Table of Contents

[1. Introduction 2](#_Toc498875714)

[2. Assumption and Justification 3](#_Toc498875715)

[3. Part I: Mathematical Formulation 5](#_Toc498875716)

[3.1 Design of Static Images and Displays 5](#_Toc498875717)

[3.1.1 Ferris Wheel 5](#_Toc498875718)

[3.1.2 Dragon 5](#_Toc498875719)

[3.2 Arrangement of drones in the Launch Area 7](#_Toc498875720)

[3.3 Takeoff Flight Paths and Animations 8](#_Toc498875721)

[3.4 Our Own Design: the Interstellar System 10](#_Toc498875722)

[4. Part II: Practicality of the Light Show 13](#_Toc498875723)

[4.1 List of parameters and symbols used 13](#_Toc498875724)

[4.2 Cost Analysis 14](#_Toc498875725)

[4.3 Benefit Analysis 14](#_Toc498875726)

[4.4 Cost-Benefit Analysis 17](#_Toc498875727)

[5. Strength of Model 18](#_Toc498875728)

[6. Weakness of Model 19](#_Toc498875729)

[7. Conclusion 20](#_Toc498875730)

[8. Bibliography 21](#_Toc498875731)

[9. Appendix 22](#_Toc498875732)

[A. Programme for Recursive Zonal Equal Area Partition (Matlab) 22](#_Toc498875733)

[a. Partition 22](#_Toc498875734)

[b. Illustration 28](#_Toc498875735)

[c. Tracing the coordinates from the image and calculate the similarity 31](#_Toc498875736)

[d. Simulation of Difference Equation 39](#_Toc498875737)

[e. Generate the animation during take off procedure 41](#_Toc498875738)

[f. Transfer the polar coordinates to Cartesian coordinates for the Ferris Wheel 42](#_Toc498875739)

[g. Program used to design the trail of the take-off stage 44](#_Toc498875740)

[B. Table of coordinates of drones deployed 48](#_Toc498875741)

[a. Ferris Wheel 48](#_Toc498875742)

[b. Dragon 56](#_Toc498875743)

[c. Interstellar System 64](#_Toc498875744)

[i. Star (unit radius is used) 64](#_Toc498875745)

[ii. Planet (unit radius is used) 75](#_Toc498875746)

[C. Table of used in the test of Navigation System 78](#_Toc498875747)

# 1. Introduction

In light of intensification of scientific research and development of technology, many firms across the world have installed advanced technology tools in performance or major displays in their celebration. One of the novel ideas of using drones, one kind of unmanned aerial vehicles commonly used in military field, as alternatives of fireworks in a celebration event has now gained great popularity due to their reusability and environmentally friendly nature.

In order to plan a successful outdoor aerial light show by employing such drones during an annual festival, the organising committee will need to determine the feasibility, manageability and practicality of hosting such event. Specifically, we need to tackle the issue from two aspects - the programming of the flow of aerial light show and the cost and benefit of holding the aerial light show.

Image displayed and locations and flights paths of all drones during the display should be designed to prevent crash of drones.

Consideration of the decision to hold aerial light show is based on the cost and benefit analysis, that considers these factors: audience density in the projected area, projection area, number of drones, angle of elevation of the display from the projection area, cost of renting drones, cost of renting launch area, cost of regulating air space, cost of hiring manpower as operators and monitors and return rate. These factors are all important in determining whether it is worthwhile to hold an aerial light show. What should an organiser consider in making the decision that whether hold this event or not? Under what circumstances should the organiser hold this event?

The aim of our planning is to maximise the number of audience engaged, which in turn maximises the monetary revenues to be reaped from potential advertising, while at the same time minimise all resources employed. Conventionally, an event will have a limited budget. Since using more drones will be more costly, our model will be more likely to be accepted by the operators if we can keep the budget as low as possible while achieving the same effects. These resources include capital resources such as drones, land resources such as the required launch area for drones to take off, required air space for the drones to operate in, and labour resources such as operators employed to monitor the light show.

# 

# 2. Assumption and Justification

***2.1.*** *There are no obstacles such as birds and unusual weather elements that will affect the drones’ flight paths in the air space during the event.*

Justification: Since this is an aerial light show, the venue is most likely to be an open area without any hindrance. We do not consider the accidents where any aerial object, such as birds or other drones for civil use disrupt the event.

***2.2*** *There is only one launch area and all drones take off from the same launch area.*

Justification: It is more manageable for operators to place the drones prior to the event. Moreover, only one operator is required to monitor all drones during the takeoff process, reducing manpower. When all drones take off from the same area and ascend to the air space directly above the launch area, the flight paths are relatively shorter compared to ascending from separated launch areas further away from the air space, making it less draining for drones’ batteries.

***2.3*** *The battery provides constant power output throughout the aerial light show, and the battery life is able to last throughout the aerial light show.*

Justification: Intel’s Shooting Star drone used in the light show can stay airborne for up to 20 min. Since there are only 3 display of patterns in our light show, it is likely that the duration of will not exceed 20 min. Hence, battery life will not be a significant factor that determines the maximum duration of light show. Only factors such as take-off time, display time and transition time need to be taken into account to determine duration of light show.

***2.4*** *All formations of drones will be displayed at an altitude range of between 150m and 200m above the ground.*

Justification: Intel’s Shooting Star drone used in the light show can fly to a maximum of an altitude of 1.5km. However, the prime view angle for a clear and complete view of the display can be achieved at around 200m. Moreover, most skyscrapers in the cities have a height of less than 100m. Hence, by operating the drones at an altitude range between 150m and 200m, the audience are able to obtain the best visual effect without obstruction. Consistent height for all 3 displays also allows us to simplify our model by ensuring consistent duration of ascending and descending.

***2.5*** *There are private firms willing to advertise during the light show at this event, resulting in monetary revenues to be gained by the organising committee.*

Justification: It is usually difficult to quantify the social and private benefits of hosting such celebration events since benefits are usually in the form of increased social welfare and improved societal vibrancy. By introducing the idea of advertisement to be gained by the organisers, benefits can be comparable with the monetary costs. We can hence create a cost-benefit model of hosting this event so as to determine whether or not it is practical and worthwhile to host the light show.

***2.6*** *Advertising in the light show will generate the same amount monetary revenue across all audience for the organisers.*

Justification: This is so that we will be able to quantify the amount of benefits from hosting such a light show based on the total number of audience in the projection area of the drone display.

***2.7*** *Horizontal 2D image is the most suitable orientation of display for the audience to enjoy.*

Justification: We do not know the specific venue for the aerial light show. Thus, we do not know the locations of target audience cannot determine whether the image should be tilted towards which side. By setting it to be horizontal, audience from all direction is able to view the image.

***2.8*** *Organiser has limited budget and need to strike a balance between the visual effect of the display and expenditure.*

Justification: In real world, all events have a limited budget, and have an expected desired effect. To make a rational decision, decision maker needs to consider both of these factors.

# 

# 

# 3. Part I: Mathematical Formulation

# 

## 3.1 Design of Static Images and Displays

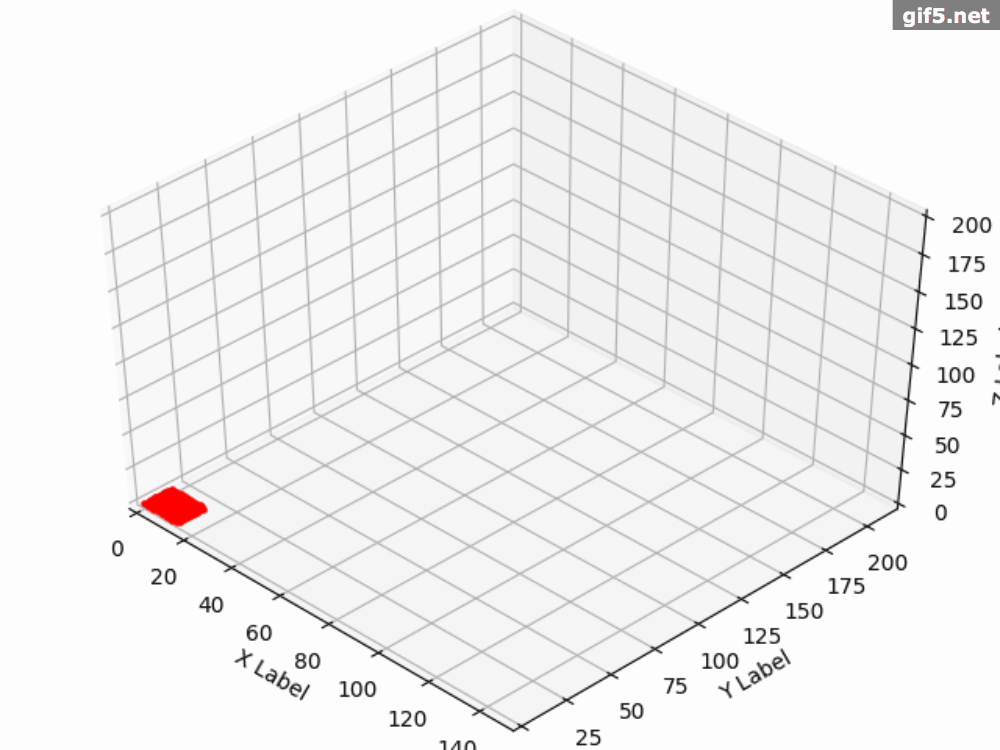
In order to determine the number of drones and locations of the drones, we must first design the static images the displays. To ensure the visual fidelity of the simulation of the static images in the air and a prime view of the entire display using a minimum number of drones, we have planned drones’ position in two different ways.

### 3.1.1 Ferris Wheel

In the **Ferris Wheel** display**,** drones assembling the frame of pattern are placed equidistant to one another due to its nature being a relatively simple geometrical shape. Hence, the visual effect will only be compromised to a small extent using this method of equal distribution of drones.

The image of ferris wheel is divided into three components: the main outer circle O with radius 75m, the 10 axes, P, each with length 150m, and the isosceles triangle, Q, with side length 100m, 100m, 50m.

A total of 320 drones are used in the display of this image. 150 drones are distributed on O with equal arc length between two consecutive drones. 100 drones are assigned to P with constant distance 15m between two drones on the same axis. 70 drones are location on the isosceles triangle with m between two drones on the same side. Our data are attached in Appendix B.a.



*Figure 1: Launching of drones for the Ferris Wheel display*

### 

### 3.1.2 Dragon

The second display, the 2D pattern of a dragon, is relatively more complex and is difficult to be differentiated, and the third display of the planet-star system which is 3D in nature.

Firstly, the points are sorted in order by tracing the pixels at the sides of the pattern. This points forms an set , where Rx, Ry denote the coordinate of the pixel in the image, and Rn denotes the order of the pixels that is been traced.

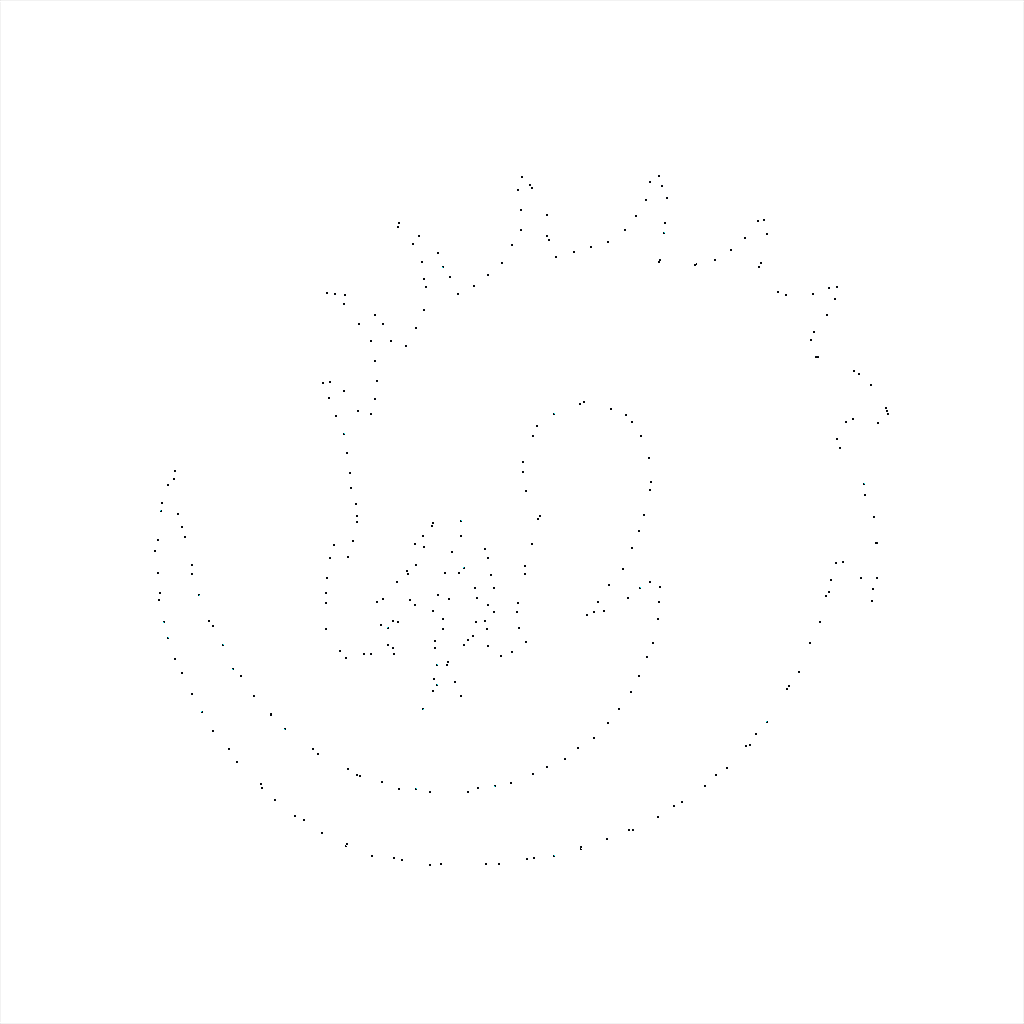
Then, the size of the set is reduced from to . This is achieved via a sampling process which follows the equation below, and a new set R^\prime will be formed.

Let  be

The points in will be connected using straight line afterwards. The pixels of the new image generated will be compared with the original one pixel by pixel. The correctness will be calculated as similarity.

Where matches are the number of points which are, both drawn in the original image and the generated one. Only the points that is drawn in at least one of the images are counted.

The similarity of our generated graph is 0.416 (3s.f.)

* *

*Figure2: Original static image of the dragon Figure3: Display of the dragon using drones*

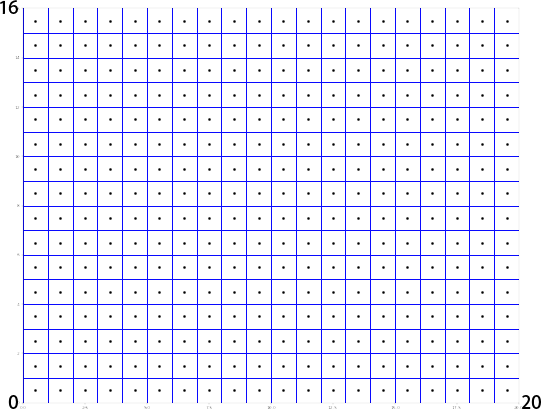
The coordinates of the drones used is shown in Appendix B.b.

## 

## 

## 3.2 Arrangement of drones in the Launch Area

In our model, all 320 drones are placed in a designated launch area prior to the light show. They are distributed equally, as represented by different coordinates in the Cartesian coordinate system below, such that they cover the entire launch area in an tessellation pattern.



*Figure 4: The launch area of the drones. Each dot in the 16\*20 grid represents a drone.*

## 

## 

## 3.3 Takeoff Flight Paths and Animations

To prevent clashing of drones due to unforeseen events during the flight, the relative velocity of each of the drones to its four closest drones will be monitored.

As individual drones ascend in different paths, their coordinates are updated every 1 second. The distance between two drones at time can thus be calculated and represented by , as shown in the ….

The expected change in relative velocity is thus

Once there is a change in relative velocity that is larger or smaller in magnitude than the expected value, , at time , of three or more nearby drones, the drone will change its direction to the vector sum of the changes in relative velocities, by applying an acceleration

This process is modelled by difference equations to prove that the drones will not clash into each other.

Define

Define

where is a state variable

The movement equation is

where is the change in variable in a unit time

The solution to the difference equation will be

in the state space.

The equilibrium point is achieved when

When

the equilibrium is stable.

The acceleration

where is a predetermined coefficient.

Let

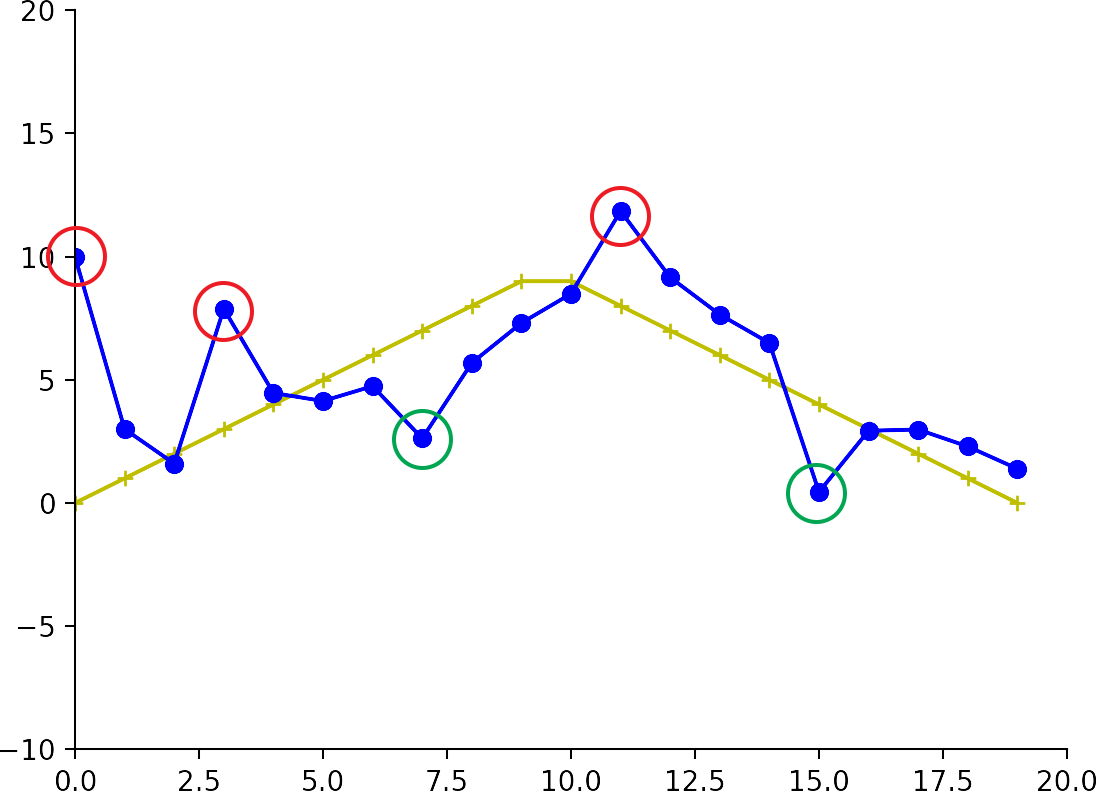
The equilibrium point therefore exists if

Graphing

And

Giving random turbulence to , taking

Taking the value of to follow the table in Appendix C.



*Figure 5: Testing for the navigation system*

A random increase is represented by red circle while a random decrease is represented by green circle.

As shown on the graph, with random turbulence, the system is capable of self-adjusting to approach and achieve the equilibrium state on

## 

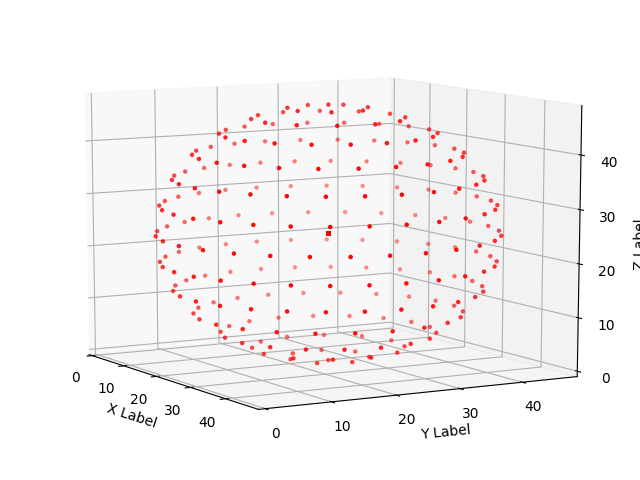
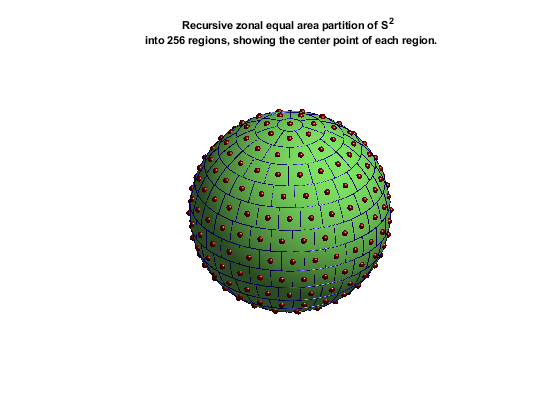
## 

## 3.4 Our Own Design: the Interstellar System

In the third display, we have designed a pattern that simulates planets and its orbital around a star. The star can be represented by drones in a spherical formation. Set the radius of the star, , to be 24 meters.

The surface area of the star,

Using the recursive zonal equal area partition algorithm, each of the planets’ and star’s surface is divided into equal sections, with an area of . A drone is positioned at the geometrical center of each section, as illustrated below.

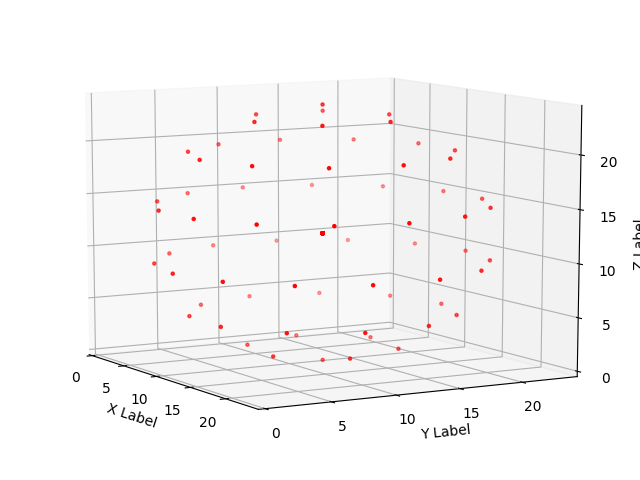
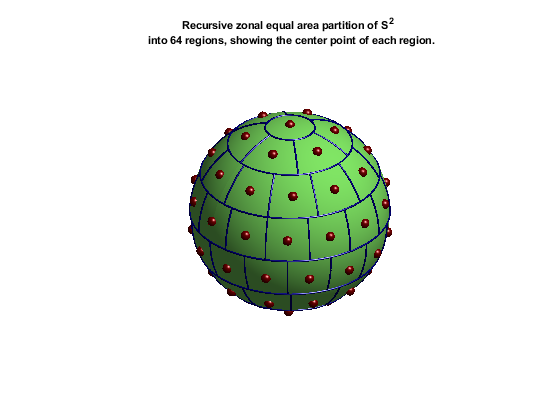


*Figure6: Arrangement of equally distributed drones on the surface of the star*

Number of drones used,

Set the radius of the planet, , to be 12 meters.

The surface area of the planet,



*Figure 7: Arrangement of equally distributed drones on the surface of the planet*

Number of drones used,

Establishing a cartesian coordinates system with axis The origin has the coordinates .

Taking the orbit of the planet to be elliptical with major axis on the axis and minor axis on the axis, centered at the origin .

Assume that the star is located at the focus on the positive x axis,,

In the ellipse,

Hence, center of the star,

The path of the planet’s center is on the ellipse. Hence, it be represented by the 3 by 1 matrix

Based on our calculation above, the relative coordinates of drones used with respect to the center of the ‘star’ or ‘planet’ is recorded. Now shifting the coordinate system horizontally such that the origin of the ‘star’ coordinate system is at

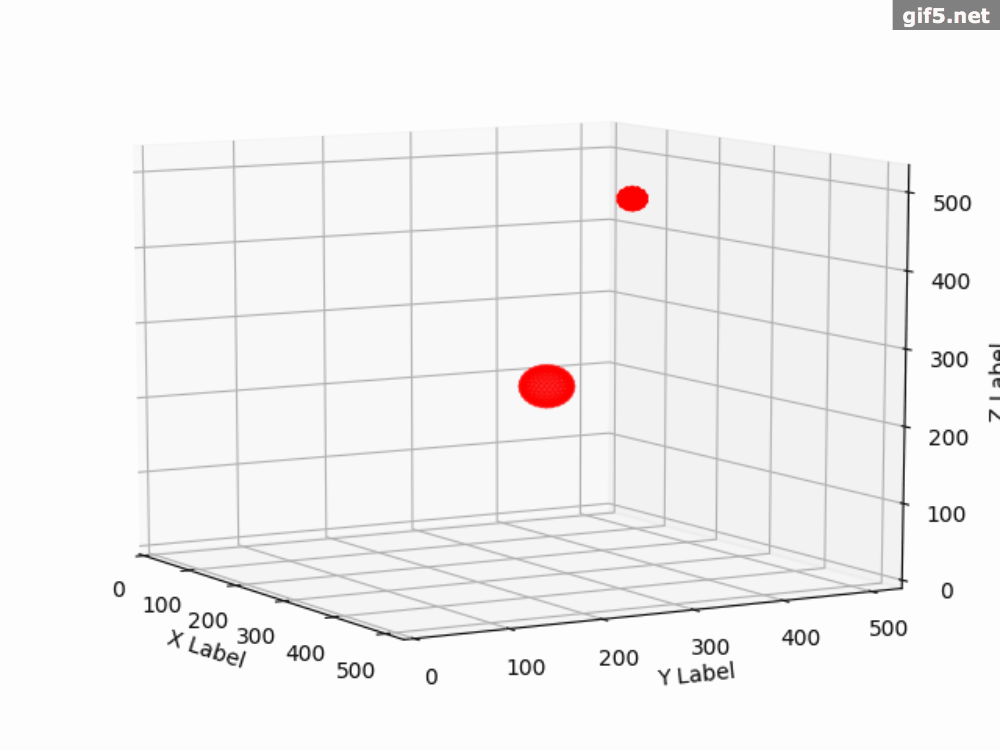
While the origin of the ‘planet’ coordinate system is at the perihelion

A new set of absolute coordinates of drones in the same system is obtained. The locus of each drone is recorded in Appendix B.c.

Based on our aerial show time, the orbital period,

Assuming the planets revolve around the star at a constant angular velocity

Hence, a simulation of a planet-star system is acquired:



*Figure 8: The simulation of planet-star system*

# 

# 

# 4. Part II: Practicality of the Light Show

In order to determine the practicality of the 3-display light show, and make a recommendation to the mayor regarding whether or not it is worth to do the aerial light show, we need to conduct a cost-benefit analysis on the light show which considers a series of specific requirements of the light show which determine how successful the light show might be.

This can be done through the modelling of the cost-benefit ratio, , an index that is reflective of the practicality and manageability of the lightshow.

## 4.1 List of parameters and symbols used

|  |  |
| --- | --- |
| Symbol | Parameters |
|  | 2D area of the displayed design in the air |
|  | Projection area of the display in the audience |
|  | Magnification ratio of 2D design in the air to its projection area in the audience |
|  | Number of drones |
|  | Audience density in the projected area |
|  | Total number of audience engaged |
|  | Angle of elevation of the display from the projection area |
|  | Average rental cost of launch area per unit area per drone |
|  | Average cost of regulating air space per drone |
|  | Duration of the light show |
|  | Duration of set-up prior to the show |
|  | Total cost of renting drones |
|  | Total cost of renting launch area |
|  | Total cost of regulating air space |
|  | Total cost of hiring manpower as operators and monitors |
|  | Rental price of one drone per day |
|  | Total costs spent on the light show |
|  | Total benefits to be reaped from the light show |
|  | Return rate of engaging one of the audience of the light show per unit time |
|  | Height of the display in the air |
|  | The length of the 2D design in air |
|  | The width of the 2D design in air |
|  | The cost-benefit ratio of hosting the event |

## 4.2 Cost Analysis

The total cost of the light show can be broken down into four components,

,

The cost of renting all drones in one day

,

where is the rental price of one drone per day.

Assuming the all drones are equally distributed in the launch area in a tessellation pattern prior to takeoff, the size of launch area is directly proportional to the the number of drones used. Since the renting cost of launch area is calculated based on the size of the area,

,

where average rental cost of launch area per unit area per drone

Similarly, the volume of air space required for drones to operate in is also proportional to the number of drones used. Hence, cost associated with clearing and regulating air space

,

where average cost of regulating air space per drone.

Since the entire drone system during the lightshow can be controlled by only a single operator and a laptop, cost of hiring manpower as operators, , is a fixed cost that does not vary with the number of drones used.

Hence, can be expressed as

,

which is linear with respect to the number of drones.

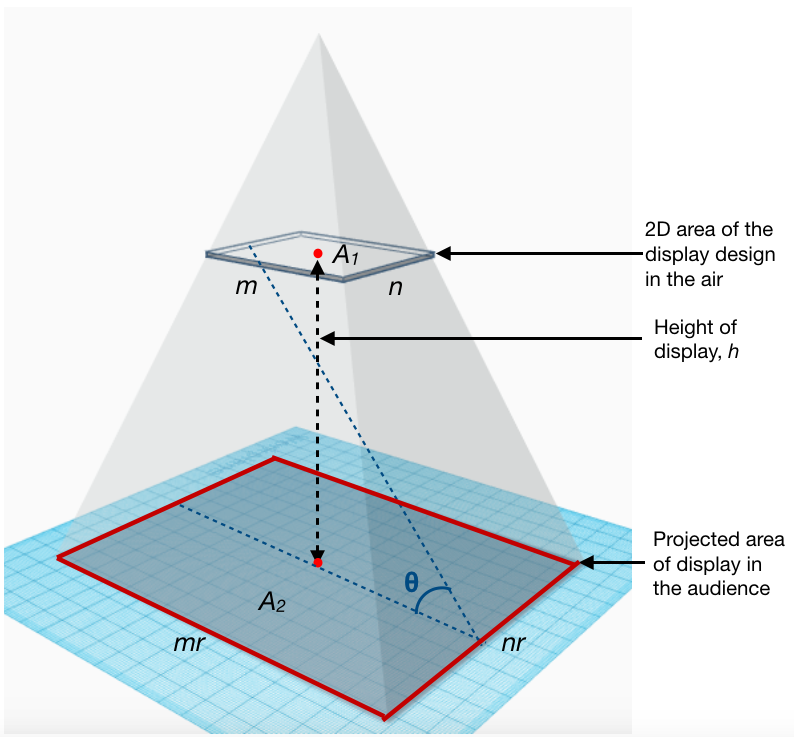
## 4.3 Benefit Analysis

Benefits of hosting such a light show during a festival in terms of monetary values can be modelled by modifying the concept of “effective cost per mille” (eCPM), a calculation of advertisement revenue generated by a banner or campaign, divided by the number of recipients of that advertisement expressed in units of 1,000.

This can be applicable to our context due to the potential of the aerial light show as a medium for aerial advertising, which usually incorporates the use of flogos, manned aircraft or drones to transport or display *static* logos or sponsorship branding. Hence, there is potential monetary revenue to be gained from hosting such light shows when private firms pay to advertise on this platform.

Just as how the eCPM is used to evaluate the return of an advertisement campaign, we created a similar index, Return Rate , which is the monetary revenue to be gained by the host of the event from engaging one of the audience of the light show per unit time, to quantify the total benefits of this aerial light show.

In a 2D display at a height of horizontally above the ground level, the projected area of the display in which the the audience on the ground have a clear and complete view of the display can be illustrated by a rectangular pyramid as illustrated below (Figure 9)



*Figure 9: Rectangular pyramid illustrating the projection area of the display on the ground*

Given a horizontal 2D display of a rectangular shape with length and width at height (cross-section of the rectangular pyramid), the projection area would be directly proportional to the 2D design in the air (base of the rectangular pyramid). Hence, the size of the projection area

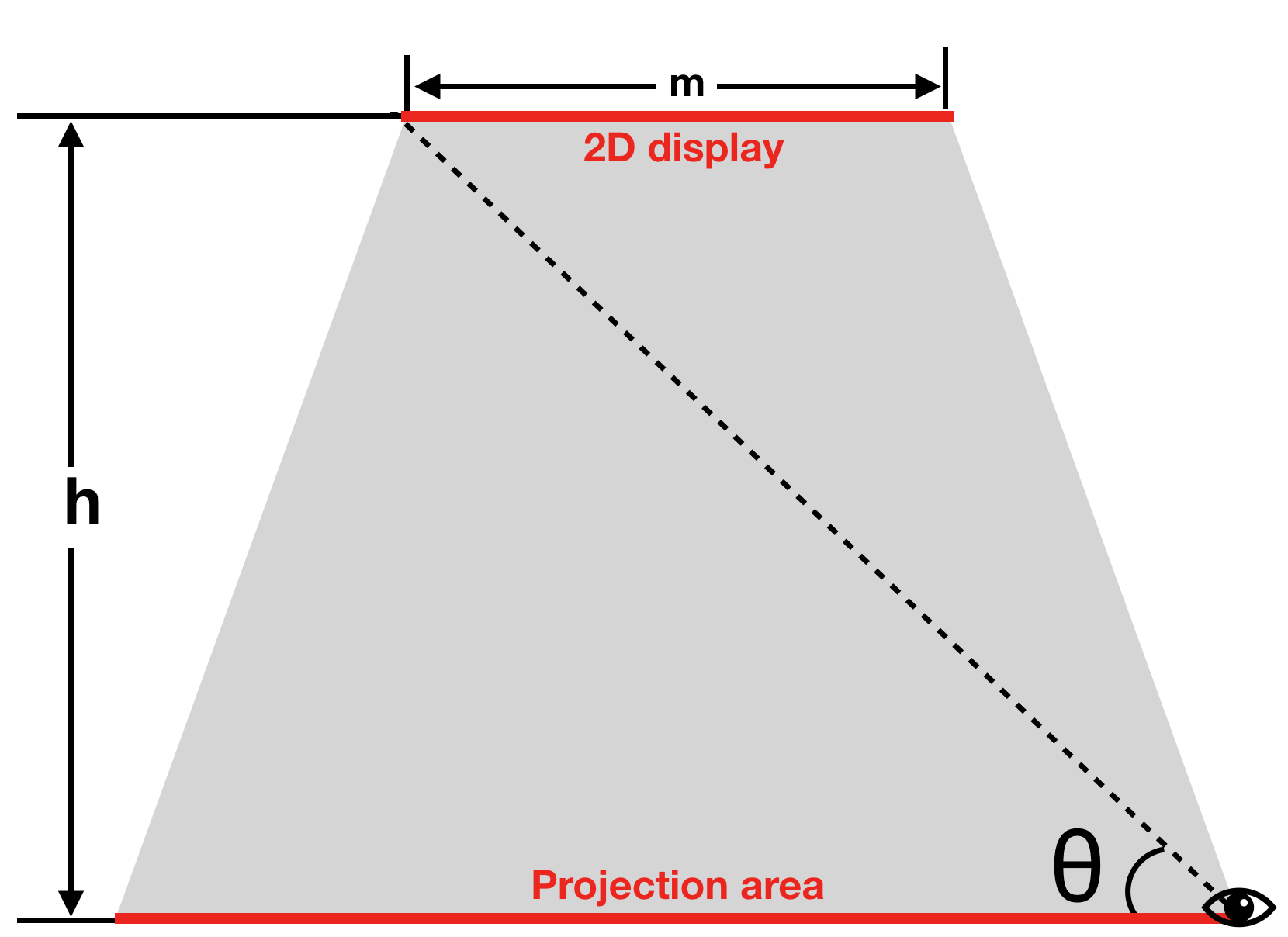
,

where

= area of the displayed design in the air,

= magnification ratio of 2D design in the air to its projection area in the audience.

To obtain a clear and complete view of the entire display, there must be a minimum angle of elevation from any point in the projection area, () (Figure 10).



*Figure 10: Front view of the projection of the 2D display in 3D air space*

Using the minimum angle of elevation, height of the display and size of the display in the air, the projected area

Since , the magnification ratio of any display can be calculated by

The total number of audience engaged, , can be calculated by the product of the projection area of the display in the audience and the audience density in the projection area, .

Hence,

,

Thus, we calculate of total benefit, , as

,

where

= audience density in the projected area,

= Return Rate per capita per unit time,

= Duration of the light show.

## 4.4 Cost-Benefit Analysis

After analysing cost and benefit individually, the cost-benefit ratio can be modelled by

,

further represented by

For the mayor to decide whether he should hold this aerial light show, we need to compare this cost-beneficial ratio against an expected return rate from investing in the light show. This is because apart from gaining revenues from advertisements, this event is likely to have more purposes such as to increase welfare of the general public and improve on social vibrancy, since it is a be a large-scale public celebration event. Thus, covering the cost or making profit is not the sole nor paramount aim. For example, assuming 50% as the mayor’s expected return rate from investing in this light show event, when , it is practical and worthy to host the event.

## 

# 

# 

# 5. Strength of Model

1. Our model balances the visual fidelity of the simulation from a static image design and the number of drones by using pixel by pixel comparison. This allows us to maximise visual fidelity of the display while minimising the number of drones used.
2. We have included fixed stations on the ground in our design which constantly monitor the movement of the entire fleet of drones relative to the ground so as to prevent drones from deviating from the programmed paths all-together and crush into surrounding buildings, ensuring high level of safety,
3. Our model has designed flights paths for each drone so as to minimise possibilities of clashing between drones by ensuring that all drones stay on their respective paths by using the in-built navigation system to keep their positions relative to surrounding drones constant. This ensures high level of safety in the course of the light show.
4. Our model quantifies the benefit our aerial light show by introducing return rate by engaging one of the audience of the light show per unit time. Hence, we are able to evaluate the practicality of the light show.
5. Abstract benefits of hosting the lightshow, such as the satisfaction that the event generates, are also taken into consideration. This is done by checking the calculated value of against 0.5.

# 

# 

# 

# 6. Weakness of Model

1. Our model fails to consider whether there is a better orientation of the display to maximise audience’s enjoyment level. We will be able to determine the best orientation of display if we know the specific location of this event of the location of our targeted audience.
2. In the process of designing and programming our images, our method of selection of points may result in omission of the turning points, making the image slightly lose its original shape.
3. Our model has insufficient database to come up with an accurate Return rate, . Thus, the total benefit estimated might not be accurate.
4. Our equation only quantifies the monetary gain of this event and neglects the satisfaction it generates. Therefore, as long as reaches 50%, we assume this event is successful and worth carrying out.
5. Our model fails to accurately quantify the satisfaction the event generates.
6. Due to time constraint, we are unable to calculate the specific path of two drones to test our Navigation System. With the data of locus of two drones, we are able to calculate of the two drones and hence better verify the robustness of our model.
7. Due to time constraint, we did not follow Kepler’s Laws of Planetary Motion to describe the orbit of the planet around the star. Instead, constant angular velocity is assumed to simplify the model. Additionally, the rotations of the star and planet around their own axes is not considered, making the model less realistic.

# 

# 

# 7. Conclusion

Through analysing the image and evaluation about the cost and benefit of carrying out the aerial light show, we are able to construct models to determine the flight paths and locations of each drone for taking off, transition, and animation as well as ensuring the safety during the flight. Though modelling we are able to make a recommendation to the mayor as to whether we should hold the aerial light show. Our flight paths minimise the possibility of clashing of drones and organisers are advised to make a rational decision by taking all the factors, such as audience density in the projected area, projection area, number of drones, angle of elevation of the display from the projection area, cost of renting drones, cost of renting launch area, cost of regulating air space, cost of hiring manpower as operators and monitors and return rate into consideration. Moreover, we developed algorithm to help organiser to evaluate the cost and benefit of holding this event. By simply keying the factors according to city or venue’s condition, the organiser will be able to weigh the cost and benefit of this event. Our model has taken many factors and parameters into considerations; thus we believe it can help organisers make sound decisions in addition to its maximization of the benefits.

# 8. Bibliography

Intel® Shooting Star™ Drone - Designed for Arts ... (n.d.).

Retrieved November 19, 2017, from

https://www.bing.com/cr?IG=C858209F42B245B59614130C978B2284&CID=0B97483172C66F340D7C430E73C06E0D&rd=1&h=\_Dvd86VcPCvUbjVFaYM0LmzTGOwGd51wZbhQBBNF9k8&v=1&r=https%3a%2f%2fnewsroom.intel.com%2fwp-content%2fuploads%2fsites%2f11%2f2017%2f07%2fIntel-Shooting-Star-Tech-Fact-Sheet-073117-1.pdf&p=DevEx,5065.1

List of tallest buildings in Singapore. (2017, November 14).

Retrieved November 19, 2017, from https://en.wikipedia.org/wiki/List\_of\_tallest\_buildings\_in\_Singapore#cite\_note-MBS\_sky-4

What is eCPM and how is it calculated? (n.d.).

Retrieved November 19, 2017, from

http://www.reviveconsultant.com/articles/what-is-ecpm-and-how-is-it-calculated/

*Drone Light Shows Powered by Intel*. (2017). *Intel*.

Retrieved 19 November 2017, from https://www.intel.sg/content/www/xa/en/technology-innovation/aerial-technology-light-show.html

*Mathematical method for simulating evolution of solar system improved*. (2017). *Phys.org*.

Retrieved 19 November 2017, from

https://phys.org/news/2013-04-mathematical-method-simulating-evolution-solar.ht

ml

*A partition of the unit sphere into regions of equal area and small diameter, Paul Charles Leopardi January 2006*

Retrieved 19 November 2017, from

https://www.researchgate.net/publication/228337855\_A\_partition\_of\_the\_unit\_sphere\_into\_regions\_of\_equal\_area\_and\_small\_diameter

*EQSP: Recursive Zonal Sphere Partitioning Toolbox*

Retrieved 19 November 2017, from

https://www.mathworks.com/matlabcentral/fileexchange/13356-eqsp--recursive-zonal-sphere-partitioning-toolbox

# 9. Appendix

## A. Programme for Recursive Zonal Equal Area Partition (Matlab)

### a. Partition

pdefault.extra\_offset = false;

popt = partition\_options(pdefault, varargin{:});

gdefault.fontsize = 16;

gdefault.show\_title = true;

gdefault.long\_title = false;

gdefault.stereo = false;

gdefault.show\_points = true;

gopt = illustration\_options(gdefault, varargin{:});

opt\_args = option\_arguments(popt,gopt);

subplot(2,2,1);axis off

illustrate\_steps\_1\_2(dim,N,opt\_args);

subplot(2,2,2);axis off

illustrate\_steps\_3\_5(dim,N,opt\_args);

subplot(2,2,3);axis off

illustrate\_steps\_6\_7(dim,N,opt\_args);

subplot(2,2,4);axis off

cla

gopt.fontsize = 32;

switch dim

case 2

opt\_args = option\_arguments(popt,gopt);

project\_s2\_partition(N,opt\_args{:});

case 3

opt\_args = option\_arguments(popt,gopt);

[s,m] = eq\_caps(dim,N);

max\_collar = min(4,size(m,2)-2);

for k = 1:max\_collar

subn = 9+2\*k-mod(k-1,2);

subplot(4,4,subn);axis off

project\_s2\_partition(m(1+k),opt\_args{:});

end

end

%

% end function

function illustrate\_steps\_1\_2(dim,N,varargin)

% Illustrate steps 1 and 2 of the EQ partition of S^dim into N regions;

%

% illustrate\_steps\_1\_2(dim,N,options);

gdefault.fontsize = 14;

gdefault.show\_title = true;

gdefault.long\_title = false;

gopt = illustration\_options(gdefault, varargin{:});

h = [0:1/90:1];

% Plot a circle to represent dth coordinate of S^d

Phi = h\*2\*pi;

plot(sin(Phi),cos(Phi),'k','LineWidth',1)

axis equal;axis off;hold on

c\_polar = polar\_colat(dim,N);

k = [-1:1/20:1];

j = ones(size(k));

% Plot the bounding parallels of the polar caps

plot(sin(c\_polar)\*k, cos(c\_polar)\*j,'r','LineWidth',2)

plot(sin(c\_polar)\*k,-cos(c\_polar)\*j,'r','LineWidth',2)

% Plot the North-South axis

plot(zeros(size(j)),k,'b','LineWidth',1)

% Plot the polar angle

plot(sin(c\_polar)\*h,cos(c\_polar)\*h,'b','LineWidth',2)

text(0.05,2/3,'\theta\_c','Fontsize',gopt.fontsize);

% Plot the ideal collar angle

Delta\_I = ideal\_collar\_angle(dim,N);

theta = c\_polar + Delta\_I;

plot(sin(theta)\*h,cos(theta)\*h,'b','LineWidth',2)

mid = c\_polar + Delta\_I/2;

text(sin(mid)\*2/3,cos(mid)\*2/3,'\Delta\_I','Fontsize',gopt.fontsize);

% Plot an arc to indicate angles

theta = h\*(c\_polar + Delta\_I);

plot(sin(theta)/5,cos(theta)/5,'b','LineWidth',1)

text(-0.9,-0.1,sprintf('V(\\theta\_c) = V\_R \n = \\sigma(S^{%d})/%d',dim,N),...

'Fontsize',gopt.fontsize);

caption\_angle = min(mid + 2\*Delta\_I,pi-c\_polar);

text(sin(caption\_angle)/3,cos(caption\_angle)/3,sprintf('\\Delta\_I = V\_R^{1/%d}',dim),...

'Fontsize',gopt.fontsize);

if gopt.show\_title

title\_str = sprintf('EQ(%d,%d) Steps 1 to 2\n',dim,N);

title(title\_str,'Fontsize',gopt.fontsize);

end

hold off

%

% end function

function illustrate\_steps\_3\_5(dim,N,varargin)

% Illustrate steps 3 to 5 of the EQ partition of S^dim into N regions;

%

% illustrate\_steps\_3\_5(dim,N,options);

gdefault.fontsize = 14;

gdefault.show\_title = true;

gdefault.long\_title = false;

gopt = illustration\_options(gdefault, varargin{:});

h = [0:1/90:1];

Phi = h\*2\*pi;

plot(sin(Phi),cos(Phi),'k','LineWidth',1)

axis equal;axis off;hold on

c\_polar = polar\_colat(dim,N);

n\_collars = num\_collars(N,c\_polar,ideal\_collar\_angle(dim,N));

r\_regions = ideal\_region\_list(dim,N,c\_polar,n\_collars);

s\_cap = cap\_colats(dim,N,c\_polar,r\_regions);

k = [-1:1/20:1];

j = ones(size(k));

plot(sin(c\_polar)\*k, cos(c\_polar)\*j,'r','LineWidth',2);

plot(zeros(size(j)),k,'b','LineWidth',1)

for collar\_n = 0:n\_collars

zone\_n = 1+collar\_n;

theta = s\_cap(zone\_n);

plot(sin(theta)\*h,cos(theta)\*h,'b','LineWidth',2);

theta\_str = sprintf('\\theta\_{F,%d}',zone\_n);

text(sin(theta)\*1.1,cos(theta)\*1.1,theta\_str,'Fontsize',gopt.fontsize);

if collar\_n ~= 0

plot(sin(theta)\*k, cos(theta)\*j,'r','LineWidth',2);

theta\_p = s\_cap(collar\_n);

arc = theta\_p + (theta-theta\_p)\*h;

plot(sin(arc)/5,cos(arc)/5,'b','LineWidth',1);

mid = (theta\_p + theta)/2;

text(sin(mid)/2,cos(mid)/2,'\Delta\_F','Fontsize',gopt.fontsize);

y\_str = sprintf('y\_{%d} = %3.1f...',collar\_n,r\_regions(zone\_n));

text(-sin(mid)+1/20,cos(mid)+(mid-pi)/30,y\_str,'Fontsize',gopt.fontsize);

end

end

if gopt.show\_title

title\_str = sprintf('EQ(%d,%d) Steps 3 to 5\n',dim,N);

title(title\_str,'Fontsize',gopt.fontsize);

end

hold off

%

% end function

function illustrate\_steps\_6\_7(dim,N,varargin)

% Illustrate steps 6 to 7 of the EQ partition of S^dim into N regions;

%

% illustrate\_steps\_6\_7(dim,N,options);

gdefault.fontsize = 14;

gdefault.show\_title = true;

gdefault.long\_title = false;

gopt = illustration\_options(gdefault, varargin{:});

h = [0:1/90:1];

Phi = h\*2\*pi;

plot(sin(Phi),cos(Phi),'k','LineWidth',1)

axis equal;axis off;hold on

c\_polar = polar\_colat(dim,N);

n\_collars = num\_collars(N,c\_polar,ideal\_collar\_angle(dim,N));

r\_regions = ideal\_region\_list(dim,N,c\_polar,n\_collars);

n\_regions = round\_to\_naturals(N,r\_regions);

s\_cap = cap\_colats(dim,N,c\_polar,n\_regions);

k = [-1:1/20:1];

j = ones(size(k));

plot(sin(c\_polar)\*k, cos(c\_polar)\*j,'r','LineWidth',2);

plot(zeros(size(j)),k,'b','LineWidth',1)

for collar\_n = 0:n\_collars

zone\_n = 1+collar\_n;

theta = s\_cap(zone\_n);

plot(sin(theta)\*h,cos(theta)\*h,'b','LineWidth',2);

theta\_str = sprintf('\\theta\_{%d}',zone\_n);

text(sin(theta)\*1.1,cos(theta)\*1.1,theta\_str,'Fontsize',gopt.fontsize);

if collar\_n ~= 0

plot(sin(theta)\*k, cos(theta)\*j,'r','LineWidth',2);

theta\_p = s\_cap(collar\_n);

arc = theta\_p + (theta-theta\_p)\*h;

plot(sin(arc)/5,cos(arc)/5,'b','LineWidth',1);

mid = (theta\_p + theta)/2;

Delta\_str = sprintf('\\Delta\_{%i}',collar\_n);

text(sin(mid)/2,cos(mid)/2,Delta\_str,'Fontsize',gopt.fontsize);

m\_str = sprintf('m\_{%d} =%3.0f',collar\_n,n\_regions(zone\_n));

text(-sin(mid)+1/20,cos(mid)+(mid-pi)/30,m\_str,'Fontsize',gopt.fontsize);

end

end

if gopt.show\_title

title\_str = sprintf('EQ(%d,%d) Steps 6 to 7\n',dim,N);

title(title\_str,'Fontsize',gopt.fontsize);

end

hold off

%

% end function

function arg = option\_arguments(popt,gopt)

k = 1;

if isfield(popt,'extra\_offset')

arg{k} = 'offset';

if popt.extra\_offset

arg{k+1} = 'extra';

else

arg{k+1} = 'normal';

end

k = k+2;

end

if isfield(gopt,'fontsize')

arg{k} = 'fontsize';

arg{k+1} = gopt.fontsize;

k = k+2;

end

if isfield(gopt,'stereo')

arg{k} = 'proj';

if gopt.stereo

arg{k+1} = 'stereo';

else

arg{k+1} = 'eqarea';

end

k = k+2;

end

if isfield(gopt,'show\_title')

arg{k} = 'title';

if gopt.show\_title

if isfield(gopt,'long\_title')

if gopt.long\_title

arg{k+1} = 'long';

else

arg{k+1} = 'short';

end

else

arg{k+1} = 'show';

end

else

arg{k+1} = 'none';

end

k = k+2;

elseif isfield(gopt,'long\_title')

arg{k} = 'title';

if gopt.long\_title

arg{k+1} = 'long';

else

arg{k+1} = 'short';

end

k = k+2;

end

if isfield(gopt,'show\_points')

arg{k} = 'points';

if gopt.show\_points

arg{k+1} = 'show';

else

arg{k+1} = 'hide';

end

k = k+2;

end

if isfield(gopt,'show\_surfaces')

arg{k} = 'surf';

if gopt.show\_surfaces

arg{k+1} = 'show';

else

arg{k+1} = 'hide';

end

k = k+2;

end

### b. Illustration

pdefault.extra\_offset = false;

popt = partition\_options(pdefault, varargin{:});

gdefault.fontsize = 16;

gdefault.show\_title = true;

gdefault.show\_points = true;

gdefault.show\_sphere = true;

gopt = illustration\_options(gdefault, varargin{:});

dim = 2;

surf\_jet;

if gopt.show\_title

if gopt.show\_points

pointstr = ', showing the center point of each region';

else

pointstr = '';

end

titlestr = sprintf(...

'\nRecursive zonal equal area partition of {S^2} \n into %d regions%s.',...

N,pointstr);

title(titlestr,'FontWeight','bold','FontUnits','normalized',...

'FontSize',gopt.fontsize/512);

end

frame\_no = 1;

if nargout > 0

movie\_frame(frame\_no) = getframe(gcf);

frame\_no = frame\_no + 1;

end

if gopt.show\_sphere

show\_s2\_sphere;

hold on

if nargout > 0

movie\_frame(frame\_no) = getframe(gcf);

frame\_no = frame\_no + 1;

end

end

R = eq\_regions(dim,N,popt.extra\_offset);

top\_colat = 0;

for i = N:-1:2

if top\_colat ~= R(2,1,i)

top\_colat = R(2,1,i);

pause(0);

end

show\_s2\_region(R(:,:,i),N);

if nargout > 0

movie\_frame(frame\_no) = getframe(gcf);

frame\_no = frame\_no + 1;

end

end

if gopt.show\_points

x = eq\_point\_set(dim,N,popt.extra\_offset);

show\_r3\_point\_set(x,'sphere','hide','title','hide');

hold on

if nargout > 0

movie\_frame(frame\_no) = getframe(gcf);

frame\_no = frame\_no + 1;

end

end

hold off

%

% end function

function show\_s2\_region(region,N)

%SHOW\_S2\_REGION Illustrate a region of S^2

%

%Syntax

% show\_s2\_region(region,N);

%

%Description

% SHOW\_S2\_REGION(REGION,N) uses 3D surface plots to illustrate a region of S^2.

% The region is given as a 2 x 2 matrix in spherical polar coordinates

tol = eps\*2^5;

dim = size(region,1);

t = region(:,1);

b = region(:,2);

if abs(b(1)) < tol

b(1) = 2\*pi;

end

pseudo = 0;

if abs(t(1)) < tol && abs(b(1)-2\*pi) < tol

pseudo = 1;

end

n = 21;

delta = 1/(n-1);

h = 0:delta:1;

t\_to\_b = zeros(dim,n);

b\_to\_t = t\_to\_b;

r = sqrt(1/N)/12;

for k = 1:dim

if ~pseudo || k < 2

L = 1:dim;

j(L) = mod(k+L,dim)+1;

t\_to\_b(j(1),:) = t(j(1))+(b(j(1))-t(j(1)))\*h;

t\_to\_b(j(2),:) = t(j(2))\*ones(1,n);

t\_to\_b\_x = polar2cart(t\_to\_b);

[X,Y,Z] = fatcurve(t\_to\_b\_x,r);

surface(X,Y,Z,-ones(size(Z)),...

'FaceColor','interp','FaceLighting','phong','EdgeColor','none')

axis equal

hold on

end

end

grid off

axis off

%

% end function

Other programs using Python

### c. Tracing the coordinates from the image and calculate the similarity

import json

import queue

import sys

import warnings

import matplotlib.image

import numpy

BLACK = [0., 0., 0., 1] # The color that need to be displayed

def load\_image(filename):

# Return an np.array

return matplotlib.image.imread(filename)

def cal\_angle(a, b, c):

# a, b, c: dict ['position': (x, y)]

# return

a\_p = numpy.array(a['position'])

b\_p = numpy.array(b['position'])

c\_p = numpy.array(c['position'])

ab = b\_p - a\_p

bc = c\_p - b\_p

# sin(abc)

with warnings.catch\_warnings():

warnings.filterwarnings('error')

try:

cross = numpy.cross(ab, bc)

ab\_norm = numpy.linalg.norm(ab)

bc\_norm = numpy.linalg.norm(bc)

return (cross / ab\_norm / bc\_norm)

except Warning:

print('ab: {0}'.format(ab), flush=True)

print('bc: {0}'.format(bc), flush=True)

assert(False)

def is\_black(pixel):

result = (pixel == BLACK)

for boolean in result:

if not boolean:

return False

return True # if the pixel is black. i.e. should this pixel be displayed

def at\_edge(point, image):

# point: tuple(x, y)

# image: numpy.array(2D), pixels

# return: Bool. At edge & should be

if (point[0] < 0 or point[1] < 0 or point[0] >= image.shape[0] or point[1] >= image.shape[1]):

return False

if (is\_black(image[point[0]][point[1]]) and (point[0] == 0 or point[1] == 0 or point[0] == image.shape[0] - 1 or point[1] == image.shape[1] - 1)):

return True

if (is\_black(image[point[0]][point[1]])):

for inc in [(-1, -1), (-1, 0), (-1, 1), (0, 1), (1, 1), (1, 0), (1, -1), (0, -1)]:

new\_point = ((point[0] + inc[0]), (point[1] + inc[1]))

try:

if (not is\_black(image[new\_point[0]][new\_point[1]])):

return True

except IndexError:

assert(False)

return True

return False

def next\_point(point, image, visited):

# point: {'position': tuple(x, y)}

# image: numpy.array(2D), pixels

# visited: dict of tuple(s) (x, y), denote the points that has been visited.

Queue = queue.Queue()

Queue.put(point['position'])

test\_point = point['position']

while (not Queue.empty()):

point = Queue.get()

for inc in [(-1, -1), (-1, 0), (-1, 1), (0, 1), (1, 1), (1, 0), (1, -1), (0, -1)]:

new\_point = ((point[0] + inc[0]), (point[1] + inc[1]))

if (new\_point not in visited):

if (at\_edge(new\_point, image)):

# if (numpy.sqrt((test\_point[0] - new\_point[0])\*\*2 + (test\_point[1] - new\_point[1])\*\*2 > 2)):

# print("Warning: distance > 2! Origin: {0}; point: {3} Current: {1}; inc: {2}".format(

# test\_point, new\_point, inc, point), flush=True)

# for x in (-1, 0, 1):

# print("{0} {1} {2}".format(is\_black(image[point[0] + x][point[1] - 1]), is\_black(

# image[point[0] + x][point[1]]), is\_black(image[point[0] + x][point[1] + 1])))

visited[new\_point] = True

return {'position': new\_point}

else:

if (point[0] < 0 or point[1] < 0 or point[0] >= image.shape[0] or point[1] >= image.shape[1]):

visited[new\_point] = True

Queue.put(new\_point)

else:

pass

# print('INFO: Pop: {0}'.format(point), flush=True)

return None

def find\_angle(image):

# Return a list of dict(s) with

# ['position': (x, y), 'angle':]

# dict of tuple(s) (x, y), denote the points that has been visited.

visited = {}

# Find the first point

point = None

y = 0

for row in image:

x = 0

for pixel in row:

# if (is\_black(pixel)):

# if (is\_black(pixel) and not at\_edge((x, y), image)):

# print("Warning: pixel not at edge! ({0}, {1})".format(

# x, y), flush=True)

# for xx in (-1, 0, 1):

# print("{0} {1} {2}".format(is\_black(image[x + xx][y - 1]), is\_black(

# image[x + xx][y]), is\_black(image[x + xx][y + 1])))

if (at\_edge((x, y), image) and (x, y) not in visited): # if $pixel need to be drawn

point = {'position': (x, y)}

visited[point['position']] = True

break

x += 1

y += 1

if (point is not None):

break

print(point, flush=True)

angle = []

angle.append(point)

point = next\_point(point, image, visited)

if (point is not None):

angle[-1]['next'] = point

angle.append(point)

point = next\_point(point, image, visited)

while (point is not None):

try:

point['angle'] = cal\_angle(angle[-2], angle[-1], point)

except AssertionError:

print('angle[-2]: {0}\nangle[-1]: {1}\npoint: {2}'.format(

angle[-2], angle[-1], point), flush=True)

assert(False)

angle[-1]['next'] = point

angle.append(point)

point = next\_point(point, image, visited)

i = 1

Traced = False

while (point is None and i < len(angle)):

Traced = True

point = next\_point(angle[-i], image, visited)

i += 1

if (Traced):

print("INFO: Trace back {0} steps".format(i), flush=True)

if len(angle) % 100 == 0:

# print("INFO: {0}".format(angle[-1]), flush=True)

print("INFO: {0} points has been identified".format(

len(angle)), flush=True)

# for i in angle:

# print(i['position'], flush=True)

angle[-1]['next'] = angle[0]

if (len(angle) > 2):

angle[0]['angle'] = cal\_angle(angle[-2], angle[-1], angle[0])

angle[1]['angle'] = cal\_angle(angle[-1], angle[0], angle[1])

import matplotlib.pyplot as plt

plt.figure(figsize=(256, 256), dpi=32)

xx = []

yy = []

for x, y in visited.keys():

xx.append(x)

yy.append(y)

plt.axis([0, image.shape[1], 0, image.shape[0]])

plt.scatter(yy, xx, s=32)

plt.savefig('visited.png')

plt.cla()

return angle

def line\_to(start, end, image):

if (start[0] == end[0]):

if (start[1] > end[1]):

start, end = end, start

for i in range(start[1], end[1] + 1):

image[start[0]][i] = BLACK

else:

if (start[0] < end[0]):

start, end = end, start

inc = (start[1] - end[1]) / (start[0] - end[0])

for i in range(start[0] - end[0] + 1):

y = int(start[1] - inc \* i)

if (y >= image.shape[1]):

y = image.shape[1] - 1

try:

image[start[0] - i][y] = BLACK

except IndexError:

assert(False)

def fit\_image(vertices, shape):

image = numpy.ones(shape)

for i in range(-1, len(vertices), 1):

line\_to(vertices[i], vertices[(i + 1) % len(vertices)], image)

return image

def similarity(vertices, image):

# vertices: list of tuple(s)(x, y) in order.

image\_fitted = fit\_image(vertices, image.shape)

import matplotlib.pyplot as plt

xx = []

yy = []

for x in range(image\_fitted.shape[0]):

for y in range(image\_fitted.shape[1]):

if (is\_black(image\_fitted[x][y])):

xx.append(x)

yy.append(y)

plt.axis([0, image\_fitted.shape[1], 0, image\_fitted.shape[0]])

plt.scatter(yy, xx, s=32)

plt.savefig('fitted.png')

plt.cla()

correct = 0

incorrect = 0

for x in range(image\_fitted.shape[0]):

for y in range(image\_fitted.shape[1]):

f = is\_black(image\_fitted[x][y])

if (at\_edge((x, y), image)): # or f):

i = is\_black(image[x][y])

if (f or i):

if (f == i):

correct += 1

else:

incorrect += 1

# print(incorrect + correct, flush=True)

return (correct / (incorrect + correct))

def processing(angle, image):

# import matplotlib.pyplot as plt

# img = numpy.ones(image.shape)

# for i in range(len(angle)):

# img[angle[i]['position'][0]][angle[i]['position'][1]] = BLACK

# plt.imshow(img)

# plt.axis('off')

# plt.savefig('image.png')

# plt.cla()

import matplotlib.pyplot as plt

xx = []

yy = []

for a in angle:

x, y = a['position']

xx.append(x)

yy.append(y)

plt.axis([0, image.shape[1], 0, image.shape[0]])

plt.scatter(yy, xx, s=32)

plt.savefig('image.png')

plt.cla()

threshold = 0.2 # Assumption

max\_len = len(angle)

min\_len = 0

vertices = []

while (max\_len > min\_len):

backup = [element for element in angle]

# test\_len = (max\_len + min\_len) >> 1

test\_len = 320

vertices.clear()

i = 0

j = 0

while i < len(angle):

if (((j \* (test\_len)) % (max\_len)) >= test\_len):

angle[i - 1]['next'] = angle[(i + 1) % len(angle)]

angle.pop(i)

i -= 1

else:

vertices.append(angle[i]['position'])

i += 1

j += 1

if (len(angle) == len(backup)):

return vertices

sim = similarity(vertices, image)

if (sim >= threshold):

max\_len = len(vertices) - 1

else:

angle = backup

if (min\_len == len(vertices)):

break

min\_len = len(vertices)

print("INFO: One iteration completed.", flush=True)

print("INFO: Max points: {0}; Min points: {1}; Similarity: {2}; size of vertices: {3}".format(

max\_len, min\_len, sim, len(vertices)), flush=True)

break

return vertices

if \_\_name\_\_ == '\_\_main\_\_':

print("INFO: Task Start", flush=True)

image = load\_image(sys.argv[1])

print("INFO: Image Loaded", flush=True)

angle = find\_angle(image)

\_angle = [{'position': element['position']} for element in angle]

json.dump(\_angle, open('angle.json', 'w'))

print("INFO: Points at edges identified", flush=True)

vertices = processing(angle, image)

json.dump(vertices, open('vertices.json', 'w'))

print(vertices)

import matplotlib.pyplot as plt

xx = []

yy = []

for y, x in vertices:

xx.append(x)

yy.append(y)

plt.axis([0, image.shape[1], 0, image.shape[0]])

plt.scatter(xx, yy, s=32)

plt.savefig('vertices.png')

plt.cla()

print("INFO: Task Complete", flush=True)

### d. Simulation of Difference Equation

import random

import matplotlib.pyplot as plt

import numpy as np

def v\_n\_1(v, tol, k=-0.7):

# -k(v(n+1) - v(n))t=v\_{tol\_n}

return (tol / (-k)) + v

def v\_n(v, tol, tol\_1):

# v(n)-v(n-1)=v\_{tol\_n}

return tol - tol\_1 + v

def func(x, tol, v\_0):

v = np.zeros(x.shape)

v[0] = v\_0

for i in x[1:]:

v[i] = v\_n(v[i - 1], tol[i], tol[i - 1])

return v

def a\_n(v\_n, tol, k=-0.7):

# a=-k(v(n)-vtol(n))

return (-k) \* (v\_n - tol)

def funcK(x, tol, v\_0):

# Calculate v

t = 1

v = np.zeros(x.shape)

v[0] = v\_0

for i in x[1:]:

# v[i + 1] = v\_n\_1(v[i], tol[i + 1])

if (i % 4 == 0):

disturbance = random.randint(-10, 10)

v[i - 1] += disturbance

print(disturbance)

v[i] = v[i - 1] - a\_n(v[i - 1], tol[i - 1]) \* t

return v

if \_\_name\_\_ == '\_\_main\_\_':

v\_0 = 10

x = np.array(list(range(20)))

tol = np.array(list(list(range(10)) + list(range(9, -1, -1))))

t = x

yK = funcK(x, tol, v\_0)

# y = func(x, tol, v\_0)

plt.plot(t, tol, marker='+', color='y', linestyle='-')

plt.plot(t, yK, marker='o', color='b', linestyle='-')

# plt.plot(t, y, 'go')

print(yK)

plt.axis([0, 20, -10, 20])

plt.show()

### e. Generate the animation during take off procedure

import json

from sys import argv

import matplotlib.pyplot as plt

import numpy

DefaultColor = '#000000'

BackgroundColor = '#FFFFFF'

def main(path, color=DefaultColor):

# data: list: [tuple:(x, y)]

xx = numpy.array(list(range(20)) \* 16, dtype='float32') + 0.5

yy = numpy.array([y for y in range(16)

for i in range(20)], dtype='float32') + 0.5

plt.figure(figsize=(20, 16), dpi=32)

plt.axis([0, 20, 0, 16])

plt.scatter(xx, yy, c=DefaultColor, s=32)

horizontal = numpy.linspace(0, 20)

vertical = numpy.linspace(0, 16)

for x in range(20):

for y in range(16):

plt.plot(horizontal, numpy.ones(horizontal.shape) \* y, c='b')

plt.plot(numpy.ones(vertical.shape) \* x, vertical, c='b')

plt.savefig(path)

plt.cla()

if \_\_name\_\_ == '\_\_main\_\_':

main(argv[1])

### f. Transfer the polar coordinates to Cartesian coordinates for the Ferris Wheel

import json

import sys

import numpy

def loadCSV(filename):

return numpy.array(list(map(float, open(filename, 'r').read().split())))

# return open(filename, 'r').read().split()

def loadCSV2(filename):

return list(map(lambda x: tuple(map(float, x.split(','))), open(filename, 'r').read().split('\n')))

def polar2Cart(angle, r=75):

x = numpy.cos(angle / 180 \* numpy.pi) \* r

y = numpy.sin(angle / 180 \* numpy.pi) \* r

cart = []

for i in range(x.shape[0]):

cart.append((x[i], y[i],))

return cart

if \_\_name\_\_ == '\_\_main\_\_':

angle = loadCSV(sys.argv[1])

wheel = polar2Cart(angle)

spoke = []

for r in numpy.linspace(12.5, 75, num=5, endpoint=False):

spoke.extend(polar2Cart(numpy.linspace(

2 \* numpy.pi, 0, num=17, endpoint=False), r=r))

triangle = loadCSV2(sys.argv[2])

ferris = wheel + triangle + spoke + [(0, 0)]

x = 0

y = 0

for ix, iy in ferris:

x = min(x, ix)

y = min(y, iy)

ferris = numpy.array(ferris)

for i in range(ferris.shape[0]):

ferris[i][0] -= x

ferris[i][1] -= y

ferris = ferris.tolist()

json.dump(ferris, open(sys.argv[3], 'w'))

### g. Program used to design the trail of the take-off stage

import json

import sys

import matplotlib.pyplot

import numpy

from mpl\_toolkits.mplot3d import Axes3D

DefaultColor = 0

BackgroundColor = 1

Height = 200

Step = 80

origin\_height = int(sys.argv[3])

target\_height = int(sys.argv[4])

def line(origin, target):

# print(origin, target)

x = numpy.linspace(origin[0], target[0], num=Step, endpoint=True)

y = numpy.linspace(origin[1], target[1], num=Step, endpoint=True)

z = numpy.linspace(origin[2], target[2], num=Step, endpoint=True)

return (x.tolist(), y.tolist(), z.tolist(),)

def pair(origin\_img, target\_img):

p = []

updated = True

while updated:

updated = False

prev = None

for i in range(origin\_img.shape[0]):

for j in range(origin\_img.shape[1]):

if origin\_img[i][j][origin\_height] > 0:

prev = (i, j, origin\_height)

origin\_img[i][j][origin\_height] -= 1

break

if prev is not None:

break

for i in range(target\_img.shape[0]):

for j in range(target\_img.shape[1]):

if target\_img[i][j][target\_height] > 0:

p.append(((prev[0], prev[1], prev[2]),

(i, j, target\_height), ))

target\_img[i][j][target\_height] -= 1

updated = True

break

if updated:

break

return p

print("INFO: Start", flush=True)

fig = matplotlib.pyplot.figure()

# ax = fig.add\_subplot(111, projection='3d')

ax = Axes3D(fig)

origin\_vertices = json.load(open(sys.argv[1], 'r'))

target\_vertices = json.load(open(sys.argv[2], 'r'))

print("INFO: Vertices loaded", flush=True)

x\_max = 0

y\_max = 0

for i in range(len(origin\_vertices)):

origin\_vertices[i][0] = int(origin\_vertices[i][0])

origin\_vertices[i][1] = int(origin\_vertices[i][1])

for i in range(len(target\_vertices)):

target\_vertices[i][0] = int(target\_vertices[i][0])

target\_vertices[i][1] = int(target\_vertices[i][1])

for x, y in origin\_vertices:

x\_max = max(x\_max, x)

y\_max = max(y\_max, y)

for x, y in target\_vertices:

x\_max = max(x\_max, x)

y\_max = max(y\_max, y)

origin = numpy.ones((x\_max + 1, y\_max + 1, max(origin\_height, target\_height) +

1), dtype='int64') \* BackgroundColor

target = numpy.ones(origin.shape) \* BackgroundColor

assert(len(origin\_vertices) == len(target\_vertices))

for vertex in origin\_vertices:

origin[vertex[0]][vertex[1]][origin\_height] += 1

for vertex in target\_vertices:

target[vertex[0]][vertex[1]][target\_height] += 1

print("INFO: Intermediate graph plotted", flush=True)

assert(len(origin) == len(target))

pairs = pair(origin, target)

print("INFO: Pairs found", flush=True)

x = []

y = []

z = []

xn = []

yn = []

zn = []

for pair in pairs:

xx, yy, zz = line(pair[0], pair[1])

x.extend(xx)

y.extend(yy)

z.extend(zz)

xn.append(xx)

yn.append(yy)

zn.append(zz)

print("INFO: Trail Designed", flush=True)

for i in range(Step):

ax = Axes3D(fig)

ax.axis([0, x\_max, 0, y\_max])

ax.set\_zlim(min(

origin\_height, target\_height), max(origin\_height, target\_height))

for j in range(len(xn)):

ax.scatter(xn[j][i], yn[j][i], zn[j][i], c='r', marker='.')

ax.set\_xlabel('X Label')

ax.set\_ylabel('Y Label')

ax.set\_zlabel('Z Label')

ax.view\_init(elev=45., azim=315.)

matplotlib.pyplot.savefig("movie%d.png" % i)

matplotlib.pyplot.cla()

# ax.scatter(xt, yt, zt, c='b', marker='^')

# ax.set\_xlabel('X Label')

# ax.set\_ylabel('Y Label')

# ax.set\_zlabel('Z Label')

# matplotlib.pyplot.show()

# for ii in range(0, 360, 1):

# ax.view\_init(elev=10., azim=ii)

# matplotlib.pyplot.savefig("movie%d.png" % ii)

# fig.savefig('tmp.png')

print("INFO: Completed", flush=True)

## B. Table of coordinates of drones deployed

### a. Ferris Wheel

|  |  |
| --- | --- |
| 149.9342123 | 142.6141814 |
| 149.7369644 | 145.7493456 |
| 149.4086026 | 148.8734999 |
| 148.9497028 | 151.9811634 |
| 148.3610701 | 155.0668842 |
| 147.6437371 | 158.1252489 |
| 146.7989623 | 161.1508922 |
| 145.8282278 | 164.1385059 |
| 144.7332364 | 167.0828489 |
| 143.5159093 | 169.9787556 |
| 142.178382 | 172.8211458 |
| 140.723001 | 175.605033 |
| 139.1523195 | 178.3255331 |
| 137.4690931 | 180.9778736 |
| 135.6762746 | 183.5574013 |
| 133.7770093 | 186.0595909 |
| 131.7746292 | 188.4800527 |
| 129.6726471 | 190.8145403 |
| 127.4747505 | 193.0589584 |
| 125.1847955 | 195.2093693 |
| 122.8067992 | 197.2620006 |
| 120.3449336 | 199.2132513 |
| 117.8035176 | 201.0596981 |
| 115.1870096 | 202.7981018 |
| 112.5 | 204.4254127 |
| 109.7472026 | 205.9387758 |
| 106.9334469 | 207.3355363 |
| 104.063669 | 208.6132438 |
| 101.1429035 | 209.7696566 |
| 98.17627458 | 210.8027461 |
| 95.16898655 | 211.7106999 |
| 92.12631526 | 212.4919251 |
| 89.05359859 | 213.1450512 |
| 85.95622714 | 213.6689324 |
| 82.83963475 | 214.0626496 |
| 79.70928896 | 214.325512 |
| 76.57068149 | 214.4570587 |
| 73.42931851 | 214.4570587 |
| 70.29071104 | 214.325512 |
| 67.16036525 | 214.0626496 |
| 64.04377286 | 213.6689324 |
| 60.94640141 | 213.1450512 |
| 57.87368474 | 212.4919251 |
| 54.83101345 | 211.7106999 |
| 51.82372542 | 210.8027461 |
| 48.85709645 | 209.7696566 |
| 45.93633102 | 208.6132438 |
| 43.06655313 | 207.3355363 |
| 40.25279737 | 205.9387758 |
| 37.5 | 204.4254127 |
| 34.81299038 | 202.7981018 |
| 32.19648242 | 201.0596981 |
| 29.65506639 | 199.2132513 |
| 27.19320077 | 197.2620006 |
| 24.81520452 | 195.2093693 |
| 22.52524946 | 193.0589584 |
| 20.32735294 | 190.8145403 |
| 18.22537083 | 188.4800527 |
| 16.2229907 | 186.0595909 |
| 14.32372542 | 183.5574013 |
| 12.53090695 | 180.9778736 |
| 10.84768049 | 178.3255331 |
| 9.276998997 | 175.605033 |
| 7.821617982 | 172.8211458 |
| 6.484090677 | 169.9787556 |
| 5.266763558 | 167.0828489 |
| 4.171772232 | 164.1385059 |
| 3.201037685 | 161.1508922 |
| 2.356262915 | 158.1252489 |
| 1.638929945 | 155.0668842 |
| 1.05029722 | 151.9811634 |
| 0.591397401 | 148.8734999 |
| 0.263035556 | 145.7493456 |
| 0.065787743 | 142.6141814 |
| 0 | 139.4735074 |
| 0.065787743 | 136.3328334 |
| 0.263035556 | 133.1976692 |
| 0.591397401 | 130.0735149 |
| 1.05029722 | 126.9658514 |
| 1.638929945 | 123.8801306 |
| 2.356262915 | 120.8217659 |
| 3.201037685 | 117.7961226 |
| 4.171772232 | 114.8085089 |
| 5.266763558 | 111.8641659 |
| 6.484090677 | 108.9682592 |
| 7.821617982 | 106.125869 |
| 9.276998997 | 103.3419818 |
| 10.84768049 | 100.6214817 |
| 12.53090695 | 97.96914121 |
| 14.32372542 | 95.38961348 |
| 16.2229907 | 92.88742388 |
| 18.22537083 | 90.4669621 |
| 20.32735294 | 88.13247446 |
| 22.52524946 | 85.88805643 |
| 24.81520452 | 83.73764549 |
| 27.19320077 | 81.68501419 |
| 29.65506639 | 79.73376355 |
| 32.19648242 | 77.88731671 |
| 34.81299038 | 76.14891299 |
| 37.5 | 74.52160212 |
| 40.25279737 | 73.00823896 |
| 43.06655313 | 71.61147847 |
| 45.93633102 | 70.33377103 |
| 48.85709645 | 69.17735819 |
| 51.82372542 | 68.14426868 |
| 54.83101345 | 67.23631489 |
| 57.87368474 | 66.45508968 |
| 60.94640141 | 65.8019636 |
| 64.04377286 | 65.27808243 |
| 67.16036525 | 64.88436525 |
| 70.29071104 | 64.62150277 |
| 73.42931851 | 64.48995614 |
| 76.57068149 | 64.48995614 |
| 79.70928896 | 64.62150277 |
| 82.83963475 | 64.88436525 |
| 85.95622714 | 65.27808243 |
| 89.05359859 | 65.8019636 |
| 92.12631526 | 66.45508968 |
| 95.16898655 | 67.23631489 |
| 98.17627458 | 68.14426868 |
| 101.1429035 | 69.17735819 |
| 104.063669 | 70.33377103 |
| 106.9334469 | 71.61147847 |
| 109.7472026 | 73.00823896 |
| 112.5 | 74.52160212 |
| 115.1870096 | 76.14891299 |
| 117.8035176 | 77.88731671 |
| 120.3449336 | 79.73376355 |
| 122.8067992 | 81.68501419 |
| 125.1847955 | 83.73764549 |
| 127.4747505 | 85.88805643 |
| 129.6726471 | 88.13247446 |
| 131.7746292 | 90.4669621 |
| 133.7770093 | 92.88742388 |
| 135.6762746 | 95.38961348 |
| 137.4690931 | 97.96914121 |
| 139.1523195 | 100.6214817 |
| 140.723001 | 103.3419818 |
| 142.178382 | 106.125869 |
| 143.5159093 | 108.9682592 |
| 144.7332364 | 111.8641659 |
| 145.8282278 | 114.8085089 |
| 146.7989623 | 117.7961226 |
| 147.6437371 | 120.8217659 |
| 148.3610701 | 123.8801306 |
| 148.9497028 | 126.9658514 |
| 149.4086026 | 130.0735149 |
| 149.7369644 | 133.1976692 |
| 149.9342123 | 136.3328334 |
| 150 | 139.4735074 |
| 75 | 139.4735074 |
| 74.10714286 | 136.0154866 |
| 73.21428572 | 129.0994449 |
| 72.32142858 | 124.4887504 |
| 71.42857144 | 119.3017191 |
| 70.5357143 | 114.1146879 |
| 69.64285716 | 108.9276566 |
| 68.75000002 | 103.7406253 |
| 67.85714288 | 98.55359407 |
| 66.96428574 | 93.3665628 |
| 66.0714286 | 88.17953153 |
| 65.17857146 | 82.99250026 |
| 64.28571432 | 77.805469 |
| 63.39285718 | 72.61843773 |
| 62.50000004 | 67.43140646 |
| 61.6071429 | 62.24437519 |
| 60.71428576 | 57.05734393 |
| 59.82142862 | 51.87031266 |
| 58.92857148 | 46.68328139 |
| 58.03571434 | 41.49625012 |
| 57.1428572 | 36.3092189 |
| 56.25000006 | 31.1221876 |
| 55.35714292 | 25.9351563 |
| 54.46428578 | 20.7481251 |
| 53.57142864 | 15.5610938 |
| 52.6785715 | 10.3740625 |
| 51.78571436 | 5.1870313 |
| 50.89285722 | 0 |
| 75.89285714 | 136.0154866 |
| 76.78571428 | 129.0994449 |
| 77.67857142 | 124.4887504 |
| 78.57142856 | 119.3017191 |
| 79.4642857 | 114.1146879 |
| 80.35714284 | 108.9276566 |
| 81.24999998 | 103.7406253 |
| 82.14285712 | 98.55359407 |
| 83.03571426 | 93.3665628 |
| 83.9285714 | 88.17953153 |
| 84.82142854 | 82.99250026 |
| 85.71428568 | 77.805469 |
| 86.60714282 | 72.61843773 |
| 87.49999996 | 67.43140646 |
| 88.3928571 | 62.24437519 |
| 89.28571424 | 57.05734393 |
| 90.17857138 | 51.87031266 |
| 91.07142852 | 46.68328139 |
| 91.96428566 | 41.49625012 |
| 92.8571428 | 36.3092189 |
| 93.74999994 | 31.1221876 |
| 94.64285708 | 25.9351563 |
| 95.53571422 | 20.7481251 |
| 96.42857136 | 15.5610938 |
| 97.3214285 | 10.3740625 |
| 98.21428564 | 5.1870313 |
| 99.10714278 | 0 |
| 73.2143 | 0 |
| 71.4286 | 0 |
| 69.6429 | 0 |
| 67.8572 | 0 |
| 66.0715 | 0 |
| 64.2858 | 0 |
| 62.5001 | 0 |
| 60.7144 | 0 |
| 58.9287 | 0 |
| 57.143 | 0 |
| 55.3573 | 0 |
| 53.5716 | 0 |
| 51.7859 | 0 |
| 50.0002 | 0 |
| 75 | 0 |
| 76.7857 | 0 |
| 78.5714 | 0 |
| 80.3571 | 0 |
| 82.1428 | 0 |
| 83.9285 | 0 |
| 85.7142 | 0 |
| 87.4999 | 0 |
| 89.2856 | 0 |
| 91.0713 | 0 |
| 92.857 | 0 |
| 94.6427 | 0 |
| 96.4284 | 0 |
| 98.2141 | 0 |
| 99.9998 | 0 |
| 87.42491396 | 140.84154 |
| 87.43348018 | 140.7613624 |
| 87.44152903 | 140.6811313 |
| 87.44906017 | 140.6008498 |
| 87.45607327 | 140.5205215 |
| 87.46256807 | 140.4401496 |
| 87.46854427 | 140.3597375 |
| 87.47400163 | 140.2792885 |
| 87.47893994 | 140.198806 |
| 87.48335897 | 140.1182933 |
| 87.48725855 | 140.0377538 |
| 87.49063851 | 139.9571908 |
| 87.49349872 | 139.8766076 |
| 87.49583905 | 139.7960077 |
| 87.49765941 | 139.7153944 |
| 87.49895972 | 139.634771 |
| 87.49973993 | 139.5541409 |
| 99.84982791 | 142.2095726 |
| 99.86696037 | 142.0492174 |
| 99.88305806 | 141.8887551 |
| 99.89812033 | 141.7281923 |
| 99.91214655 | 141.5675356 |
| 99.92513613 | 141.4067919 |
| 99.93708854 | 141.2459676 |
| 99.94800327 | 141.0850696 |
| 99.95787987 | 140.9241046 |
| 99.96671794 | 140.7630792 |
| 99.9745171 | 140.6020001 |
| 99.98127702 | 140.4408741 |
| 99.98699744 | 140.2797078 |
| 99.9916781 | 140.118508 |
| 99.99531882 | 139.9572813 |
| 99.99791944 | 139.7960345 |
| 99.99947985 | 139.6347743 |
| 112.2747419 | 143.5776052 |
| 112.3004405 | 143.3370724 |
| 112.3245871 | 143.096379 |
| 112.3471805 | 142.8555347 |
| 112.3682198 | 142.6145498 |
| 112.3877042 | 142.3734341 |
| 112.4056328 | 142.1321977 |
| 112.4220049 | 141.8908508 |
| 112.4368198 | 141.6494032 |
| 112.4500769 | 141.4078651 |
| 112.4617756 | 141.1662465 |
| 112.4719155 | 140.9245575 |
| 112.4804962 | 140.682808 |
| 112.4875171 | 140.4410083 |
| 112.4929782 | 140.1991683 |
| 112.4968792 | 139.9572981 |
| 112.4992198 | 139.7154078 |
| 124.6996558 | 144.9456377 |
| 124.7339207 | 144.6249275 |
| 124.7661161 | 144.3040028 |
| 124.7962407 | 143.9828772 |
| 124.8242931 | 143.6615639 |
| 124.8502723 | 143.3400763 |
| 124.8741771 | 143.0184279 |
| 124.8960065 | 142.6966319 |
| 124.9157597 | 142.3747018 |
| 124.9334359 | 142.052651 |
| 124.9490342 | 141.7304929 |
| 124.962554 | 141.4082408 |
| 124.9739949 | 141.0859083 |
| 124.9833562 | 140.7635086 |
| 124.9906376 | 140.4410553 |
| 124.9958389 | 140.1185617 |
| 124.9989597 | 139.7960413 |
| 137.1245698 | 146.3136703 |
| 137.1674009 | 145.9127825 |
| 137.2076452 | 145.5116267 |
| 137.2453008 | 145.1102196 |
| 137.2803664 | 144.708578 |
| 137.3128403 | 144.3067185 |
| 137.3427213 | 143.904658 |
| 137.3700082 | 143.502413 |
| 137.3946997 | 143.1000004 |
| 137.4167948 | 142.6974369 |
| 137.4362927 | 142.2947392 |
| 137.4531926 | 141.8919242 |
| 137.4674936 | 141.4890085 |
| 137.4791952 | 141.0860089 |
| 137.488297 | 140.6829422 |
| 137.4947986 | 140.2798253 |
| 137.4986996 | 139.8766747 |
| 75 | 139.4735074 |

### b. Dragon

|  |  |
| --- | --- |
| 569 | 35 |
| 507 | 44 |
| 465 | 60 |
| 484 | 52 |
| 518 | 43 |
| 555 | 38 |
| 598 | 38 |
| 624 | 41 |
| 633 | 40 |
| 661 | 47 |
| 683 | 51 |
| 709 | 61 |
| 728 | 70 |
| 755 | 83 |
| 779 | 96 |
| 803 | 109 |
| 826 | 130 |
| 843 | 140 |
| 872 | 171 |
| 877 | 173 |
| 919 | 227 |
| 950 | 282 |
| 968 | 343 |
| 976 | 404 |
| 976 | 462 |
| 968 | 524 |
| 954 | 585 |
| 932 | 646 |
| 901 | 704 |
| 860 | 759 |
| 807 | 810 |
| 749 | 850 |
| 689 | 880 |
| 628 | 901 |
| 584 | 923 |
| 605 | 946 |
| 619 | 961 |
| 559 | 965 |
| 497 | 951 |
| 435 | 919 |
| 398 | 935 |
| 391 | 980 |
| 336 | 937 |
| 318 | 890 |
| 285 | 885 |
| 226 | 915 |
| 242 | 912 |
| 228 | 905 |
| 236 | 884 |
| 233 | 839 |
| 200 | 814 |
| 139 | 821 |
| 158 | 824 |
| 195 | 817 |
| 141 | 813 |
| 163 | 796 |
| 178 | 778 |
| 191 | 757 |
| 197 | 733 |
| 192 | 686 |
| 143 | 693 |
| 83 | 685 |
| 96 | 689 |
| 90 | 674 |
| 113 | 668 |
| 134 | 655 |
| 152 | 641 |
| 168 | 619 |
| 175 | 597 |
| 181 | 575 |
| 188 | 552 |
| 160 | 540 |
| 98 | 521 |
| 84 | 508 |
| 100 | 503 |
| 126 | 507 |
| 153 | 507 |
| 172 | 496 |
| 195 | 482 |
| 211 | 465 |
| 225 | 446 |
| 235 | 426 |
| 182 | 400 |
| 143 | 350 |
| 160 | 376 |
| 200 | 407 |
| 149 | 348 |
| 171 | 368 |
| 194 | 379 |
| 216 | 382 |
| 226 | 384 |
| 257 | 382 |
| 280 | 372 |
| 303 | 358 |
| 274 | 329 |
| 237 | 280 |
| 235 | 267 |
| 263 | 319 |
| 297 | 339 |
| 248 | 279 |
| 234 | 257 |
| 275 | 298 |
| 297 | 313 |
| 323 | 318 |
| 348 | 321 |
| 372 | 318 |
| 392 | 313 |
| 362 | 279 |
| 350 | 261 |
| 387 | 297 |
| 351 | 252 |
| 371 | 259 |
| 394 | 268 |
| 418 | 278 |
| 442 | 283 |
| 468 | 287 |
| 487 | 288 |
| 508 | 294 |
| 524 | 295 |
| 556 | 290 |
| 577 | 284 |
| 562 | 266 |
| 579 | 261 |
| 604 | 257 |
| 624 | 256 |
| 637 | 256 |
| 671 | 256 |
| 699 | 274 |
| 703 | 314 |
| 691 | 335 |
| 665 | 326 |
| 635 | 321 |
| 633 | 364 |
| 596 | 360 |
| 560 | 370 |
| 533 | 393 |
| 531 | 429 |
| 537 | 392 |
| 550 | 430 |
| 571 | 418 |
| 598 | 409 |
| 626 | 400 |
| 647 | 394 |
| 686 | 396 |
| 695 | 396 |
| 718 | 398 |
| 736 | 395 |
| 717 | 412 |
| 658 | 407 |
| 598 | 427 |
| 567 | 460 |
| 578 | 464 |
| 600 | 468 |
| 617 | 472 |
| 617 | 448 |
| 631 | 450 |
| 640 | 464 |
| 648 | 472 |
| 660 | 460 |
| 662 | 449 |
| 671 | 463 |
| 680 | 445 |
| 685 | 439 |
| 693 | 464 |
| 692 | 433 |
| 700 | 495 |
| 687 | 513 |
| 706 | 481 |
| 649 | 502 |
| 589 | 512 |
| 528 | 529 |
| 467 | 509 |
| 407 | 528 |
| 376 | 588 |
| 393 | 643 |
| 385 | 623 |
| 379 | 583 |
| 392 | 549 |
| 402 | 650 |
| 420 | 662 |
| 448 | 672 |
| 480 | 675 |
| 490 | 673 |
| 522 | 666 |
| 544 | 659 |
| 565 | 650 |
| 593 | 639 |
| 614 | 621 |
| 636 | 606 |
| 649 | 601 |
| 618 | 660 |
| 610 | 673 |
| 630 | 645 |
| 647 | 614 |
| 653 | 592 |
| 616 | 686 |
| 636 | 685 |
| 658 | 684 |
| 689 | 677 |
| 707 | 669 |
| 732 | 659 |
| 753 | 649 |
| 774 | 633 |
| 793 | 619 |
| 812 | 601 |
| 825 | 581 |
| 840 | 564 |
| 850 | 540 |
| 859 | 522 |
| 871 | 494 |
| 874 | 474 |
| 877 | 451 |
| 882 | 439 |
| 883 | 390 |
| 879 | 371 |
| 870 | 328 |
| 852 | 284 |
| 826 | 239 |
| 781 | 185 |
| 722 | 136 |
| 660 | 104 |
| 599 | 82 |
| 538 | 70 |
| 476 | 59 |
| 521 | 64 |
| 551 | 73 |
| 588 | 82 |
| 626 | 91 |
| 667 | 110 |
| 692 | 122 |
| 732 | 146 |
| 758 | 162 |
| 800 | 203 |
| 833 | 245 |
| 860 | 296 |
| 878 | 350 |
| 862 | 299 |
| 783 | 184 |
| 420 | 522 |
| 454 | 509 |
| 491 | 514 |
| 524 | 531 |
| 560 | 521 |
| 599 | 512 |
| 637 | 503 |
| 669 | 505 |
| 592 | 434 |
| 632 | 414 |
| 671 | 407 |
| 713 | 413 |
| 739 | 422 |
| 758 | 430 |
| 744 | 398 |
| 751 | 393 |
| 774 | 380 |
| 550 | 381 |
| 564 | 382 |
| 588 | 371 |
| 599 | 361 |
| 609 | 347 |
| 639 | 370 |
| 632 | 329 |
| 660 | 342 |
| 670 | 336 |
| 662 | 348 |
| 696 | 342 |
| 703 | 343 |
| 703 | 304 |
| 708 | 281 |
| 532 | 296 |
| 214 | 415 |
| 94 | 518 |
| 133 | 541 |
| 165 | 543 |
| 111 | 696 |
| 157 | 691 |
| 194 | 685 |
| 198 | 732 |
| 237 | 849 |
| 263 | 902 |
| 295 | 881 |
| 318 | 888 |
| 340 | 943 |
| 354 | 958 |
| 387 | 979 |
| 403 | 968 |
| 402 | 927 |
| 424 | 915 |
| 384 | 978 |
| 483 | 949 |
| 525 | 962 |
| 559 | 966 |
| 604 | 967 |
| 634 | 960 |
| 585 | 914 |
| 623 | 905 |
| 661 | 893 |
| 607 | 907 |
| 726 | 866 |
| 745 | 853 |
| 792 | 825 |
| 821 | 802 |
| 851 | 773 |
| 874 | 744 |
| 896 | 715 |
| 915 | 684 |
| 932 | 651 |
| 944 | 618 |
| 957 | 584 |
| 966 | 549 |
| 970 | 515 |
| 976 | 479 |
| 823 | 797 |
| 977 | 390 |
| 971 | 353 |
| 965 | 315 |
| 953 | 281 |
| 935 | 250 |
| 914 | 215 |
| 893 | 189 |

### c. Interstellar System

#### i. Star (unit radius is used)

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 1 |
| 0.214382638 | 0.103241237 | 0.971278195 |
| 0.052948139 | 0.231980955 | 0.971278195 |
| -0.148357389 | 0.186034281 | 0.971278195 |
| -0.237946777 | 2.91E-17 | 0.971278195 |
| -0.148357389 | -0.186034281 | 0.971278195 |
| 0.052948139 | -0.231980955 | 0.971278195 |
| 0.214382638 | -0.103241237 | 0.971278195 |
| 0.442040414 | 0.049805993 | 0.895611319 |
| 0.357915232 | 0.264153462 | 0.895611319 |
| 0.177886952 | 0.407721225 | 0.895611319 |
| -0.049805993 | 0.442040414 | 0.895611319 |
| -0.264153462 | 0.357915232 | 0.895611319 |
| -0.407721225 | 0.177886952 | 0.895611319 |
| -0.442040414 | -0.049805993 | 0.895611319 |
| -0.357915232 | -0.264153462 | 0.895611319 |
| -0.177886952 | -0.407721225 | 0.895611319 |
| 0.049805993 | -0.442040414 | 0.895611319 |
| 0.264153462 | -0.357915232 | 0.895611319 |
| 0.407721225 | -0.177886952 | 0.895611319 |
| 0.627655716 | 0.015652756 | 0.778333665 |
| 0.584449886 | 0.229379677 | 0.778333665 |
| 0.470750775 | 0.415440024 | 0.778333665 |
| 0.300272173 | 0.551392173 | 0.778333665 |
| 0.093576315 | 0.620838288 | 0.778333665 |
| -0.124406227 | 0.615402142 | 0.778333665 |
| -0.327383542 | 0.535739416 | 0.778333665 |
| -0.490873571 | 0.39145861 | 0.778333665 |
| -0.595157002 | 0.199962118 | 0.778333665 |
| -0.627655716 | -0.015652756 | 0.778333665 |
| -0.584449886 | -0.229379677 | 0.778333665 |
| -0.470750775 | -0.415440024 | 0.778333665 |
| -0.300272173 | -0.551392173 | 0.778333665 |
| -0.093576315 | -0.620838288 | 0.778333665 |
| 0.124406227 | -0.615402142 | 0.778333665 |
| 0.327383542 | -0.535739416 | 0.778333665 |
| 0.490873571 | -0.39145861 | 0.778333665 |
| 0.595157002 | -0.199962118 | 0.778333665 |
| 0.783664361 | -0.01776606 | 0.62093038 |
| 0.756925725 | 0.203737354 | 0.62093038 |
| 0.668865469 | 0.40873518 | 0.62093038 |
| 0.526617711 | 0.580619712 | 0.62093038 |
| 0.341706517 | 0.705465888 | 0.62093038 |
| 0.129112294 | 0.773159414 | 0.62093038 |
| -0.09394184 | 0.778216161 | 0.62093038 |
| -0.309385364 | 0.720226464 | 0.62093038 |
| -0.499764326 | 0.603888302 | 0.62093038 |
| -0.649655355 | 0.438626701 | 0.62093038 |
| -0.74691517 | 0.237830174 | 0.62093038 |
| -0.783664361 | 0.01776606 | 0.62093038 |
| -0.756925725 | -0.203737354 | 0.62093038 |
| -0.668865469 | -0.40873518 | 0.62093038 |
| -0.526617711 | -0.580619712 | 0.62093038 |
| -0.341706517 | -0.705465888 | 0.62093038 |
| -0.129112294 | -0.773159414 | 0.62093038 |
| 0.09394184 | -0.778216161 | 0.62093038 |
| 0.309385364 | -0.720226464 | 0.62093038 |
| 0.499764326 | -0.603888302 | 0.62093038 |
| 0.649655355 | -0.438626701 | 0.62093038 |
| 0.74691517 | -0.237830174 | 0.62093038 |
| 0.900268578 | -0.050124983 | 0.43243956 |
| 0.886104103 | 0.166779931 | 0.43243956 |
| 0.820442478 | 0.373992201 | 0.43243956 |
| 0.707099719 | 0.559469404 | 0.43243956 |
| 0.552662894 | 0.712432279 | 0.43243956 |
| 0.366107311 | 0.82399118 | 0.43243956 |
| 0.158274901 | 0.887662708 | 0.43243956 |
| -0.05875587 | 0.899746506 | 0.43243956 |
| -0.272371964 | 0.859540307 | 0.43243956 |
| -0.47015879 | 0.769380751 | 0.43243956 |
| -0.640621695 | 0.634507581 | 0.43243956 |
| -0.773853996 | 0.462759138 | 0.43243956 |
| -0.862112716 | 0.264116815 | 0.43243956 |
| -0.900268578 | 0.050124983 | 0.43243956 |
| -0.886104103 | -0.166779931 | 0.43243956 |
| -0.820442478 | -0.373992201 | 0.43243956 |
| -0.707099719 | -0.559469404 | 0.43243956 |
| -0.552662894 | -0.712432279 | 0.43243956 |
| -0.366107311 | -0.82399118 | 0.43243956 |
| -0.158274901 | -0.887662708 | 0.43243956 |
| 0.05875587 | -0.899746506 | 0.43243956 |
| 0.272371964 | -0.859540307 | 0.43243956 |
| 0.47015879 | -0.769380751 | 0.43243956 |
| 0.640621695 | -0.634507581 | 0.43243956 |
| 0.773853996 | -0.462759138 | 0.43243956 |
| 0.862112716 | -0.264116815 | 0.43243956 |
| 0.973455922 | -0.062631854 | 0.220138182 |
| 0.962986248 | 0.155552778 | 0.220138182 |
| 0.904228423 | 0.365937344 | 0.220138182 |
| 0.800128809 | 0.557972284 | 0.220138182 |
| 0.655907395 | 0.722028164 | 0.220138182 |
| 0.478796046 | 0.849878537 | 0.220138182 |
| 0.277675864 | 0.935112451 | 0.220138182 |
| 0.062631854 | 0.973455922 | 0.220138182 |
| -0.155552778 | 0.962986248 | 0.220138182 |
| -0.365937344 | 0.904228423 | 0.220138182 |
| -0.557972284 | 0.800128809 | 0.220138182 |
| -0.722028164 | 0.655907395 | 0.220138182 |
| -0.849878537 | 0.478796046 | 0.220138182 |
| -0.935112451 | 0.277675864 | 0.220138182 |
| -0.973455922 | 0.062631854 | 0.220138182 |
| -0.962986248 | -0.155552778 | 0.220138182 |
| -0.904228423 | -0.365937344 | 0.220138182 |
| -0.800128809 | -0.557972284 | 0.220138182 |
| -0.655907395 | -0.722028164 | 0.220138182 |
| -0.478796046 | -0.849878537 | 0.220138182 |
| -0.277675864 | -0.935112451 | 0.220138182 |
| -0.062631854 | -0.973455922 | 0.220138182 |
| 0.155552778 | -0.962986248 | 0.220138182 |
| 0.365937344 | -0.904228423 | 0.220138182 |
| 0.557972284 | -0.800128809 | 0.220138182 |
| 0.722028164 | -0.655907395 | 0.220138182 |
| 0.849878537 | -0.478796046 | 0.220138182 |
| 0.935112451 | -0.277675864 | 0.220138182 |
| 0.998850686 | 0.047930235 | 6.12E-17 |
| 0.963141933 | 0.268993712 | 6.12E-17 |
| 0.879137222 | 0.47656872 | 6.12E-17 |
| 0.7510489 | 0.660246583 | 6.12E-17 |
| 0.585299849 | 0.810816925 | 6.12E-17 |
| 0.390201421 | 0.920729521 | 6.12E-17 |
| 0.175536664 | 0.984472894 | 6.12E-17 |
| -0.047930235 | 0.998850686 | 6.12E-17 |
| -0.268993712 | 0.963141933 | 6.12E-17 |
| -0.47656872 | 0.879137222 | 6.12E-17 |
| -0.660246583 | 0.7510489 | 6.12E-17 |
| -0.810816925 | 0.585299849 | 6.12E-17 |
| -0.920729521 | 0.390201421 | 6.12E-17 |
| -0.984472894 | 0.175536664 | 6.12E-17 |
| -0.998850686 | -0.047930235 | 6.12E-17 |
| -0.963141933 | -0.268993712 | 6.12E-17 |
| -0.879137222 | -0.47656872 | 6.12E-17 |
| -0.7510489 | -0.660246583 | 6.12E-17 |
| -0.585299849 | -0.810816925 | 6.12E-17 |
| -0.390201421 | -0.920729521 | 6.12E-17 |
| -0.175536664 | -0.984472894 | 6.12E-17 |
| 0.047930235 | -0.998850686 | 6.12E-17 |
| 0.268993712 | -0.963141933 | 6.12E-17 |
| 0.47656872 | -0.879137222 | 6.12E-17 |
| 0.660246583 | -0.7510489 | 6.12E-17 |
| 0.810816925 | -0.585299849 | 6.12E-17 |
| 0.920729521 | -0.390201421 | 6.12E-17 |
| 0.984472894 | -0.175536664 | 6.12E-17 |
| 0.962986248 | 0.155552778 | -0.220138182 |
| 0.904228423 | 0.365937344 | -0.220138182 |
| 0.800128809 | 0.557972284 | -0.220138182 |
| 0.655907395 | 0.722028164 | -0.220138182 |
| 0.478796046 | 0.849878537 | -0.220138182 |
| 0.277675864 | 0.935112451 | -0.220138182 |
| 0.062631854 | 0.973455922 | -0.220138182 |
| -0.155552778 | 0.962986248 | -0.220138182 |
| -0.365937344 | 0.904228423 | -0.220138182 |
| -0.557972284 | 0.800128809 | -0.220138182 |
| -0.722028164 | 0.655907395 | -0.220138182 |
| -0.849878537 | 0.478796046 | -0.220138182 |
| -0.935112451 | 0.277675864 | -0.220138182 |
| -0.973455922 | 0.062631854 | -0.220138182 |
| -0.962986248 | -0.155552778 | -0.220138182 |
| -0.904228423 | -0.365937344 | -0.220138182 |
| -0.800128809 | -0.557972284 | -0.220138182 |
| -0.655907395 | -0.722028164 | -0.220138182 |
| -0.478796046 | -0.849878537 | -0.220138182 |
| -0.277675864 | -0.935112451 | -0.220138182 |
| -0.062631854 | -0.973455922 | -0.220138182 |
| 0.155552778 | -0.962986248 | -0.220138182 |
| 0.365937344 | -0.904228423 | -0.220138182 |
| 0.557972284 | -0.800128809 | -0.220138182 |
| 0.722028164 | -0.655907395 | -0.220138182 |
| 0.849878537 | -0.478796046 | -0.220138182 |
| 0.935112451 | -0.277675864 | -0.220138182 |
| 0.973455922 | -0.062631854 | -0.220138182 |
| 0.886104103 | 0.166779931 | -0.43243956 |
| 0.820442478 | 0.373992201 | -0.43243956 |
| 0.707099719 | 0.559469404 | -0.43243956 |
| 0.552662894 | 0.712432279 | -0.43243956 |
| 0.366107311 | 0.82399118 | -0.43243956 |
| 0.158274901 | 0.887662708 | -0.43243956 |
| -0.05875587 | 0.899746506 | -0.43243956 |
| -0.272371964 | 0.859540307 | -0.43243956 |
| -0.47015879 | 0.769380751 | -0.43243956 |
| -0.640621695 | 0.634507581 | -0.43243956 |
| -0.773853996 | 0.462759138 | -0.43243956 |
| -0.862112716 | 0.264116815 | -0.43243956 |
| -0.900268578 | 0.050124983 | -0.43243956 |
| -0.886104103 | -0.166779931 | -0.43243956 |
| -0.820442478 | -0.373992201 | -0.43243956 |
| -0.707099719 | -0.559469404 | -0.43243956 |
| -0.552662894 | -0.712432279 | -0.43243956 |
| -0.366107311 | -0.82399118 | -0.43243956 |
| -0.158274901 | -0.887662708 | -0.43243956 |
| 0.05875587 | -0.899746506 | -0.43243956 |
| 0.272371964 | -0.859540307 | -0.43243956 |
| 0.47015879 | -0.769380751 | -0.43243956 |
| 0.640621695 | -0.634507581 | -0.43243956 |
| 0.773853996 | -0.462759138 | -0.43243956 |
| 0.862112716 | -0.264116815 | -0.43243956 |
| 0.900268578 | -0.050124983 | -0.43243956 |
| 0.761218652 | 0.187060491 | -0.62093038 |
| 0.677682918 | 0.393943303 | -0.62093038 |
| 0.539245344 | 0.568911173 | -0.62093038 |
| 0.357121319 | 0.697789242 | -0.62093038 |
| 0.146065449 | 0.770136577 | -0.62093038 |
| -0.076823775 | 0.780092027 | -0.62093038 |
| -0.293489194 | 0.72684906 | -0.62093038 |
| -0.486377863 | 0.614721105 | -0.62093038 |
| -0.639863091 | 0.452792102 | -0.62093038 |
| -0.741510417 | 0.254180576 | -0.62093038 |
| -0.783084978 | 0.034976851 | -0.62093038 |
| -0.761218652 | -0.187060491 | -0.62093038 |
| -0.677682918 | -0.393943303 | -0.62093038 |
| -0.539245344 | -0.568911173 | -0.62093038 |
| -0.357121319 | -0.697789242 | -0.62093038 |
| -0.146065449 | -0.770136577 | -0.62093038 |
| 0.076823775 | -0.780092027 | -0.62093038 |
| 0.293489194 | -0.72684906 | -0.62093038 |
| 0.486377863 | -0.614721105 | -0.62093038 |
| 0.639863091 | -0.452792102 | -0.62093038 |
| 0.741510417 | -0.254180576 | -0.62093038 |
| 0.783084978 | -0.034976851 | -0.62093038 |
| 0.595919654 | 0.197677697 | -0.778333665 |
| 0.492371547 | 0.389572799 | -0.778333665 |
| 0.329436165 | 0.534479671 | -0.778333665 |
| 0.12676592 | 0.614920408 | -0.778333665 |
| -0.091194166 | 0.621192667 | -0.778333665 |
| -0.29815489 | 0.552539924 | -0.778333665 |
| -0.469153734 | 0.417242711 | -0.778333665 |
| -0.583565714 | 0.231619869 | -0.778333665 |
| -0.627591056 | 0.018060253 | -0.778333665 |
| -0.595919654 | -0.197677697 | -0.778333665 |
| -0.492371547 | -0.389572799 | -0.778333665 |
| -0.329436165 | -0.534479671 | -0.778333665 |
| -0.12676592 | -0.614920408 | -0.778333665 |
| 0.091194166 | -0.621192667 | -0.778333665 |
| 0.29815489 | -0.552539924 | -0.778333665 |
| 0.469153734 | -0.417242711 | -0.778333665 |
| 0.583565714 | -0.231619869 | -0.778333665 |
| 0.627591056 | -0.018060253 | -0.778333665 |
| 0.371578118 | 0.244560968 | -0.895611319 |
| 0.199515606 | 0.39758507 | -0.895611319 |
| -0.026006952 | 0.444076573 | -0.895611319 |
| -0.244560968 | 0.371578118 | -0.895611319 |
| -0.39758507 | 0.199515606 | -0.895611319 |
| -0.444076573 | -0.026006952 | -0.895611319 |
| -0.371578118 | -0.244560968 | -0.895611319 |
| -0.199515606 | -0.39758507 | -0.895611319 |
| 0.026006952 | -0.444076573 | -0.895611319 |
| 0.244560968 | -0.371578118 | -0.895611319 |
| 0.39758507 | -0.199515606 | -0.895611319 |
| 0.444076573 | 0.026006952 | -0.895611319 |
| 0.129862906 | 0.19938479 | -0.971278195 |
| -0.074917108 | 0.225845291 | -0.971278195 |
| -0.223283012 | 0.082239682 | -0.971278195 |
| -0.203512254 | -0.123294085 | -0.971278195 |
| -0.030492618 | -0.235984891 | -0.971278195 |
| 0.165488581 | -0.170974262 | -0.971278195 |
| 0.236853503 | 0.022783474 | -0.971278195 |
| 1.22E-16 | 0 | -1 |

#### ii. Planet (unit radius is used)

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 1 |
| 0.38632628 | 0.223045582 | 0.894987527 |
| 2.73E-17 | 0.446091163 | 0.894987527 |
| -0.38632628 | 0.223045582 | 0.894987527 |
| -0.38632628 | -0.223045582 | 0.894987527 |
| -8.19E-17 | -0.446091163 | 0.894987527 |
| 0.38632628 | -0.223045582 | 0.894987527 |
| 0.77733281 | 0.074226284 | 0.624695255 |
| 0.613804214 | 0.482700969 | 0.624695255 |
| 0.255397117 | 0.737921508 | 0.624695255 |
| -0.18409676 | 0.758857181 | 0.624695255 |
| -0.565141217 | 0.538861062 | 0.624695255 |
| -0.76675733 | 0.147780363 | 0.624695255 |
| -0.724933409 | -0.290219558 | 0.624695255 |
| -0.452948252 | -0.636076819 | 0.624695255 |
| -0.037155226 | -0.779984184 | 0.624695255 |
| 0.390434322 | -0.676252083 | 0.624695255 |
| 0.694063731 | -0.357814723 | 0.624695255 |
| 0.974456411 | -0.00662639 | 0.22447894 |
| 0.880829972 | 0.41683062 | 0.22447894 |
| 0.612744355 | 0.757729213 | 0.22447894 |
| 0.223297203 | 0.948550243 | 0.22447894 |
| -0.210376698 | 0.951499265 | 0.22447894 |
| -0.602382914 | 0.765992187 | 0.22447894 |
| -0.875079806 | 0.428770962 | 0.22447894 |
| -0.974456411 | 0.00662639 | 0.22447894 |
| -0.880829972 | -0.41683062 | 0.22447894 |
| -0.612744355 | -0.757729213 | 0.22447894 |
| -0.223297203 | -0.948550243 | 0.22447894 |
| 0.210376698 | -0.951499265 | 0.22447894 |
| 0.602382914 | -0.765992187 | 0.22447894 |
| 0.875079806 | -0.428770962 | 0.22447894 |
| 0.951499265 | 0.210376698 | -0.22447894 |
| 0.765992187 | 0.602382914 | -0.22447894 |
| 0.428770962 | 0.875079806 | -0.22447894 |
| 0.00662639 | 0.974456411 | -0.22447894 |
| -0.41683062 | 0.880829972 | -0.22447894 |
| -0.757729213 | 0.612744355 | -0.22447894 |
| -0.948550243 | 0.223297203 | -0.22447894 |
| -0.951499265 | -0.210376698 | -0.22447894 |
| -0.765992187 | -0.602382914 | -0.22447894 |
| -0.428770962 | -0.875079806 | -0.22447894 |
| -0.00662639 | -0.974456411 | -0.22447894 |
| 0.41683062 | -0.880829972 | -0.22447894 |
| 0.757729213 | -0.612744355 | -0.22447894 |
| 0.948550243 | -0.223297203 | -0.22447894 |
| 0.730702625 | 0.27537159 | -0.624695255 |
| 0.465829043 | 0.626704987 | -0.624695255 |
| 0.053058032 | 0.77906398 | -0.624695255 |
| -0.37655853 | 0.684075663 | -0.624695255 |
| -0.686620419 | 0.371898156 | -0.624695255 |
| -0.778685177 | -0.058354387 | -0.624695255 |
| -0.623522892 | -0.470079825 | -0.624695255 |
| -0.270396496 | -0.732558239 | -0.624695255 |
| 0.168578878 | -0.762454589 | -0.624695255 |
| 0.554031649 | -0.550276994 | -0.624695255 |
| 0.763583286 | -0.163390342 | -0.624695255 |
| 0.282850659 | 0.344953375 | -0.894987527 |
| -0.157313056 | 0.417432543 | -0.894987527 |
| -0.440163715 | 0.072479169 | -0.894987527 |
| -0.282850659 | -0.344953375 | -0.894987527 |
| 0.157313056 | -0.417432543 | -0.894987527 |
| 0.440163715 | -0.072479169 | -0.894987527 |
| 1.22E-16 | 0 | -1 |

## C. Table of used in the test of Navigation System

|  |  |
| --- | --- |
|  |  |
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 8 |
| 11 | 7 |
| 12 | 6 |
| 13 | 5 |
| 14 | 4 |
| 15 | 3 |
| 16 | 2 |
| 17 | 1 |
| 18 | 0 |