**Abstract**

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have recently been applied in choreographed light shows and thus evoked a new boom in technology. In 2016, Intel® used a cluster of its Shooting Star™ drones that were controlled by only one pilot and a single laptop to perform an aerial light show, resulting in a twinkling digital galaxy floating above the skyline. Yet Intel is not the only company interested in aerial shows using drones since this newly formed technology has been brought under the spotlight and remained to be a heating topic.

Other related researches and finding also focused on the control and orientation of drones in application.

In response to the Mayor’s asking of an outdoor aerial light show using drones, after a careful investigation, we would like to present our mathematical model, conclusions and recommendations for the aerial light show. In brief, we address the problem of optimizing the flight paths for each drone through mapping the locus function with three-dimensional system of coordinates, taking the number of drones required, safety concerns, launch area required and air space required into consideration.

We formulate an optimization model to account for the optimal flight path for each drone in order to form the shape of our three-dimensional design of display——a Ferris wheel, a dragon and a map of China——a mixture of both static and dynamic images. Based on the historical data from all the aerial light shows using drones that have already taken place so far, we determine the initial conditions, prerequisites and several basic parameters of drones for our model. To solve the model, the shortest path and its length of every two nodes in the incomplete undirected graph are calculated with Euclidean distance. In search of the optimal model, we run through the particular path for each drone and group those paths that share similar characteristics to minimize the number of functions controlling the drones using clustering analysis. Then we use the binary integer programming to optimize the total flight distance in order to find the most suitable solution to our model. To ground this model in reality, we animate the image and the whole display process through computer simulation to adjust and update colliding paths.

We show that this strategy is not optimal but can be improved by optimizing the coordinates and functions of the flight path for each drone. If the Mayor were to adopt this model and strategy for the aerial light show on the annual festival, the cost would be approximately......We modify the model to reflect the flight paths and generalize the model to other fields including but not limited to the combat drone operation as well as control and orientation system. We conclude with a series of recommendations for how best to design and distribute the particular path for each drone. The simulation examples validate the feasibility of our strategy.

Our suggested solution, which is easy to implement, includes a detailed aerial display program and flight paths for each drone. We firmly believe that our algorithm is broad and flexible enough to accommodate various local conditions, safety concerns and other unexpected incidents. Since our model is based on the control and orientation of UAVs, our strategy may also contribute to other technologies related to drones.

**Key words**: Euclidean distance, the Roberts edge detection operator, binary integer programming

**Letter to the Mayor**

Dear Mayor,

Our team has carefully planned the light show on the night of the annual festival and succeeded in creating three possible sky displays including the pattern of a Ferris wheel, a dragon and a map of China. The whole performance will include 477 drones, and all the people in this city can enjoy this well-organized fantastic light show.

All the drones that will be used in this show should be ready about 20 minutes before the show starts and the place for the light show needs to be spacious, especially to avoid the city center in order to make sure the traffic won't be blocked, and after the careful examination, the square in the north of the city is a suitable place. Our team also suggests that the show should begin at 8:00 pm since it would be completely dark at that time and the audience can fully enjoy this visual feast. Therefore, the drones should be in place at about 7:40, and before that, we want to make sure that all the roads around the place should be cleared so that the potential accident can be avoided. The apron will cover approximately the space of 100m\*70m on the ground and the whole performance will occupy the total space of 200m\*70m\*200m. The total time of the light show will be no longer than 20 minutes according to limited time a drone can constantly fly in the sky.

The brightest shining point of our display is the third image designed. The broad territory of China fully unfolds the prosperity of our motherland and our patriotism, echoing the theme --- what ethnic is what worldwide. Another great brilliance that lies thoroughly in our model is how we improve our algorithm to avoid crashes between any two drones during the overall display. Remarkable progress has been made through our adjustment for each drone since the total number of crashed drones is lowered down from 61 to 0. Since our model involves 477 drones, it’s clearly a great challenge to ensure that any two flying tracks do not possess intersections. Yet we manage to conquer the challenge and successfully present a structural model in response to the task.

Our model effectively achieves all of the goals we set initially. It is definitely a feasible solution and could handle large quantities of data. Admittedly, there remain several flaws in our robust and effective model. But we firmly believe that with a larger number of drones, more adjustment of the flight paths, and more factors being taken into consideration, the model can be improved to a higher and more realistic level. In addition, our model generalizes the algorithm used in the control and orientation of UAVs, flexible and broad enough to accommodate various local conditions, safety concerns and other unexpected incidents. We proudly declare that the application of our model maintains a vast potential for future development including combat drone control and orientation system.

Attached on the next page is our designed images for the aerial light show. We express our sincerest gratitude for your trust in us to organize this festive event, which we hope will develop into a worldwide carnival in which everyone enjoys and appreciates our drone light show.

Best Wishes

**Background**

Pilotless aircraft, often referred to as "unmanned aerial vehicle", with the abbreviation of “UAV”, is a pilotless aircraft operated by radio remote control equipment, its own program control device, or operated entirely or intermittently by the on-board computers. Since the birth of the aircraft in the early twentieth century, people have proposed the idea of unmanned aircrafts because of the safety problem of the aircrafts. In 1930s, the British Ferrell company remade a double-fixed wing aircraft into an unmanned drone, which was the first time UVA had entered the history of aviation. Since then, UVA has been used in a lot of domains including aerial photography, news report, wildlife protection and also performances despite the military use. In a recent UVA show performed by YiHang GHOSTDRONE 2.0, engineers designed a set of intelligent and efficient unmanned aerial vehicle remote control system, which realized the function of using only one computer as a ground control station to autonomously control, monitor the flight task of thousands of UAVs, and set the color change of aircrafts’ lights. They presented a large-scale visual feast in the form of fancy lighting show in only 15 minutes.

In our task, the main goal is to organize a beautifully performed light show by using approximately 480 drones and create 3 possible displays. The main challenge is how to minimize the total time the whole performance would cost due to the limited time a drone could fly constantly in the sky and how to reduce the total distance that drones would move from one displayed pattern to another.

**Restatement of the problem**

We are asked to organize an outdoor aerial light show with the utilization of drones. The main challenge is to depict three possible images and the overall show process as well as determine the optimal flight paths of each drone device that would simulate our image on display. To be more specific, the required launch area, required number of drones, required air space and the transitioning flight paths between two images for our three images——the Ferris wheel, the dragon and the map of China. The great barrier lies upon us is how to avoid crashes of the flight paths during the display and the transitions between two images. The effect of changes in the total distance, average distance, and time required based on the consideration of safety concerns, limited space and limited flying speed will also have to be investigated to explore the advantages we may achieve in future events.

**Assumptions**

We make the following assumptions about the initial conditions and basic parameters of the drones in this paper. The functions and performance parameters of each drone may have a slight difference, but in order to simplify the model, we assume all of them to be the same. Through the reference of the Shooting Star™ drones used in the Intel® light show, some key parameters of the drones are assumed and set as following in table 1:

Table 1: Basic Parameters of the drones

|  |  |
| --- | --- |
| Size | 384mm×384mm×93 mm |
| Propeller diameter | 6 inches (15 cm) |
| Maximum take-off weight | 280 grams |
| Maximum time of flight | 20 minutes |
| Maximum distance of flight | 1.5 km |
| Maximum airspeed of flight | 10 m/s |
| Maximum airspeed of light show mode | 3 m/s |

Each drone can automatically measure its horizontal position and vertical height with the utilization of GPS and barometer. Each drone can determine the target location of the next moment according to the preset track. With the flight control system for navigation, the drones can complete the overall effect of structural formation. The flying path of each drone may vary from the preset track, but we approximated the flight path to be a linear function to simplify the model.

Considering the convenience of formation, the take-off site and landing position of each drone is fixed to form a rectangle. To minimize the required launch space, we shape all the drones to remain a regular triangle distance with each other instead of simply fitting them into the points in a rectangle. To prevent mutual interference and ensure safety, the distance between each other during the whole display is further than the minimum safety distance by our definition.

Meteorological conditions can be complicated and unpredictable in reality; therefore, we ignore the factors related to meteorological conditions such as the wind speed and non-ideal weather. Other unexpected incidents like breakdowns of drones and defaults are assumed to be impossible as well. Additional assumptions are made to simplify analysis for individual sections. These assumptions will be discussed at the appropriate locations.

**Definition**

1. We define the flight from the take-off apron to the Ferris wheel as the **stage Ⅰ flying process**. Similarly, the flight from the Ferris wheel to the dragon, the flight from the dragon to the map of China, and the flight from the map of China to the landing apron are defined as **stage Ⅱ flying process**, **stage Ⅲ flying process**, and **stage Ⅳ flying process**.
2. To better build our model, we defined the minimum safety distance between the center of any two drones is A meters, and the maximum observing distance between the center of any two drones is B meters. As a result, the distance between any two drones would be

In our case, each drone possesses a size of 384mm×384mm×93mm, with the propeller of approximate 15cm-diameter. Therefore, we plug A=2 to be the minimum safety distance between any two drones by our assumption when taking both the lower and upper boundaries into consideration. In this way, our model is more flexible since the minimum safety distance can be defined as any number that satisfies different parameters initial settings of drones under other circumstances. The analysis, calculation and computer animation appeared in the rest of our paper are all established and formed under this assumption.

1. In order to simplify our calculation and unify the coordinates in the three displays, we mark all the drones as drone number 1-477. To clarify, the number of each drone does not change from one image to another. In this way, we can easily present the transition track of each drone’s coordinates during the overall 5 stages flying process.

**Model**

1. **Modeling**

In response to the Mayor’s demand of investigating the idea of organizing an aerial light show, we start off our model by designing the three possible images that we would like to present. The first two images——the Ferris wheel and the dragon——are mandatory. With the help of reference images, we are able to design the stick figures of these two displays. With respect to the third one, we designed a map of China with a Chinese flag occupying the large territory in the west, representing the theme of “what ethnic is what worldwide”.

After establishing the images on display, we set up a left-handed three-dimensional coordinate system with the origin at the lower left corner of the parking apron. To be more specific, x-axis stands for the north direction; y-axis stands for the east direction; z-axis stands for the height. Then we pick out points from the images with the criterion that the coordinate difference between any two adjacent points remains no less than the minimum safety distance. Under this prerequisite, our minimum number of drones required would be 477, as each point selected out from the images represent a drone in reality. To validate the feasibility of the images in our model, we write a detect to examine whether the coordinate difference between every two drones is further than the minimum safety distance. The main theory in this program we write is to detect whether or not there is a drone in a circumference within the radius of the minimum safety distance. More specifically speaking, for each drone in the Ferris wheel, if there is a drone detected within the range of the minimum safety distance, it means that the two drones would ultimately crash with each other. The test result for the Ferris wheel from our computer simulation is shown below:



Figure 1: The result of the testing program of the Ferris wheel

From the figures, we can naturally reach the conclusion that any two adjacent drones wouldn’t crash with each other. With respect to the other two images——the dragon and the map of China, the crashing test result is shown in the appendix. All the test reports clear the possibility of crashes between drones.

As for the design process of the dragon pattern, we first download a picture of dragon and erase the dragon’s claws to simplify our model and reduce the total number of drones needed to gain better control. Then we use the Roberts edge detection operator to create the simplified pattern of the dragon. The Roberts edge detection operator computes the principle of gradient based on the difference between any pair of perpendicular directions, using the difference between adjacent pixels in the diagonal direction. The code is shown as following in figure 2:



Figure 2: The code of the Roberts edge detection operator

The simplified version of the dragon pattern is shown below in figure 3:

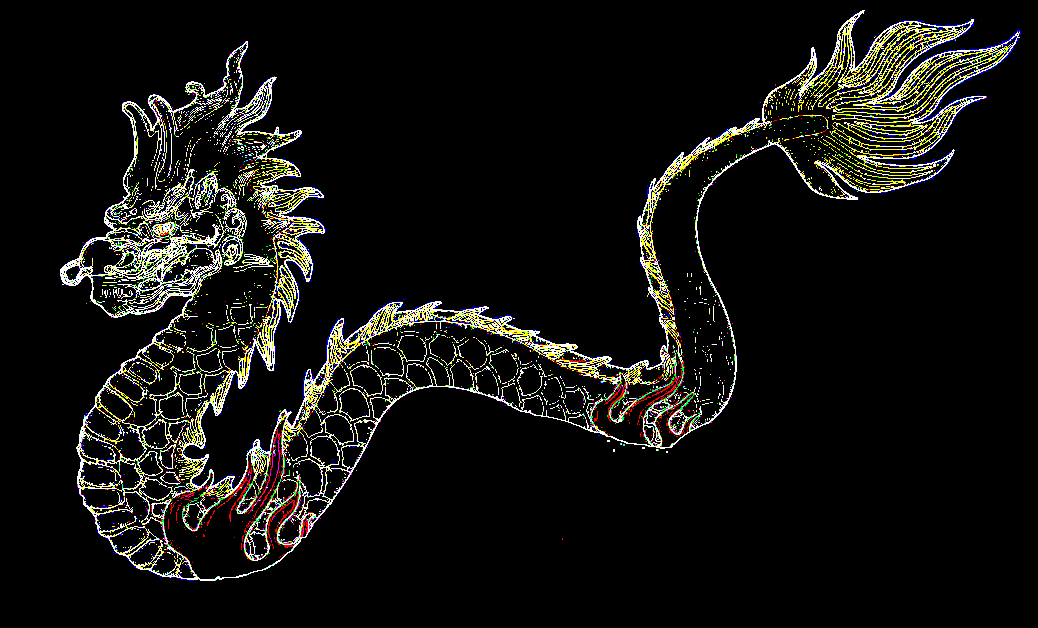


Figure 3: the changing process of the dragon pattern

Then we apply the Sobel operator to automatically generate the scatter diagram as below. By using this program, we do not need to pick out the point and take down the coordinates by ourselves, instead, it can all be done automatically by the computer.

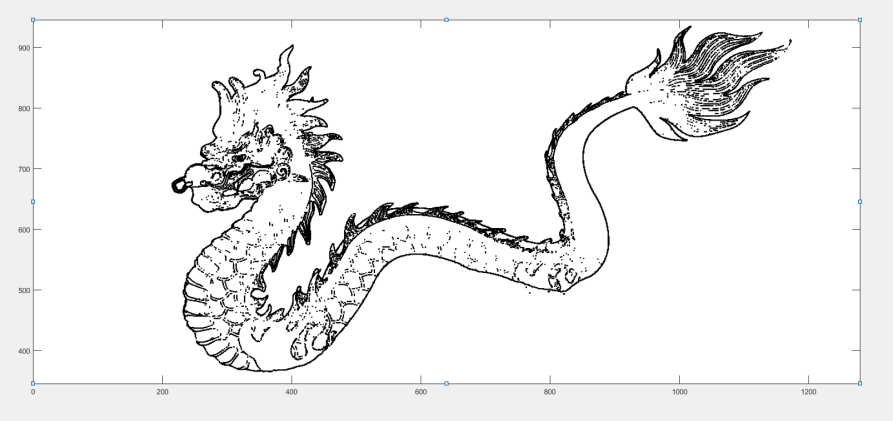


Figure 4: the scatter diagram of the map of China

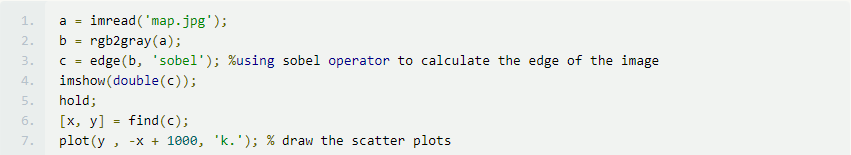


Figure 5: the program used to generate the scatter diagram

After the generation of the scatter diagram, we pick out the points with an interval of 1.5m on the edge of the diagram with help of Metlab. After the adjustment by our team, the ultimate design of the dragon pattern and the map of China is shown below in figure 6 and figure 7:

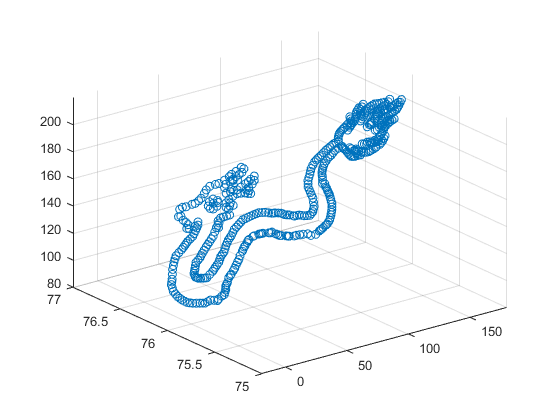
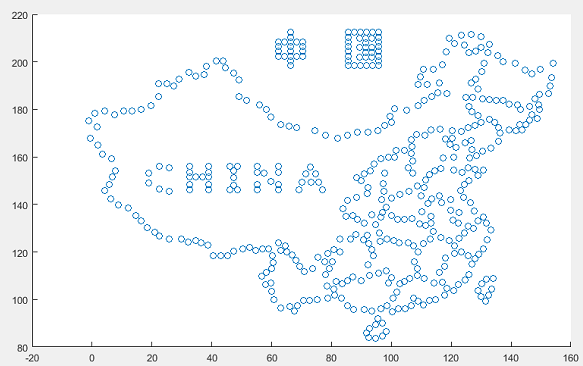


Figure 6: The shape of the dragon in MATLAB

Figure 7: The map of China in MATLAB



Lastly, we calculated the three-dimensional coordinate of each drone and use the coordinates to form matrixes for later steps in modeling. All the coordinates in the three images are attached in the appendix.

1. **The stage Ⅰ flying process**

The main challenge in this part of our model is how to distribute the flight path for each drone and make sure that there’s no intersection in any two flight paths. Since our design of the Ferris wheel is a tridimensional spatial structure which has 5 layers including the front, the back, the frame and the carriages. The following figure 8 shows the three views of the Ferris wheel:

Front view

side view

top view

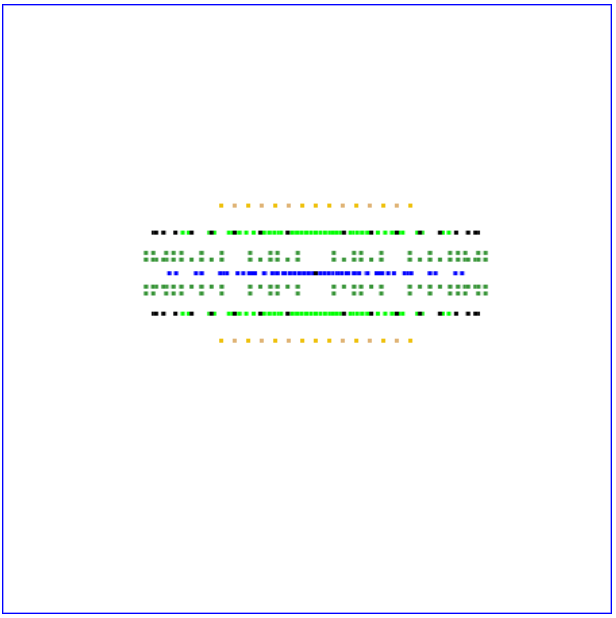
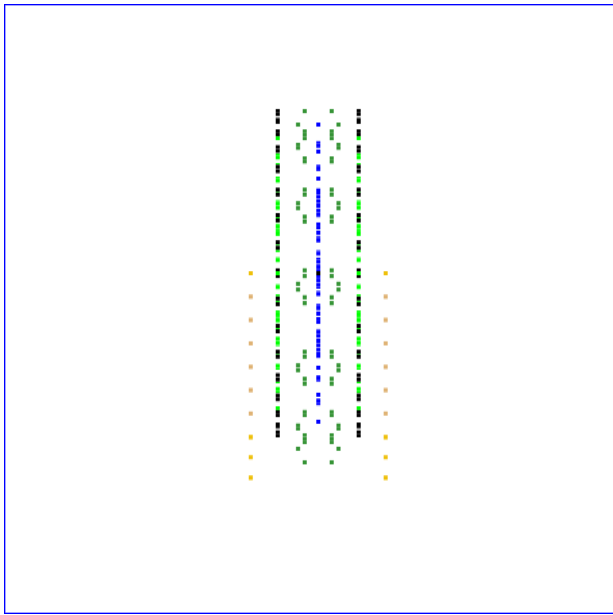
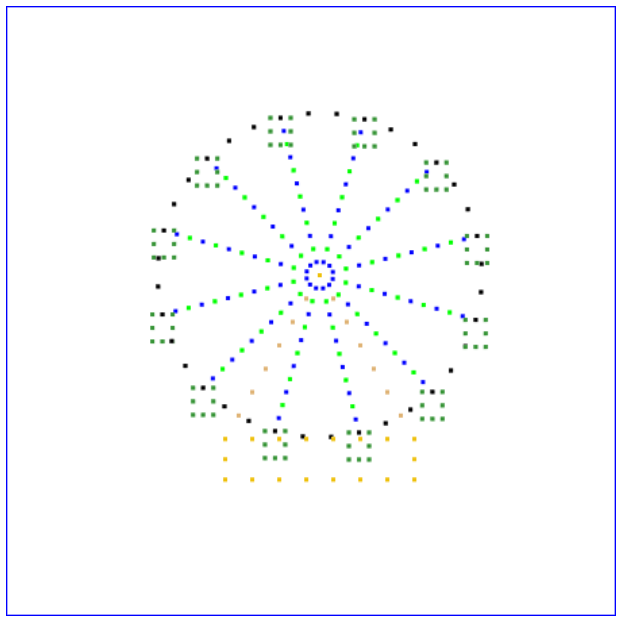


Figure 8: the three views of the Ferris wheel

The first step in our plan is controlling the drones to take off from the apron and raise up in the midair to form a horizontal Ferris wheel, and then flipping it to the upstanding position. In order to accomplish this goal, we divide all the drones in the horizontal Ferris wheel into several flying groups using the clustering analysis. The cluster criterion is the z-axis parameters. In other words, by the standard of height, we group the drones into 5 levels to gain a better control. The clustering result is shown in figure 9:

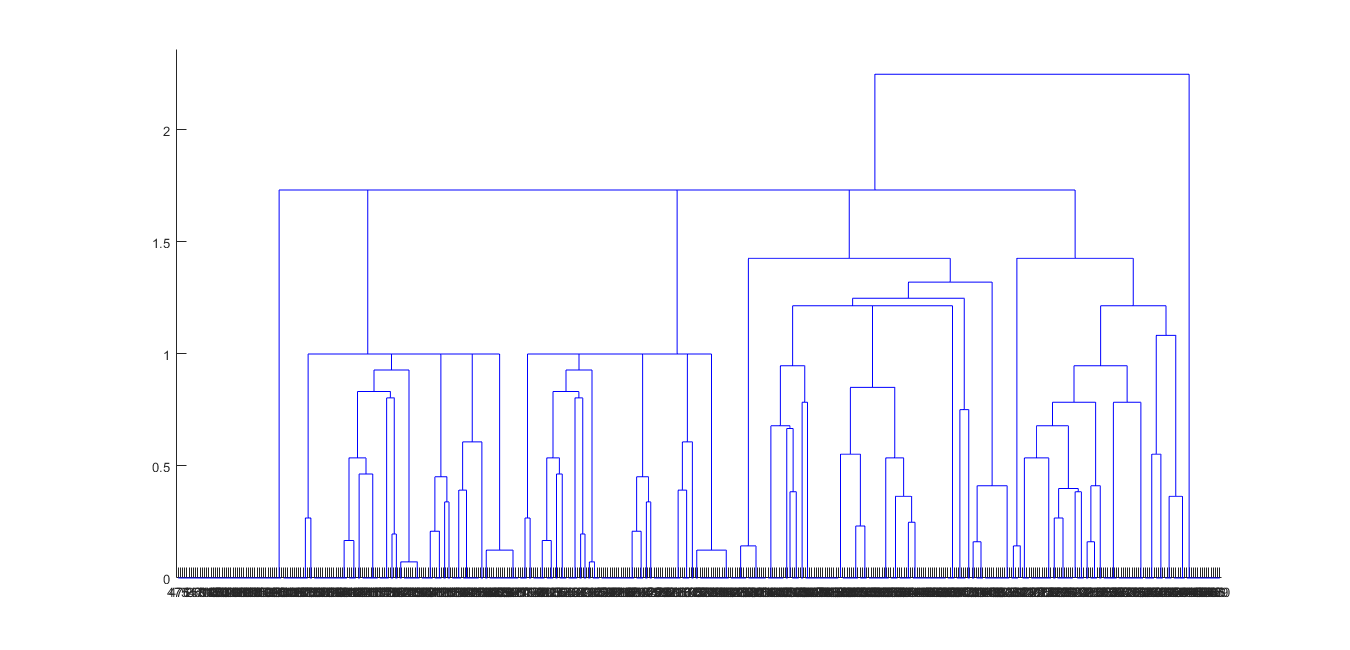


Figure 9: the clustering analysis result of the Ferris wheel

Our next step is numbering the clusters from top to bottom as cluster 1-5 and determining the priority of being distributed to a coordinate on the take-off apron for each drone. According to the clustering analysis result, the priority order would be cluster 5, cluster 3, cluster 2, cluster 1, and cluster 4.

First, we match every drone in cluster 5 with a drone on the parking apron that has the least Euclidean distance, which is

Then we match every drone in cluster 3 with a drone from the set of drones remaining on the parking apron that has the least Euclidean distance. Similarly, we repeat the matching procedure for another three times with the order of cluster 2, cluster 1, and cluster 4, and successfully form an overall match from drones forming the Ferris wheel to drones on the apron.

After affirming the initial coordinates and the final coordinates, we shift our focus to the flight paths that connect the initials with the finals. What we need to deal with now is to avoid the crashes between any two drones during the whole stage Ⅰ flying process. In the purpose of eliminating the possibility of crashes between any two drones, we adopt the method of forming the Ferris wheel on a horizontal plane with the parameter on the z-axis to be 150 first and then flipping the Ferris wheel around central axis of y=76, z=150 to shape the final image of the Ferris wheel that we desire to display.

As for the first part of the stage I flying process in which the drones fly to the horizontal plane, we connect the initial coordinate on the apron with the final coordinate in the horizontal Ferris wheel using a straight path for each drone. Hence, the general flight path for the drones can be described in a linear function, which is

The notation in the function is defined in the following table 2:

Table2: the definition of notations in the locus function

|  |  |
| --- | --- |
| notation | Definition |
|  | The matrix containing all the drones’ tridimensional coordinates |
|  | The take-off time for the drone |
|  | The actual flying time for the drone |
|  | The hovering time after arriving at the target position for the drone |

The next step after setting a general function for the flight paths of the drones is to validate the feasibility through computer animation. We begin our calculation by running a test program similar to the program mentioned earlier that is used to examine the distance between any two drones in the static initial state of each image to examine the flight locus of each drone under the condition that launch time interval between any two drones is second. The result of the first examining test is shown in the following figure 8:

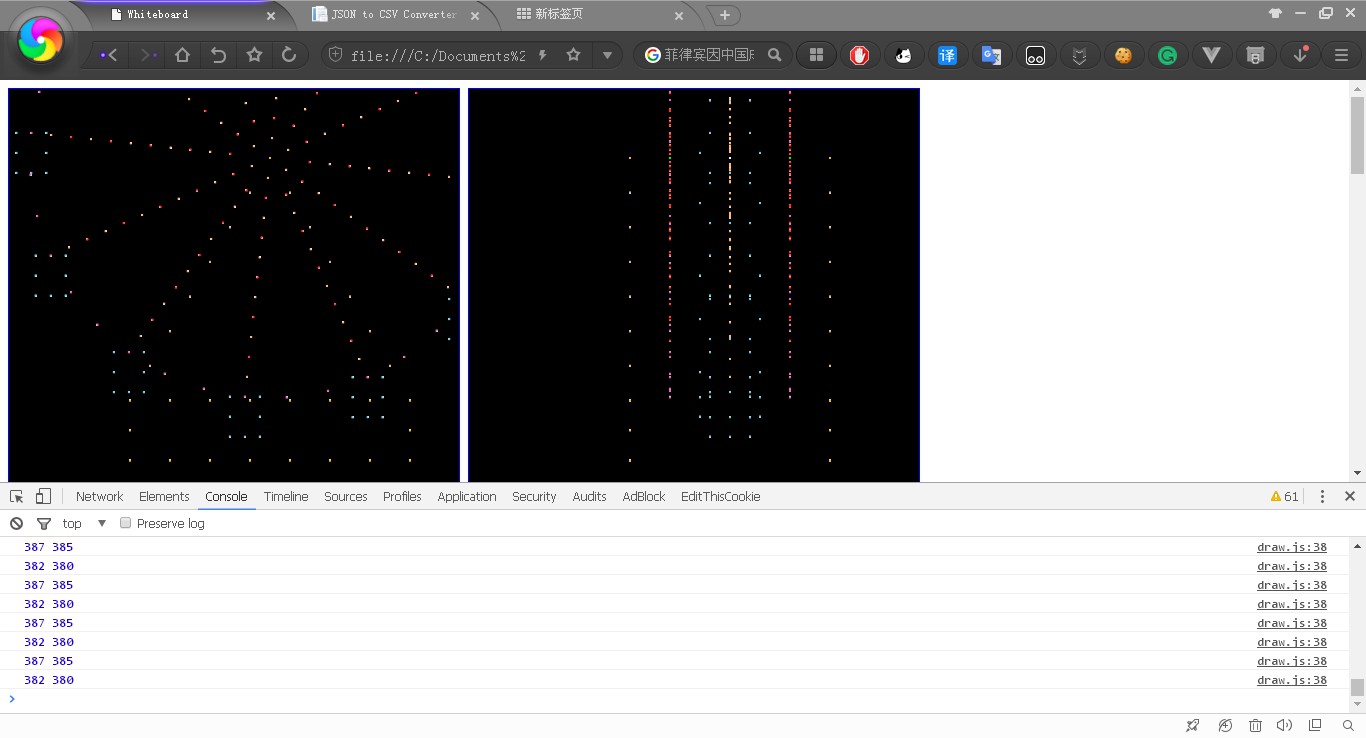


Figure 8: The first result of the examining test of Ferris wheel

From the diagram we can summarize that there are 61 drones in total that may crash with one another during the whole flying process. More specifically, the crashed pair of drones include drone (387, 385), drone (382, 380), etc. However, the crash of planes cannot be tolerated due to safety concerns. To optimize this result, we continue to adjust the launch time and time interval between each other for the crashed drones and manage to lower down the number of crashed drones to 27. With the utilization of manually postponing the take-off time of the crashed drones, we manage to completely eradicate any remaining possibility of crashing and the computer simulation validates the feasibility of our model. The specific postponing time lag is listed in the following table 3 and the testing report on the crashed drones is shown in figure 9 below:

Table 3: the adjustment of the take-off time for crashed drones in process1

|  |  |  |
| --- | --- | --- |
| Group | Delayed drone number | revised time lag |
| 1 | 386, 139, 152, 234, 233, 348, 366, 381, 383, 400, 185, 214, 209, 210, 360, 416, 424, 436 | 1s later |
| 2 | 152 | 1.5s later |
| 3 | 344, 228, 322, 324, 283 | 1s earlier |
| 4 | 230, 208, 209 | 1.5s later |

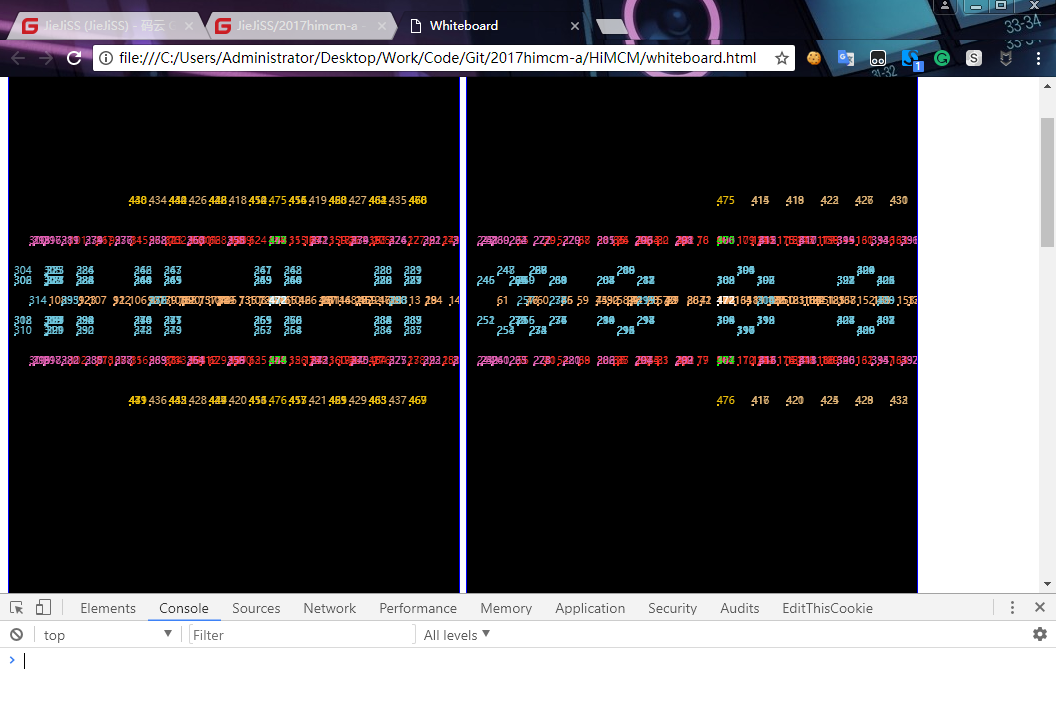


Figure 9: The testing report of the crashed drones on Ferris wheel

From the testing report of the crashed drones on Ferris wheel, it can be concluded that none of the drones would crash during the flying process till shaping the horizontal Ferris wheel. Then we flip the horizontal Ferris wheel into the upstanding position. The operative procedure is to make all the drones rotate at the same angular velocity to the upstanding shape. More detailed explanation is discussed later in the section titled” the rotation of the designs”.

Besides, a dynamic Ferris wheel can also be made by adding the following functions. It can make Ferris wheel more vivid and prevent audience's visual fatigue.

Pi(t)=(

where

(1

represents the included angle between the coordinates of drone number i and the axis y=76, z=150. The values of are listed in the appendix. Especially, the drones numbered 439 to 477 are on the brackets of the Ferris wheel or are the fixed point, therefore they are not included in formulas.

At this moment, our model has accomplished the flight up until forming the Ferris wheel pattern. The stage I flying process is well accomplished.

**3. The stage II flying process**

The main challenge of this process is how to minimize the total distance of drones flying in the air and make sure the flying pattern is optimal. In order to achieve this goal, we first divide the dragon pattern into three groups according to the shape and the outer contour of the dragon for later distributing drones on the dragon pattern to drones on the Ferris wheel. In the next step, we utilize binary integer programming to find solution that guarantees the total distance to be the shortest in each group. Then we use the testing program to check if there are drones that may crash into one another and adjust the matching method accordingly.

Our team first flip the upstanding Ferris wheel back into the previous horizontal position mentioned in the former process to lay the foundation for the following practice. The dragon consists of three parts including the head, the body and the tail part, and our team divides the dragon into three groups in accordance with the shape. We here define the head part of the dragon as group 1, the tail part as group 2 and the main body part as group 3. The division of the dragon pattern is presented in figure 10:

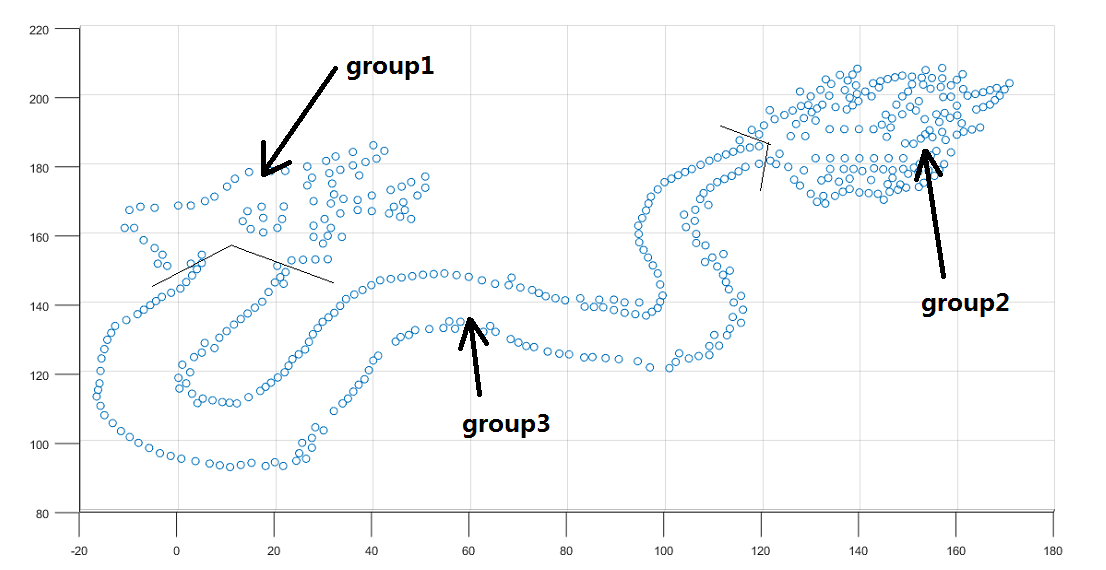


Figure7: The division of the dragon

Then we calculate the distance between every drone in the Ferris wheel (the one after rotation) and the dragon by using Euclidean distance as mentioned in the former process, which is

In this process, we do not match the drones based on the minimum Euclidean distance any more. Instead, we utilize binary integer programming to achieve our goal. The binary integer programming is a kind of special integer programming in which the conclusive variables include only 0 and 1. It can be used to find the overall optimal solution. In other words, the overall optimal solution in the process being discussed now is the shortest distance drones fly in total. We will give the specific explanation in the text below.

Our team first use the method of binary integer programming in group 1. We number the drones in the Ferris wheel from 1 to 477 and the drones in group1 from 1 to 78. Here we define as the distance between the drone in the Ferris wheel and drone in group 1. Then we define to judge the following situation: if the drone in the Ferris wheel matches the drone in group 1, then we define. Otherwise, we define .

The total number of the drones in the group1 is 78. Therefore, according to the definition above, we can define the objective function Z as the total flight distance of all the drones in group 1 as

Each drone in the Ferris wheel can only match up to one drone in group 1, besides, each drone in group 1 must match one drone in the Ferris wheel. Therefore, the constraint conditions are as the following:

Next, we utilize LINGO to run the binary integer programming, the details of the code are in the attachment.

After matching the drones in group1, the total number of drones left is 399. Then we apply the method of binary integer programming in group 2. Similarly, we number the drones remain in the Ferris wheel from 1 to 399 and the drones in group 2 from 1 to 167. Accordingly, we define as the distance between drone remaining in the Ferris wheel and drone in group2. Then we define to judge the following situation. To be more specific, if the drone remaining in the Ferris wheel matches the drone in group 2, then we define Otherwise, we define

The total number of the drones in the group 2 is 167. Therefore, according to the definition above, we can define the objective function in group 2 as

Every drone remaining in the Ferris wheel can only match up to one drone in group 2, besides, every drone in group 2 must match one drone in the Ferris wheel. Therefore, the constraint conditions are as the following:

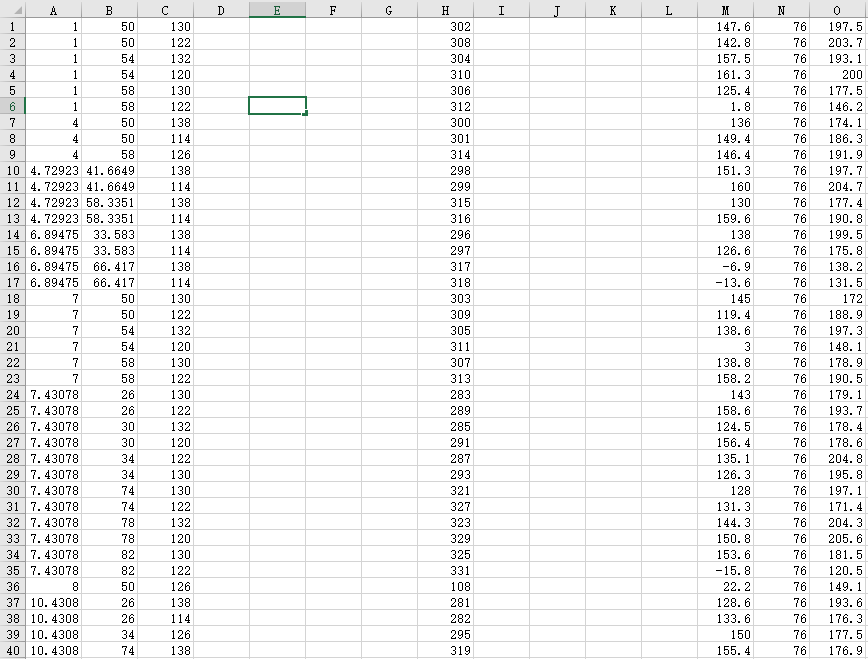
Then we again use LINGO to run the binary integer programming, the details of the code are in the attachment.

Then, there are only 232 drones left in group 3. We respectively number the drones in the Ferris wheel and group3 from 1 to 232. As mentioned above, we similarly define and The total number of the drones in the group3 is 232. Therefore, according to the definition above, we can define the objective function in group 3 as

Each drone in group 2 must match one drone remaining in the Ferris wheel. Therefore, the constraint conditions are as the following:

Again, we use LINGO to run the binary integer programming, the details of the code are in the attachment. Now every drone in stage Ⅱ process is matched, part of the matching results is shown in the following table 4:

Table 4: The matching result of the second flying process



Column A, B, C represents the x-coordinate, y-coordinate and z-coordinate of drones in the Ferris wheel respectively. Column H represents the drone's original number. Column M, N, O represents the x-coordinate, y-coordinate and z-coordinate of drones in the dragon respectively. The complete results of the matching in the second process are in the attachment.

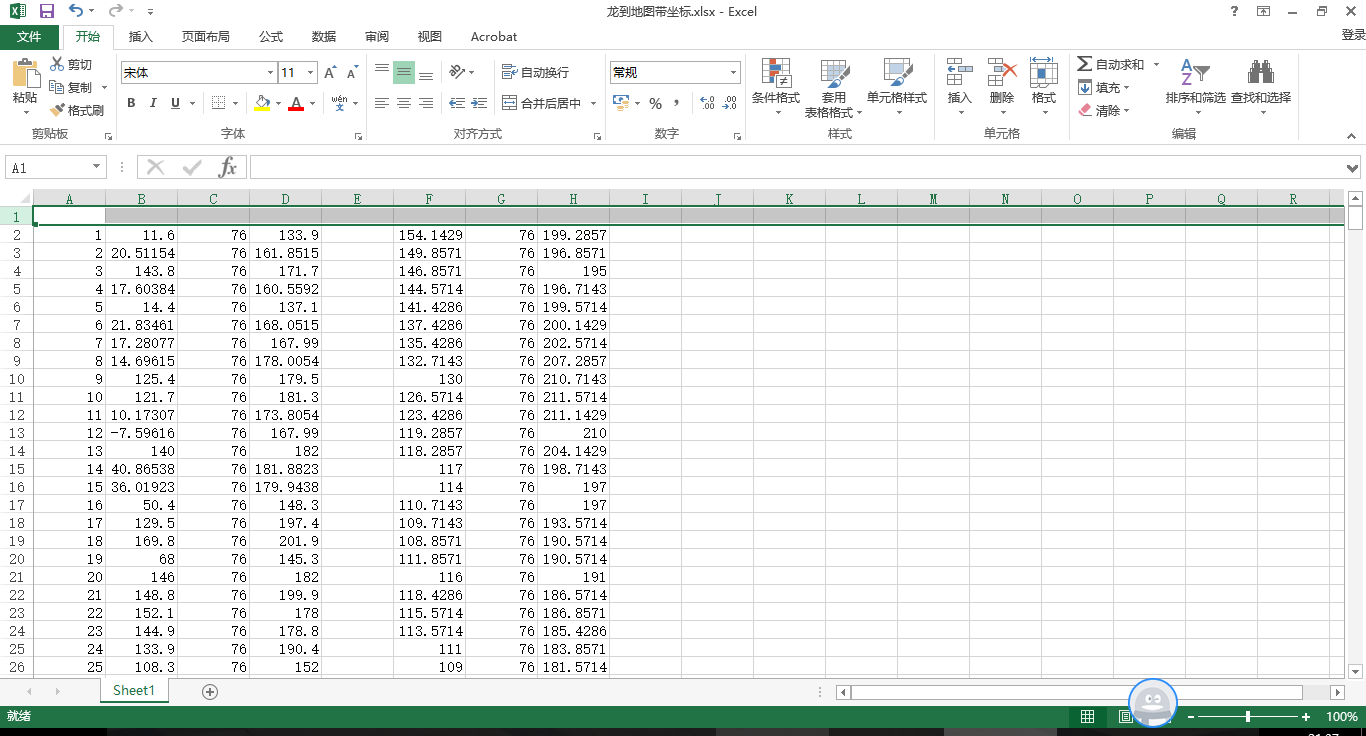
1. **Stage Ⅲ flying process**

The main challenge of this process consists of two goals. One is to minimize the total transition time from the dragon pattern to the map of China by means of optimization of the total distance. The other is to ensure that none of the drones would crash with each other in purpose of satisfying the safety concerns. To accomplish both of the goals, we apply the multi-target programming to our model. In other words, we find the optimal solutions that meet either goal separately and then weighting the two solutions to find the overall best-fit model.

With the same way as the stage II, we calculate the distance between every drone in the two patterns using Euclidean distance to utilize binary integer programming. However, in lieu of dividing the entire patterns into several clusters, we regard the overall drone fleet as a whole to simplify the model. Possessing 477 drones in total, the formulas are as following likewise:

We also apply LINGO to run the binary integer programming, the details of the code are in the attachment, part of the matching results is as the following in table xx:

Table xx: The matching result of stage Ⅲ flying process



Column A, B, C represents the x-coordinate, y-coordinate and z-coordinate of drones in the Ferris wheel respectively. Column H represents the drone's original number. Column M, N, O represents the x-coordinate, y-coordinate and z-coordinate of drones in the dragon respectively. The complete results of the matching in the second process are in the attachment.

Though the binary integer programming seems quite effective in search of the solution. However, there exists a flaw in this process. All the drones have to fly in the same surface, therefore, it's highly possible that the drones may crash into one another. In order to tackle this serious problem, our team then use the quadratic function to adjust the drones' flying path. In other words, instead of applying the flight path to a simple straight path, we utilize the shape of quadratic function as a drone’s flying pattern. By modeling in this way, we can use the space more efficiently and avoid the collision at the same time.

First, our team utilize trigonometric function to match the drones on the dragon with drones on the map. We set a reference point O (57, 57, 112.5), the center of the map pattern, to generate the relative vectors of each coordinate. As is known to us all, within the range from 0 degree to 180 degrees, the closer the cosine value of the vector angle is to 1, the nearer the two vectors are. Since the range of the cosine function defined in 0 degree to 180 degrees is from -1 to 1, we can reach the conclusion that the greater the cosine value is, the near they are. Therefore, we calculate the vectors from the reference point to the coordinate of each drone in the two patterns. The transformation technique is shown as below:

Then we calculate the cosine value of each vector in the dragon and each vector in the map.

Suppose there are two vectors **OA** and **OB**, and their included angle is then we can calculate the value of cosine between these two vectors by the following formula

In our case, the formula accordingly changes to the following:

After repeating the steps above, we are finally able to get a better matching pattern. However, there are still a lot of collisions after the pattern above is realized. Therefore, we need to further optimize our program. In the next step, we apply the quadratic function to optimize the path of every drones. Assuming the flight path function of each drone is

The shape and opening size of a quadratic function is determined by the coefficient of , so obviously *a* is the conclusive element in the function. In order to avoid the crash, we need to make the coefficient *a*in each drone's function as various as possible. Besides, the larger |*a*| is, the longer the path will be, thus the numerical value of *a*need to be close to zero as possible in order to minimize the total distance that drones will fly in the space.

In order to achieve this, we use a special distribution pattern to determine the value of *a* as the following. We first define the determinant *slope* as

*slope*=[-1.6, -1.5, -1.3, -1, -0.8, -0.6, -0.3, 0.3, 0.6, 0.8, 1, 1.3, 1.5, 1.6]

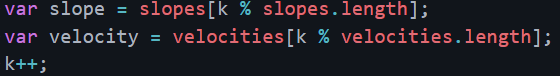
Then we define the determinant *velocity* as

*velocity*=[0.7, 1]

In the next step, we calculate the value of *a* by multiplying the number in the *slope* with the number in the *velocity* in the following pattern. If the remaining of the original number divided by 14 is *k* (0), and the remaining of the original divided by 2 is *t* (0), then we define *a* as the product of the number in the determinant *slope* and the number in the determinant *velocity.* For instance, if the original number of the drone is 20, then the value of the coefficient *a* in this drone's path function is

a= -0.3\*0.7 = -0.21

In that case the values of the coefficient *a* of drones that are numbered close to each other will be different, and therefore we can avoid the potential crash of drones.



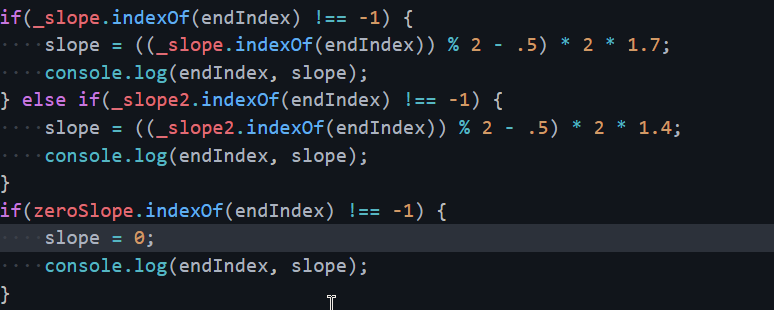


Figure xx: The assistant interpretation of the quadratic function

Besides, we also add the *sleeptime* for every drone in the similar way, the determinant *sleeptime* is as following:

*sleeptime*=[0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8]

Similarly, we add the *sleeptime* according to the remaining after the original drone's number is divided by ten. The unit of the *sleeptime* is second. The *sleeptime* is also applied to avoid the potential crash of the drones by staggering the take-off time.

However, this algorithm can't guarantee that none of the drones would crash into each other, so the adjustment of the path and the time are also needed. The figure xx below shows the crash of drones numbered 470 and 389 and the crash of drones numbered 77 and 8. The total number of crashes up to now is 13.



Figure xx: the testing report of crashed drones

Therefore, our team's main task now is how to adjust the drones' path and time properly. First, we adjust the drone's starting time by postponing the time that the drones take off or by making the drone take off earlier. The specific revised time lag is listed in the following table

Table : the adjustment of the take-off time for crashed drones in process III

|  |  |  |
| --- | --- | --- |
| Group | drone number | revised time lag |
| 1 | 160,443,101,467,458,253,288 | 1s later |
| 2 | 441, 472,129, 431 | 1.5s later |
| 3 | 362,440,466,357,457,53,410, 324, 337, 355, 413, 369 | 1s earlier |
| 4 | 30, 31, 413, 454, 453, 230, 470, 469, 256, 450, 356, 117, 358, 432, 423 | 1.5s later |

In addition, if time is not proper to adjust, we can also choose to adjust the pattern of path by changing the value of coefficient *a* in the flying path function. The specific changes that are made to the coefficient *a* are presented in the following table.

Table : the adjustment of the value of coefficient *a* for crashed drones in process III

|  |  |  |
| --- | --- | --- |
| Group | Changed drone number | the value of *a* after revision |
| 1 | (192, 230) (457, 53) (410, 356) (413, 440) (470, 357) (472, 447) (238, 458) (456, 70) (160, 252) (466, 200) (195, 469) (6, 228) (174, 443) (283, 151) (217, 26) (1, 432) (45, 339) (360, 359) |  |
| 2 | (61, 55) (337, 311) (276, 318) (252, 245) (415, 410) (463, 195) (454, 11) (453, 362) (348, 425) (424, 75) (74, 122) (124, 354) (353, 469) 467 |  |
| 3 | 339, 423 | 0 |

From the table, we can conclude that the coefficient a in the two drones that are likely to collide with each other is changed into opposite values so that they would fly towards completely opposite directions. Especially, drone number 339 and 423 are revised to fly in straight path to avoid collision. The complete results of the matching in the third process by using quadratic functions are in the attachment.

To put in a nut shell, both of the methods have strength and weakness. The binary integer programming focuses on the optimization of total distance of drones; However, according to the computer simulation ran by the detect function, an astonishing number of drones would crash with each other. As a result, there is unfortunately no guarantee of the safety consideration. In sharp contrast, the quadratic function method prioritizes the safety concerns and ensures that the solution does not result in crashes no matter how far the total distance is. When combining the two methods together with the utilization of weighting and multi-target optimization. Under the prerequisite that safety always comes first, we determine to value more of the quadratic function. The only flaw lies upon this method is probably the waste of show time. Yet time is never a crucial variable as time is of great abundance for the aerial light show when compared to the maximum flight time of drones.

As is mentioned at the beginning of this section, in order to reconcile the dilemma between the shortest path and safety concerns, we apply a multi-target programming to devise the correspondence of each drone from the dragon to the map of China. We introduce the cosine value of the angle of two vectors to measure and quantify the flight paths. Then we calculate the Euclidean distance between each two coordinates as a preparation to the multi-target programming.

We number the drones in the Ferris wheel from 1 to 477. Here we define as the distance between the drone in the dragon and drone in the map, as well as as the cosine vector angle between the drone in the dragon and drone in the map. Then we define to judge the following situation: if the drone in the dragon matches the drone in the map, then we define. Otherwise, we define .

Considering the first goal, we define function Z, the minimum total distance, as one of our target function. We define function W, the maximum total cosine vector angle, as another target function.

In a similar way, the functions are subject to

Combining the two functions into one function, we define *λ1* as the weight of the first function and similarly *λ2* as the weight of the second function. Hence, we have

Since the two conditions are equally important to the final consequence, we attach the two functions to equal weight, which means .

We also apply LINGO to run the multi-purpose binary integer programming, the details of the code are in the attachment. Nevertheless, due to limited time, we fail to collect the final data of the program.

1. **Audience's best viewing angle optimization**

This part of the essay is going to discuss the best viewing distance for audience to enjoy the visual feast.

Assuming the best horizontal distance is X. If the highest value of z-coordinate is A, the lowest value of z-coordinate is B, then we can calculate the best viewing angle  by using the following formula

Then we calculate the derivative of  to find out its minimum value. We define ' as the derivative of  and it can be calculated by the following formula

When,  has its minimum value, and we can calculate the value of X,

According to the z-coordinate of the first design, the Ferris wheel, the highest point of z-axis is approximately 224m and the lowest point of z-axis is approximately 116m. Therefore, the best viewing distance is

m

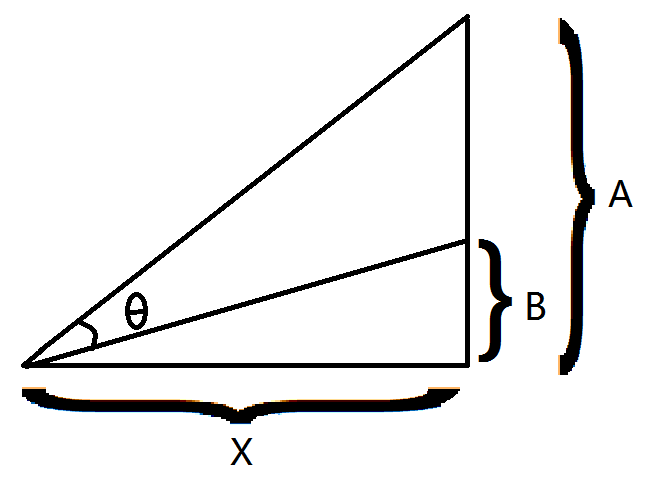


Figure 11: the sketch of the best viewing distance

According to the z-coordinate of the second design, the dragon, the highest point of z-axis is approximately 156 and the lowest point of z-axis is approximately 70. Therefore, the best viewing distance is

m

According to the z-coordinate of the third design, the map of China, the highest point of z-axis is approximately 159 and the lowest point of z-axis is approximately 63. Therefore, the best viewing distance is

m

Since audiences' viewing position is fixed, the three best distance above should be averaged in order to find the best horizontal distance for audiences to view the three patterns. The average best distance is

m

1. **the rotation of the designs**

In this part of the essay we will discuss the functions of the rotation of the Ferris wheel, the dragon and the map of China being mentioned in the former parts.

If the coordinates of a point A (x, y, z) in the space is known, then the coordinates of point

A'(x, y, z) after rotating around the axis y=0, z=0 for an angle  can be calculated by the following formula



In this way we can successfully calculate the coordinates of the terminal point. According to the formula, we can also calculate the path functions, which the designs rotate around the axis y=76, z=150, as the following

In the functions above, the  represents the angle between the vertical axis of the starting point to the axis of rotation and the axis x=76, z=150.  represents the angle that the point rotates around the axis. The sketch of the rotation of the designs is presented in figure 12:

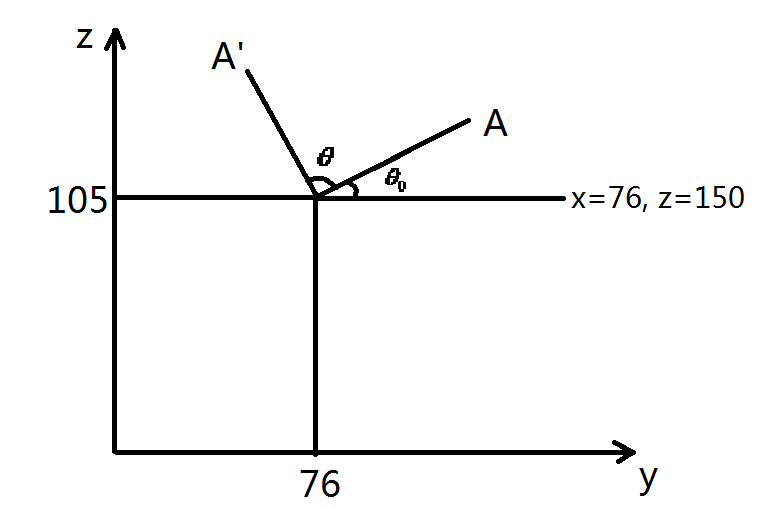


Figure 12: the sketch of rotation of the designs

The function between angle  and time *t* is as following



where



where



We utilized the Bezier Curve since it could make the animation more vivid, and the speed of the drones is more close to real situation.

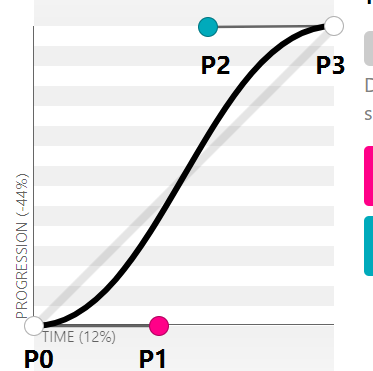


Figure xx: The image of the Bezier Curve

1. **Detection function**

This part will mainly discuss the detection function and its optimization. The detection function is of vital importance in the modeling when taking safety concerns into consideration. It will make the bugfixes and adjustment of crashed drones much more quickly and thus improve our efficiency. Therefore, the following text will cover the algorithm of the detection function and how it can be optimized.

According to the minimum safety distance mentioned in the Definition part, the distance between any two drones has to be further than this minimum value to meet the safety demands. First, our team utilize the Euclidean distance as part of the detection function. We calculate the Euclidean distance between every two drones at the same time by the following formula

In the formula above, x, y, and z represent the x-axis, y-axis, and z-axis coordinates of the drones respectively. If the distance calculated by the formula is below the minimum safety distance, then the program will send an error message. The program can calculate the total number of the error messages and report them to our team members. An example is presented in figure 13:

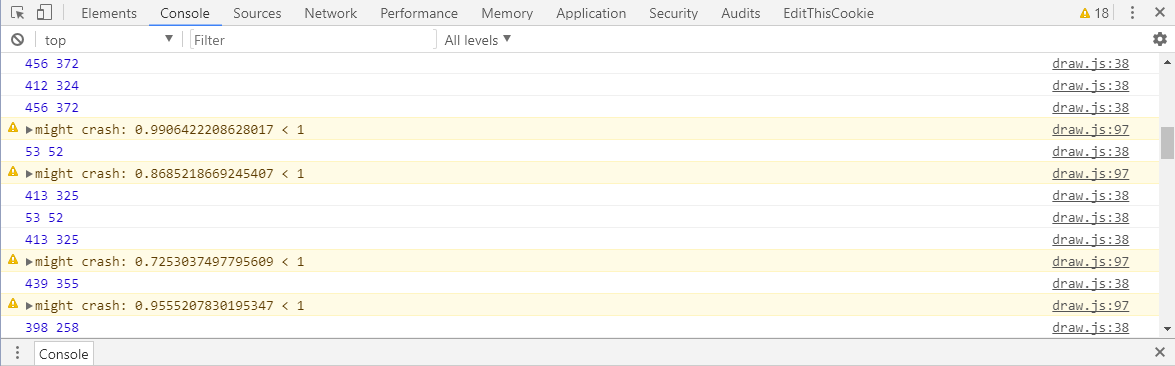


Figure 13: an example of the detection function

Then we improve the detection function using vectors. It can be applied to detecting the minimum distance between two straight lines with the advantage that it calculates the distance without the restrictions of time since time is no longer considered as an independent variable of the detection function.

Suppose the starting and ending points of two drones are A（, B（, C（ and D（. Then we can define the vector **n** as

**n**=**ABCD**

The vector **n** is perpendicular to both line AB and CD in the space. Then we calculate the minimal distance between two lines AB and CD by the formula

Vector **AC** can be replaced by any vectors whose starting point is on line AB and ending point is on line CD. The notation “d” in the formula above represents the projection of vector **AC** on the vector **n**'s direction, which equals to the minimal distance between the two lines. Then the program will judge the following situation: if d is less than the minimum safety distance, then the program will further detect if two drones will possibly crash into one another. Otherwise, the result is clear. A sketch of the vector method used in optimizing detection function is presented in figure 14:

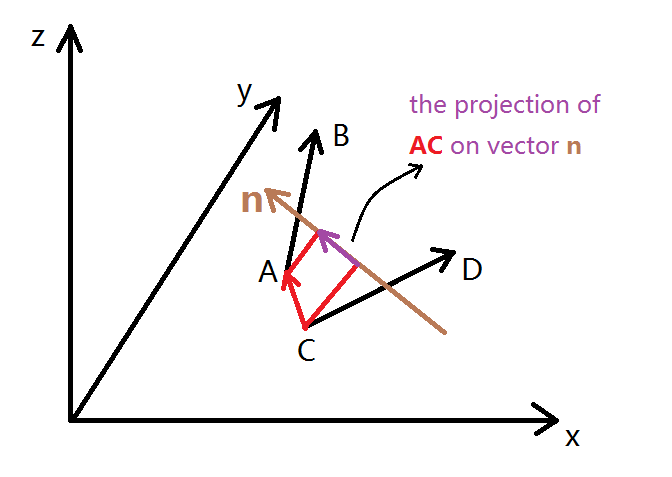


Figure 14: the sketch of the vector method

In conclusion, the second algorithm can be used as a filter program to reduce the workload of the program by straining off the matches that certainly won't collide with one another. As for the comparison and contrast between the two detection methods, the common method is easy to operate while the vector method has a smaller calculation amount. Considering that the computer can handle large quantity of calculation, we tend to adopt the common method to detect the distance between any two drones.