rather create a toolset in which surveyors, planners, or any interested party can observe any location as if they were there themselves; a la virtual reality. Firstly, one of the main functions of the suite we made was to streamline and simplify the display of terrain data in matlab. Typically, one would need to initialize a space, load in the usgs dem file, then find an orthographic image of the area stored in the usgs24kdem file. However, we streamlined this process by analyzing and processing the data from the dem file, obtaining the relevant information, and using that information to make requests to matlab's built in Web Map Service(Service).

As discussed earlier, the latitudes and longitudes of any given area described by a DEM file are located in two separate matrices. If you are able to find the minimum and maximum latitude and longitude you can initialize the space easily. So using matlab utilities we can implicitly find the bounds of the area that the DEM describes, and initialize a space on which we can start displaying the surface of the terrain. The special property of this space is that the axis are labeled with the corresponding latitude and longitude bound. However, this is really just a mapping of those latitude and longitude values onto a Cartesian space. At the moment there is no matlab documentation that shows the model/functions that map the latitudes to the Cartesian plane.

From there we need to somehow produce an image of the area from a satellite so we can "drape" the image over the terrain data after the altitudes are plotted in the corresponding locations. Using the Latitude and Longitude min/max we just found we can make a request to the matlab Web Map Service (WMS) and perform a query for images at that specific location at those specific bound. Then we can take the image and finally start displaying the surface of the terrain.

Using the vectors of geographic data, the initialized terrain space, and the image, the terrain constructor will then call the geoshow function to map the geographic data to the initialized space. Again using some abstracted model we can't discern, the geoshow function will map the geographic latitude data to some Cartesian point in the space. Once that has finished it will then take the satellite image, and partition the image into small squares. Then using those small squares geoshow can orthographically transform the square to drape over each square-raster in the terrain. Finalizing the terrain so that it produces a virtual environment of the geographical location was described in the usgs24kdem file.

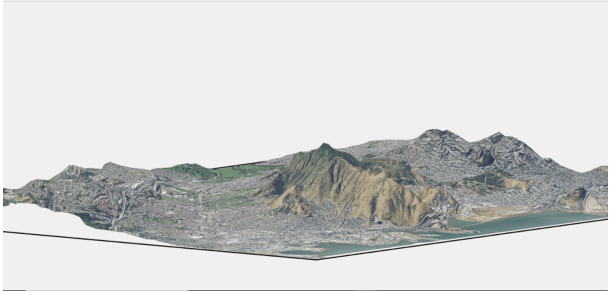

Fig. 2: San Francisco Bay Area

The Line function can be called in its current state to simply simulate a drone flying in a diagonal line while looking straight down at the ground below it. The user could easily change the flight path or the camera target in this function if they would like to. This function uses Cartesian coordinates to give a location to the camera position and the camera target, so the user would be able to set up a simple flight path or target if they want to alter it.

The LookCircle function can be called to have the camera rotate in a circle. This function has a set camera position, which can be altered by the user if they wish to edit the code of the function, but in its current state it is in the center of any terrain data that is used. The main part of this function is that the camera target is constantly changing. It is set up to go from corner to corner of the terrain data in a clockwise direction, giving the view of the terrain as if the camera was being rotated so that you can see the whole terrain.

The fly_to and change_target functions work very similarly. What these functions do is simulate a smooth transition of the "Drone's" travel, whether that be position to position, or target to target. By generating a linear equation to represent the transition between two positions. What it will do is that it will perform the linear approximation on two separate points: The camera's current position/target and the point where the user/programmer want the camera to point to in the future. Based off of these two points this function will generate a line and incrementally travel the line to simulate flight/camera movement

The Circle function can be called to have the camera rotate in a circle while having the target being fixed to one part of the terrain. In this function, the camera position is constantly changing to have the camera move in a circle, while the camera target is fixed. This function can be altered to change both the position and the target, however, in its current state it is set up to look at the center of the terrain data and to have the camera circle around that point using cartesian coordinates that were found by using the longitude and latitude data.

The sunsetlight function can be called to change the color of the light based on the degree of elevation of the light going from black when the elevation degree is negative and to the color of sunset when elevation degree is zero slowly morphing into the color of the normal sun when above a certain defined angle. This function is to be called right after a function which changes the position of the light. In this project the function we used to change the position of the light was lightangle. This function lightangle uses for input the light variable defined with the creation of the map, the azimuth degree, and the elevation degree. This elevation degree constant if you desire the sun to not change, put inside a loop with a pause in it to change the degree of the sun, or include inside the camera function loop to move both the camera and the light at once.

### *D. Verify your claims numerically*

*1) Linear Approximation:* The way the the "path" for the drone for a drone is generated between two points $X_1$ and $X_2$ is represented by a linear function:

So let us say that the initial position of a "drone" object is at a point in space $X_1$. Lets also assume that the drone objects target position is located at the point $X_2$

That means that simply the vector the represents the whole path from $X_1$ to $X_2$ is simply

$$r = X_1 - X_2 \tag{1}$$

And, since the initial position of the camera is $X_1$ so we know that the initial position of the camera exists at the point $X_1$ when $t = 0$ which we can define as $r_0$

From there we can construct our linear model $r$ a vector pointing from $X_1$ to $X_2$

$r_0$ the initial position of the camera $v(t)$ The position vector at the given period $t$

$$v(t) = r * t + r_0, 0 <= t <= 1 \tag{2}$$

It should be noted that now that we have the generalized formula/model for the linear approximation between two points can be used for not only the camera position, but also the camera target. As both matlab functions reference positions in the space. So this model can be used to simulate a smooth transition between positions and between targets.

*2) Lighting:* The model for the lighting part of the code was able to measure the height of a light source relative to the XY-Plane of a surface. lets define:

1) $P_x$ the x position of a light location
2) $P_y$ the y position of a light location
3) $P_z$ the z position of a light location

So for any light object we can have the corresponding positions. From:

$$h = \sqrt{p(1)^2 + p(2)^2}; \tag{3}$$

H being the projection of the position-vector P onto the XY-plane. Then from there we can get the angle between the vector H and the $P_z$ allowing us to get a elevation. From:

$$elevation = |arctan(Z/H)| \tag{4}$$

to get the angle of a vector from the XY-plate to the Sun. Thus allowing for a light color module to scale/manipulate the color of the light source based off of the $elevation$ value to simulate the sun at different times of the day based off the angle.

## IV. Future Plans

For our future goals we wanted to think of ways that our model could be improved and refined. One way is to account for collision avoidance. The objective would be to generate a flight path for our drone while also accounting for any obstructions or obstacles or no-fly zones that may be in the way of the drone. According to a Journal of Intelligent  Robotic Systems published by Egidio D'Amato there are numerous methods to accomplish this such as Graph-search method, NonLinear Programming method and Potential Fields methods. The objective of these methods are to optimize the data in order to find the most efficient pathing for a drone to follow. For example, the Graph-search method approximates a graph of the given space and each node of the graph represents a set number of positions and each edge of the nodes represents a transition. This data is then analyzed to determine the optimum path. A similar and more basic example would be the Travelling Salesman approach, which finds the best path with minimum exertion. Such a method would be an ideal addition to our model as it could help refine the sort of pathing that would be needed in order to obtain the most data from the terrain with the least amount of energy being expunged. (2)

(Energy Efficiency): Energy efficiency is another idea that is vital to proper method derivation. A lot of things are to be considered such as the functionality and capabilities of a given drone or UAV, as well as the most efficient way to search or cover an area. According to a Research Article by German Gramajo and Praveen Shankar about path planning for search and coverage some common methods of surveying an area is to create a serpentine pattern as well as creating a spiral that moves inward. These are both repetitive patterns but they ensure the most amount of coverage while also exerting the least amount of energy to accomplish the mission. This sort of planned pathing would be an appropriate goal to aim towards in improving our model. (8)

Our model's primary function is to observe terrain while flying over an elevated position. While researching various real-world applications that could be explored using our model we came across the concept of space exploration. The idea is to use our model in order to observe a completely different planet than our Earth. Using data gathered from NASA missions, we could form a virtual recreation of the surface of a planet and use our model to soar over the terrain and observe specific details. According

to an article by Eleanor Imster from Human World back in 2017, NASA posted a video showcasing a flyover of the surface of Pluto as well as its largest moon, Charon. These videos were derived from the image data that was gathered by the New Horizons spacecraft. This data was compiled into a digital elevation model and thus a virtual simulation became possible. The video in question simply shows the surface of Pluto while the camera travels laterally at an angle and the viewer is able to observe various craters and rock formations located on the surface of Pluto. With our model, we could take this idea a step further and truly explore the surface by picking and choosing our camera's position relative to the surface of the planet and choosing in what direction we would like our camera to look. (6)

The correlation between NASA's flyover video and our model's functionality is clear. While NASA's demonstration is vastly more sophisticated, our model is still very much capable of a similar task given the same set of data. A future goal of ours would be using the model we created and obtaining a data file containing accurate image and elevation data of another planet and creating a virtual map similar to NASA's that would allow us to soar over and survey points of interest. According to an article from nature.com NASA plans on a dual quad-copter drone to explore the surface of one of Saturn's moons. It's objective is to gather atmospheric data as well as terrain data in order to verify the existence of organic material on the surface. If NASA were to offer up this data for public access we could potentially use our same model to observe the same kind of information. (6)

## V. Conclusion

Our main objective was soaring over a terrain with a drone/camera. We used three main models. The first was a model for camera view point using camera manipulation. Next we came up with a linear approximation model to model the path of a drone. Lastly, we modeled the Lighting effects on terrain over passage of time. Our findings throughout this paper show how these models work together and affect each other. Here is the contribution section:

1) Robbert: Contributed by Researching many of the general and future applications of the model, and by suggesting/finding some of the more complex models that we didnt have time to implement. Also had a hand with Animesh with creating the Linear Approximation model, and implementing it in matlab
2) San: Contributed by generalizing the terrain loading functions into an object that can be used. Also helped implement the algorithms that help simulate flight over a path. As well as creating the queries to grab the orthoimages from the matlab Web Map Service.
3) Danielle: Contributed by creating the LookCircle, Circle, and Line functions which helped us greatly understand the camera functions for the development of the future linear approximation approach to the camera.
4) Animesh: Contributed by implementing the linear approximation generation algorithm that allows for a generalized approach to the drone flight path and drone target tracking. Also contributed to a large amount of the documentation, slides and essay.
5) Trenton: Contributed by implementing the lighting function that determines the height of the light source relative to the XY-plane on which the surface lies on. Also contributed by setting up a set of functions that can change the light based off of the height of the light object. Simulating Sunset.

## References


[1] https://wingtra.com/drone-mapping-applications/surveying-gis/.
[2] D'Amato, E., Mattei, M. & Notaro, I. J Intell Robot Syst (2019) 93: 193. https://doi.org/10.1007/s10846-018-0861-1
[3] https://en.wikipedia.org/wiki/Hour_angle/media/File:HourAngle_Observer_en.png
[4] https://edwilliams.org/sunrise_sunset_algorithm.htm
[5] https://en.wikipedia.org/wiki/Sunrise_equation
[6] https://earthsky.org/space/pluto-charon-new-horizon-flyover-video
[7] https://www.nature.com/articles/d41586-019-02027-3
[8] German Gramajo and Praveen Shankar, "An Efficient Energy Constraint Based UAV Path Planning for Search and Coverage," International Journal of Aerospace Engineering, vol. 2017, Article ID 8085623, 13 pages, 2017
[9] http://stars.astro.illinois.edu/celsph.html
[10] https://www.britannica.com/science/anomaly-astronomyref105659]