For office use only		For office use only
T1 ___		F1 ___
T2 ___		F2 ___
T3 ___		F3 ___
T4 ___		F4 ___

2017
20th Annual High School Mathematical Contest in Modeling (HiMCM) Summary Sheet
(Please make this the first page of your electronic Solution Paper.)
Team Control Number: 8028
Problem Chosen: A
Please paste or type a summary of your results on this page. Please remember not to include the name of your school, advisor, or team members on this page.

We were very excited to be assigned the task of researching the feasibility and the requirements for an air light show using hundreds of drones for our city to celebrate the 40th anniversary. We have made our best to study the mathematical principles of the design, including the representation of drones, the design of static image displays, and especially the implementation of animations of images. We have enjoyed a few intensive working hours to make our research valuable and more over, feasible for the actual realization of the air light show within two years, before the actual 40th anniversary air light show to come true.

In this paper, we did not talk about too much about the principle of the drone's fly, instead, we assume that the drone can be controlled almost accurately to any position at our will, which made us focus on the positioning for the static images to display, and the route planning for images to animate.

In the modeling of the drones, we used 3D coordinates for each drone. The control of the drones was simplified to a sequence of positions that each drone should appear at predetermined time points in 3D space. In order to design and implement the show, we need to implement some basic actions for the set of drones, including launching, launching to form static images, translation of the static images without reshaping, rotation of the static images about arbitrary axis, transition between different static images, etc. With these basic action being implemented, we were able to make a sequence of combinations of multiple actions into a complicated action to make the show much more attractive. For example, we can see a static ferris wheel in the sky, and a static dragon in the sky, and a transition from the ferris wheel to a dragon in place, but with the combination, we can easily realize a transition from the ferris wheel to the dragon while the drone as a whole, were flying to some where. We can definitely implement many more very attractive effects for the air show to make the whole air show a unforgettable experience to anyone who would be an audience of the show.

In the implementing of the show, we found out that matrix is such a powerful tool in this task. We have seen the simple but elegant way to writing complicated formulas, and especially, that easiness of using matrices in MATLAB to make a simple emulations to test our ideas. We used matrices to implemented the representation of positions of drones in 3D space, and implemented the translation of image as a whole, and rotation of image as a whole about arbitrary axis in a few matrix operations, and the most exciting part, the implementation of the transition from one image to another with the help of matrices. The math part looks so elegant and makes us very excited.

Most of the actions could be implemented easily except the transition which is the most crucial part of the whole paper. We tried to find an easy to understand while heuristic

method for this task. We came to realize that the key of transition had two parts, matching and translation. The translation was implemented using matrix without too much problems, while the transition became the last obstacle in front of us. It came to our mind that polar form might be helpful, because it was easier to find proper matches for all the drone by considering the angles and distance from the center than using x-y coordinates. Finally, we came to a simple idea, that is, to match the angles of drones in the polar form in ascending order. The one with the smallest angle in the initial image would go to match the one with the smallest angle in the target image, and so on. This idea looks so simple but reasonable, since drones were most fly around the center of all the drones, and should not travel a large angle to reach its destination in general. With this idea in hand, we realized that this idea still had a cushion to make some improvements. Although angles were considered and drone may travel across a smaller range of angles, it may still goes a longer distance to reach the target. Then, we came into an improved ideas to match the drone not only by the sorted angles. We split drone onto a few groups based on the sorted angles, and within a group, drones were actually match to each other based on the distance from the center. With this approach, we have tried to reduce the number of possible long distance travels from the previous idea. We made some experiments on the group size, and found that when the group size was selected properly, the overall performance could be greatly improved.

The last problem is the avoid collision between drones, which is very crucial for a show. Drone may be very close to each other while flying, causing potential harm to each other. In this case we need to properly adjust the positions of the involved drones to make them apart from each other. We provided two approaches, multi-layer approach, in which drone were flying in multiple layers; and a time delay approach, in which one of the colliding drones may wait until the other drone pass by. We have a simple emulation to test the multi-player approach and concluded that it can greatly reduce the number of possible collision.

When we have conquered all the mathematical problems, we designed three images, a Ferris Wheel, a dragon, and a WING representing our city. We also calculated the number of drones that are required to make the image more vivid, and the required space for the launching and the audiences.

We have determined that an air light show is a feasible program to appear in our city anniversary celebration. The show could be implemented under the current conditions that our city is able to provide. We were very eager to see the show come true within two years.

Designing and Implementing Air Light Show using Drones

#8028

November 19, 2017

Contents

Abstract

On the annual festival of Shenzhen this year, a fabulous aerial show will be held. With drones and orchestra interacting with each other, it will certainly give audience an unforgettable experience. Our team is asked to investigate how to manage the drones and organize three 3-D displays on the show: Ferris wheel, dragon and Shenzhen Citizen Center (landmark of Shenzhen, its roof looks like the wings of a large fabulous bird).

Firstly, to manage the movements of drones, we design matrices for each type of possible movements: translation, rotation in 3-dimension space, and transition from one image to another image, and also a combination of these movements.

Secondly, for each display, we have created models with all the drones represented by coordinates for each of the display. We also design the flying pattern of the drones in order to achieve the minimized travel distance.

Thirdly, we improve the models by modulating the number of drones in order to satisfy the safety considerations and reduce the frequency of potential collisions. We then determine the required launch area and required air space right from our model. Using the models of movement in the previous part, we can also determine the time duration for the whole light show.

After that, we spot out the weaknesses of our model is that all the drones only fly in one layer. This will increase the chance for collision.

Eventually, we will display the show in front of the Grand Theatre in Shenzhen. The result of the testing of our model is good enough to match the safety consideration as well as the entertainment level. However, we still need to consider other uncertain factors such as strong wind. Due to the limit of time, we are not able to investigate further the particular factors.

Keywords: Drones, Aerial show Collision Frequency Rotation and translation in 3-dimension

1 Introduction

Tourism is one of the most important industry in most of the countries. Despite of the fact that traditional sight-seeing is still the majority, new types of entertainment such as aerial show is becoming more and more attractive for the whole industry to concern. With the development of technology these years, people are now considering using drones instead of fireworks or laser to create more delicate and lively shows. Unfortunately, due to the disturbance among different drones, the accuracy of locating function of GPS and some other reasons, it can be difficult to organize drones into live shows with accompaniment of orchestra. In this paper we provided solutions to achieve the complex movements of drones and proposed three displays to be performed to form a live air show.

2 Problem Restatement

In order to make an air show come true, there two two parts. The first part is to control to drones to fly. This is mostly a problem solved by the manufactures of the drones. They should design drones to fly based on the sequence of pre-determined positions. The second part is to tell the drone where to fly. This is to supply the sequence of positions to the drone. In order to make a show more vivid and attractive, we should carefully calculate the positions of each drone involved in the show at any given time, so that the drones can displays special images.

In order to represent drones, we should use a proper coordinate system. Given that the GPS positioning is more closer to a 3D xyz-coordinate system within a smaller range, 3D xyz coordinate system is used. Drones were described by set of 4 numbers, $< t, x, y, z >$. All the positions of the drones were pre-determined before the show.

The display of static images can be easily implemented by calculating the positions of drones in the static image, which could be determined by the image directly. But, we should definitely implemented much more than displaying static images. Actually, even the forming of the static images is special, which is a combination of a translation of the whole set of drones and a transition from the launching image to the static image. So implementing a transition is the core of the whole task.

The translation is a movement in which the whole set of drones move into the same direction by the same distance. The amount of change in position can be easily determined in each time interval, and so the positioning of all the drones can be calculated.

The transition is to convert the drones from an initial image to a target image. This is a little more complicated, since not all the drone fly in the same direction. The key in the transition is to match drone one to one, and then fly the drones. The fly of drones could be implemented similar to a translation with the exception that translation of each drone is unique.

We also need to implement more actions to make the show vivid, for example, a rotation about a given axis, etc.

In addition to the actions of movement of drones, we also need to carefully design the light scheme, so that the drones can display more attractive effects, for example, flashing, etc. But this is relatively easier and determined. We only need to code into the drone about the status of each light on it. The information at a given time can be represented by a long vector

$$< t, x, y, z, red, green, blue >$$

In this paper, we are not going to discuss the lighting design. We are going to focus on the positioning of drones.

In the following parts, we were going to discuss in detail about the mathematics behind the implementation of each action.

3 Asumptions

Before proceeding to our modeling work, we need to first make some reasonable assumptions. These assumptions include but may not be limited to the following:

- Suitable weather (No wind, no rain). We are assuming the weather is good enough for our drones to operate accurately and safely.
- Safety zone for each drone - 2 meters. The distance between each pair of drones should be large enough so that they are not going to affect each other during the flight, for example, the wake flow of the former drone won't affect the later one's flight status and route.
- All drones are of the same model with the same parameters. We adopted the parameters of the intel shooting star, as in the following table.

Parameter Name	Intel Shooting Star
Size	$382 \times 382 \times 83$mm
Weight	330 g
Rotor Diameter	6" (15cm)
Max Flight Time	20 mins
Max Range	1.5 km
Max Tolerable Wind Speed	8 m/s
Cruising Speed (In show)	3 m/s

- Drones will not be damaged due to collision or malfunction.
- Drones can fly to any feasible specified position with the help of GPS.
- This positions of drone were given at an interval of 1 second
- The speed of the drone is the same regardless of the direction of movement.
- Drones are able to recalculate it's route to the next pre-determined position and go back on track immediately, even though they have missed one position.
- Exhibition is in a large open space without any obstacles in the space.
- The accuracy of GPS locating is with in 10 cm.

- In the case of Ferris Wheel, we ignore the difference between a polygon’ s side length (since the image connected by the drones is not a perfect circle but a polygon) and a circle’s arc between each neighboring drone on the circumference of the circle.
- We ignore the effect of air resistance made on accidentally falling.
- If the drone is somehow hit and headed downward, the propellers’ speed will not increase.
- The light emitted from the LED light of the drone will not harm human eyes.
- The maximum height limit for the drones from the local government is 400 m.
- The transmission signal won’t be interfered by out-side factors and other drones.

4 Modeling the Motion

We model the drones as points in the 3-D space. For simplicity, we focus on discussing the movement of one drone. The position of the drone at time t is represented by a 3-D vector.

$$P_i(t) =< x_i(t), y_i(t), z_i(t) >,$$

where $P_i(t)$ represents the position vector of the i_{th} drone at time t, $x_i(t), y_i(t), z_i(t)$ represent the x, y, and z coordinate of the i_{th} drone at time t, respectively.

The flying path of each drone is consists of a set of predetermined vectors, and the status of the light of the drone at each time point is also predetermined and coded into the drone together with the position. We are not going to discuss the operation of the light operation in this paper.

The main tasks for realizing the light show is the coordinated route planning for the whole set of drones. The light show may be split into time intervals of 1 second each. The positions of each drone at each time point are pre-calculated according to the arrangement of the light show. There are a few basis operations for the light show to come true, including:

- Alignment before launch
- Launch to form images
- Translating of images into any direction
- Transition between images
- Rotating the image about arbitrary axis
- Landing

In this section, we are going to discuss the implementation of each of the basic operations.

The drones need to perform complex movements throughout the show including translation and rotation. Each of them also has a distinct path. To simplify the question, we concentrate on a single drone and use a point on 3-dimension graph to represent its position. To accurately calculate the path for each drone, we used matrices and trigonometry to help find the positions of drones.

4.1 Alignment before launch

The drones are aligned regularly before the show. In the intel light show, we saw that the drones are aligned in a grid on the ground, in which each drone occupy a point in the grid. The distance between the grid points is about 3 m to ensure the flight safety. In our plan, we follow the same solution to organize the drones into a grid whose size is determined by the number of drones required for the whole light show.

4.2 The launch of drones

Drones will be launched from the ground to form the first image in the sky. This is a combination of a translation of image and a transition between two images, in which the initial alignment forms the beginning image. This will be discussed in the following subsections.

4.3 Implementing the translation

A translation of image is a movement in which all drones are moving to the same direction without changing shapes. Translation can be implemented easily by adding a direction vector into the current position. To perform a translation, the initial and target position of the center of the image are needed, let them be $\vec{B} =< B_x, B_y, B_z >$ and $\vec{T} =< T_x, T_y, T_z >$, respectively. The direction of the translation is $\vec{D} = \vec{T} - \vec{B} =< T_x - B_x, T_y - B_y, T_z - B_z >$. To ensure the accuracy of the path, we can finish the translation in N steps, where N is determined by the total travel distance and the speed of the drones. Let V_0 be the maximum speed of the drone, and the time interval is 1 second, so the minimum number of steps required is

$$N_{min} = \frac{|\vec{T} - \vec{B}|}{V_0(\Delta t)}.$$

In practice, we can increase this number properly to slow down to guarantee the accuracy of the drone path. Assume N is the number of steps to finish the translation, the timer interval between each step is $\Delta t = 1$. We can use the following simple algorithm to generate the route of the translation:

$$\begin{aligned}
\vec{V} &= \frac{\vec{D}}{N(\Delta t)} = \frac{\vec{D}}{N} \\
V_x &= \frac{D_x}{N} \\
V_y &= \frac{D_y}{N} \\
V_z &= \frac{D_z}{N} \\
P_i(n\Delta t) &= \vec{B} + n(\Delta t)\vec{V} = \begin{pmatrix} B_x + n(\Delta t)V_x \\ B_y + n(\Delta t)V_y \\ B_z + n(\Delta t)V_z \end{pmatrix}, \text{for each i}
\end{aligned}$$

We can define a matrix

$$T = \begin{pmatrix} 1 & 0 & 0 & V_x\Delta t \\ 0 & 1 & 0 & V_y\Delta t \\ 0 & 0 & 1 & V_z\Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

to implement the translation, so

$$P_i(n\Delta t) = T^n \begin{pmatrix} B_x \\ B_y \\ B_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & V_xn\Delta t \\ 0 & 1 & 0 & V_yn\Delta t \\ 0 & 0 & 1 & V_zn\Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & V_x\Delta t \\ 0 & 1 & 0 & V_y\Delta t \\ 0 & 0 & 1 & V_z\Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix}^n \begin{pmatrix} B_x \\ B_y \\ B_z \\ 1 \end{pmatrix}$$

4.4 3-Dimension Rotation

To implement a 3-D rotation, we need to use trigonometry. The rotation in 2-D space from $< x, y >$ by an angle of θ to $< x', y' >$ can be implemented as

$$\begin{aligned}
x' &= x\cos(\theta) - y\sin(\theta) \qquad (1) \\
y' &= x\sin(\theta) + y\cos(\theta) \qquad (2)
\end{aligned}$$

Written in matrix form, we have

$$\begin{matrix} x' \\ y' \end{matrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

In 3-D space, rotations can be implemented by multiplying a 3×3 matrix. The rotation matrix in 3-D space for arbitrary axis can be created from the basic rotation matrix about x, y, and z axis. The rotational matrix about x-axis is written as:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

the rational matrix about y-axis is written as:

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & \cos\theta \end{pmatrix}$$

the rational matrix about z-axis is written as:

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ sin\theta & \cos\theta & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Rotation about x, y, or z axis can be performed by

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R(\theta) \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

, where R is one of the three type of matrix above.

For a rotation about an arbitrary axis, we need to represent the axis using two angles, (α, β), where α is the angle between the projection of the axis into the x-y plane and the x- axis, β is the angle between the axis and the z axis. The image is first rotated by an angle of α so that the axis is rotated onto xz plane. And then the whole image is rotated about the y-axis by an angle of β so that the axis coincident with the z axis. Now the actual rotation comes, and the whole image is rotated about z axis for an angle θ. The rotated image is then rotated back so that axis goes back to the initial direction. This operation can be implemented using the basic rotation matrices above. The rotation about an axis (α, β) for θ can be implemented by three steps:

$$R_{\alpha,\beta}(\theta) = R_z(\alpha)R_x(-\beta)R_z(\theta)R_x(\beta)R_z(-\alpha)$$

Assuming the rotation is finished in N steps, where N is determined by the maximum distance a drone my travel during the rotation, and the speed of the drone. In each step, the image is rotated for an angle of $\frac{\theta}{N}$.

$$\begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} = R_{\alpha,\beta}\left(\frac{n\theta}{N}\right) \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \left[R_{\alpha,\beta}\left(\frac{\theta}{N}\right)\right]^n \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

where x_0, y_0, z_0 are the initial positions of a drone, and x_n, y_n, z_n are the positions of the drone at n_{th} step.

Using the matrix above, a rotation about arbitrary axis could be implemented.

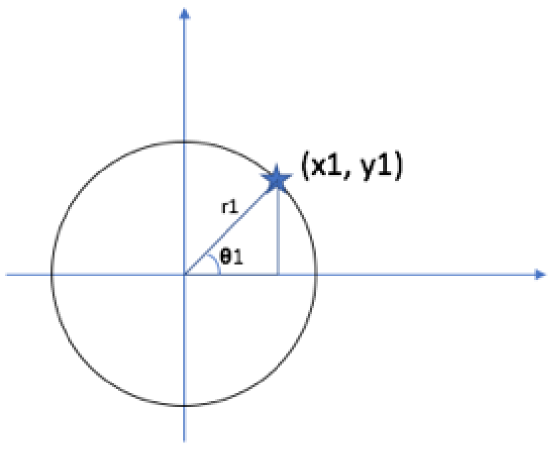
4.5 Transition between images

The transition between images is the most crucial part of the light show. In order to transform from one image to another, we need to reduce the whole transit time, and at the same time, reduce the number of potential collisions between drones. The key in the transformation is the matching of each drone in the initial image to one drone in the target image. The total transit time is determined by the drone that takes the most amount of time to finish the transition.

We have proposed our approaches to the drone matching.

4.5.1 Angle based approach in Polar coordinates

In this approach, we project the drones into a polar coordinate in the xy-plane. The original is chosen at the center of the set of drones. For each drone, we calculate the distance from its projection to the origin and the angle from the x-axis according to:

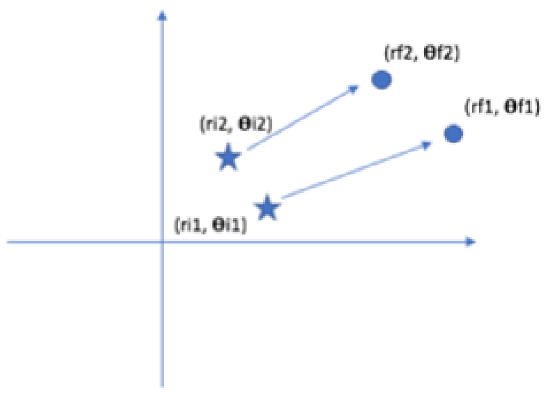


$$\begin{cases} x_i &= r_i \cos(\theta_i) \\ y_i &= r_i \sin(\theta_i) \end{cases}$$

The angles are in the range of $[0, 2\pi)$.

We try to match the drone based on their angle in the polar coordinate system. The drone in the starting image with the smallest angle should match the drone with the smallest angle in the target image, the drone in the starting image with the second smallest angle should match the drone with the second smallest angle in the target image, etc. Repeat this process until all drones are matched.

Sorted Initial Angle	Sorted Target Angle
θ_{i1}	θ_{f1}
θ_{i2}	θ_{f2}
$\dots$	$\dots$
θ_{in}	θ_{fn}



The advantage of this approach is that for each drone, it is trying to travel a smaller angle from the starting position to the target position. The route of each drone is relatively restricted in a small region, the change of collision may not be very large.

Let the initial position of a drone be (x_i, y_i, z_i), and the target position be (x_f, y_f, z_f) relative to the center of mass (x_0, y_0, z_0)

The steps for the matching are:

1. Rotate the image so that the drones form a plane that is parallel to the xy-plane using the rotation above.

2. Find the center of mass of the drones according to the centroid rule and assuming the masses of drone are all the same. Centroid of the points

$$(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$$

is calculate as:

$$\begin{aligned} X_c &= \frac{x_1 + x_2 + \cdots + x_n}{n} \\ Y_c &= \frac{y_1 + y_2 + \cdots + y_n}{n} \\ Z_c &= \frac{z_1 + z_2 + \cdots + z_n}{n} \end{aligned}$$

for both the initial image and the target image.

3. Find the coordinate of drones in the center of mass reference frame.

$$\begin{aligned} x &\to x - X_c \\ y &\to y - Y_c \\ z &\to z - Z_c \end{aligned}$$

4. Convert Cartesian Coordinates to Polar Coordinates. We have

$$\begin{cases} x_i &= r_i \cos(\theta_i) \\ y_i &= r_i \sin(\theta_i) \end{cases}$$

and

$$\begin{cases} x_f &= r_f \cos(\theta_f) \\ y_f &= r_f \sin(\theta_f) \end{cases}$$

The target positions are calculated from r_f and θ_f for each drone.

5. For each drone is the starting image and the target image, perform the above step to find their r and θ's, where $\theta \in [0, 2\pi)$

6. Sort the drone in the starting image by increasing order, and sort the drones in the target image in increasing order. When we sort the drones, we need to keep track of the index of the drone for each set of coordinate so that we can trace back which drone is it.

7. Matching the drone in order of angle. The i_{th} drone in the starting image is matched to the i_{th} drones in the target image.

8. Once the matching is done, start the transitioning procee.

9. The transition is implemented by changing the value of (r_i, θ_i, z_i) to (r_f, θ_f, z_f) for each pair of matched drones. Let N be the number of steps to transit from initial image to the target image, where N is determined by the total distance of the path and the speed of the drone. The transition process can be implemented using matrix operations. The position of the drone at the n_{th} step relative to the center of mass is determined by:

$$\begin{pmatrix} r_n \\ \theta_n \\ z_n \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \frac{r_f - r_i}{N} \\ 0 & 1 & 0 & \frac{\theta_f - \theta_i}{N} \\ 0 & 0 & 1 & \frac{z_f - z_i}{N} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_i \\ \theta_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \frac{r_f - r_i}{N} \\ 0 & 1 & 0 & \frac{\theta_f - \theta_i}{N} \\ 0 & 0 & 1 & \frac{z_f - z_i}{N} \\ 0 & 0 & 0 & 1 \end{pmatrix}^n \begin{pmatrix} r_i \\ \theta_i \\ z_i \\ 1 \end{pmatrix}$$

and

$$x_n = r_n \cos\theta_n, y_n = r_n \sin\theta_n$$

10. The matching above is performed in the center of mass system. But the center of mass of the starting image may be different from the center of mass of the target image. In this case, we can make a combination of a translation from the starting center of mass to the target center of mass and the transition from image to image. The actual position of the drone is

$$\begin{aligned} x'_n &= x_n + X_n \\ y'_n &= y_n + Y_n \\ z'_n &= z_n + Z_n \end{aligned}$$

where X_n, Y_n, Z_n is the position of the center of mass at the n_{th} step calculated from the translation from the center of mass.

4.5.2 Angle based grouping approach in Polar coordinates

In this approach, we improved the previous approach by including the r's into consideration to match drones. After we have calculated and sorted the polar coordinates of all the drones, instead of matching drones by angle, we first match drones in groups. For example, the first 10 drones in the initial image are matched to the first 10 drones in the target image, etc. Within each group, we sort the drone by the r_i or r_f in increasing order, and match the drone by the sorted values of r.

In this approach, the drone with the largest values of r_f is matched to the drone with the largest beginning r_i, which help reduce the distance that a drone travels. By using both angle and radius of drones in the matching, we have tried to reduce the amount of travel distances. The advantage of this method is that, the drone that lies in the further from the center in the initial image is matched to a drone that is further from the center in target image, which help reduce the possibility of collisions.

4.6 Combining multiple actions into one

Some times we may need to perform multiple operations into one to make the light show more attractive. For example, we can combine a translation and a transition to make the image transit to another image while moving as a whole, or we can perform a rational and translation to make the drones dance.

The combination can be performed by breaking the performance into a performance of the drones as a whole represented by the center of mass, and the performed of the drone relative to the center of mass. Then, the coordinate of the drone at each given time is calculated relatively to the corresponding coordinate system, and make a linear combination of these coordinates to generate the real path.

The determination of the center is basically to enhance the coordination system for the drones shown in our control system. By creating this center, similar amounts of drones are scattered around the center so that they could have a common axis for shaping themselves to a new image. Each drone is like a star orbiting a sun that every rotation and translation involved in the shaping is well organized. Meanwhile in our control system, we need this axis to re-create the Cartesian Coordinate and rename each point of drone to a new (x, y), it becomes fixed in the shaping of different images and basically what we do to rotate the entire image to different visions is to rotate this axis. In this case, we only need to control the axis rather than sophisticatedly control each drones' path in order to rotate the image to different visions.

We determined our center initially after launch so that the drones are well organized in the sky. In our initial image, the center of the drones remains unchanged and so the drones translate and rotate to different position according to the matched degrees we discussed in the essay. The center becomes the axis of the image once the ferris wheel has been formed and as what we could see, it is also the center of the "big circle". Then during the transformation to the dragon, the center still remained and the drones start their translation and rotation using a method base on the orbit of the center. Though the dragon is not a symmetric image, drones still follow the surrounding of the center so that the drones in the sky seems to be equally scattered around the center and audience would not need to turn their heads in each shaping of image. After the dragon, the image switch to the WING of a large fabulous bird. The center retained in the image and the drones, again, rotate and translate according to the center. The center this time is like the head of the bird and it makes the wings possible to flap like a real bird. Eventually, after all the image were showed, the drones spread out and form a square in the sky and slowly return to the ground. The center in this stage also translate from the sky to the ground and the drones surround it form a squad. By the point the whole squad returned, the center of the drones disappears and the show ends.

4.7 Collision Avoidance

Collision avoidance is a very importance issue in the route planning for the air light show. The route should be designed so that at any given time point, the distances between each pair of drones are larger than the safe distance.

There may be a few difference approaches for the collision avoidance. In this paper, we are proposing a multi-layer approach and a time delay approach.

4.7.1 Multilayer Approach

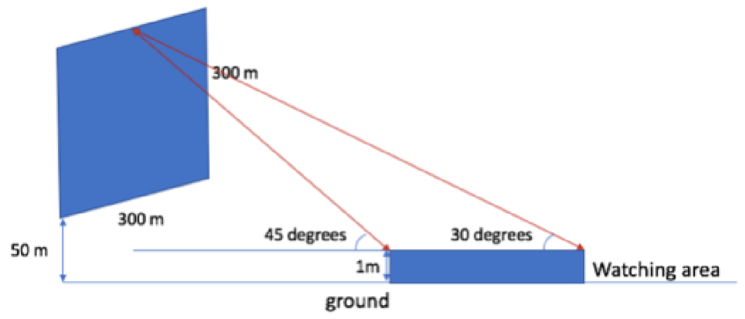
Collisions may happen when drones are too close to each other. This can be solved by grouping the drones into multiple layers separated by a short distance. The grouping can be performed randomly, or in a specific manner. The idea here is to reduce the average distances between the drones, since each layer contains less number of drones. When layering, the matching drones in the initial and target image should shift to the same direction by the same amount, so the drones in the same layer in the initial image will match drones in the same layer in the target image. Once the multi-layer transition is finished, we can make another transition to allow the drones to reach the true target position.

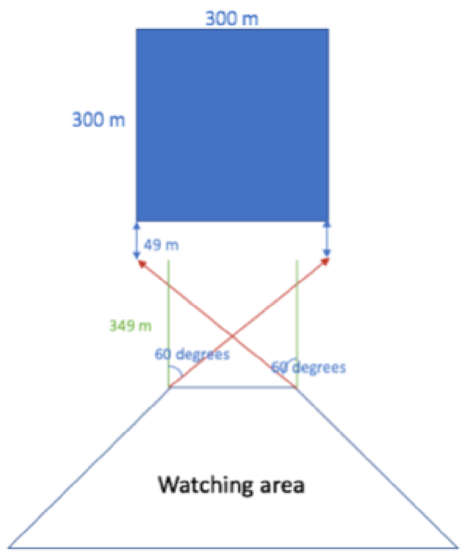
4.7.2 Time Delay Approach

The multi-player approach is actually a grouped timed delay approach, in which a group of drones delay for the same length of time together. When there are still potential collisions, we can apply an individualized time delay approach to allow one of the colliding drone to stop in the air for 1 second, allowing another drone to pass. This may cause additional drone to be stopping in the air, which should not cause problems, since the number of potential collisions may have been greatly reduced using the multilayer approach above.

4.8 Calculating the Best Watching Area

To calculate the best watching area, we need to ensure the comfortability of human vision. The angle at which the audience see the image should be in a comfortable range. Based on our research and assumption, human' s most comfortable vision is from 0 degree from horizon to at most 45 degree from horizon. We can calculate the place of the zone by using trigonometry. In order the magnify the experience of an "AIR LIGHT SHOW", we set a zone of vision from maximum 45 degrees in the front and 30 degrees. Also, we consider the height of the eyes of the audiences when they are sitting down as 1 m.





The best watching area is determined as follows:

The minimum distance from the vision zone is 349 m from the flying zone. and the maximum distance from the vision zone is $349 \times \sqrt{3} \approx 605$ m from the flying zone, giving a range of about 156 m.

For the calculation of the width of the watching area, we based on the assumption that human' s horizontal vision comfort maximum is 60 degrees to the left or right looking forward.

The width of the front row of the audiences is:

$$349 \times \sqrt{3} - 300 = 909 \text{ m}$$

The width of the back row of the audiences is:

$$909 + \frac{156}{\sqrt{3}} \times 2 \approx 1095 \text{ m}$$

In a conclusion, the best watching area is a trapezoid with 909 m as front row, 1095 m as back row and span of 156 m in depth, and the from row is 349 m away from the flying region, giving a total area of about $1002 \times 156 = 156312$ m^2 allowing about 156000 people to watch assuming there is one person in each m^2 on average.

5 Modeling the Three Images and the Requirements

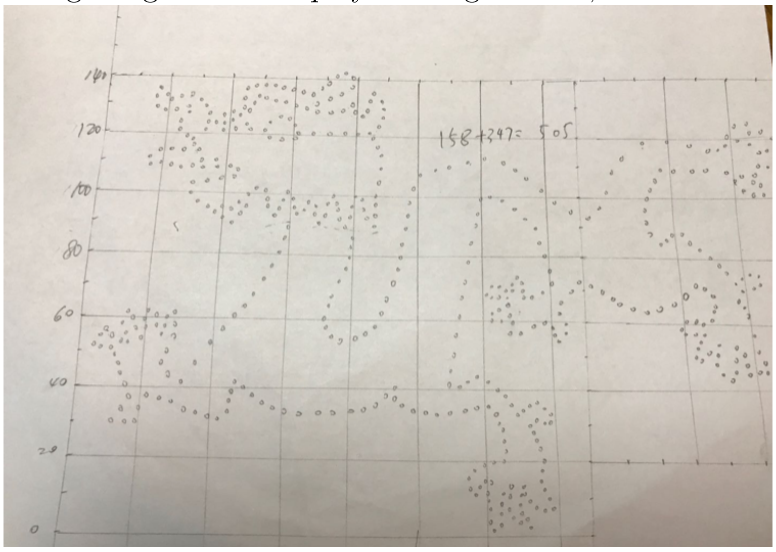
In this section we will determine the number of drones, the launch area, and the air space required for the air light show displays.

The display of dragon is the most complicated, which requires a large number of drones to make it vividly presented. So we start the calculation for the dragon display.

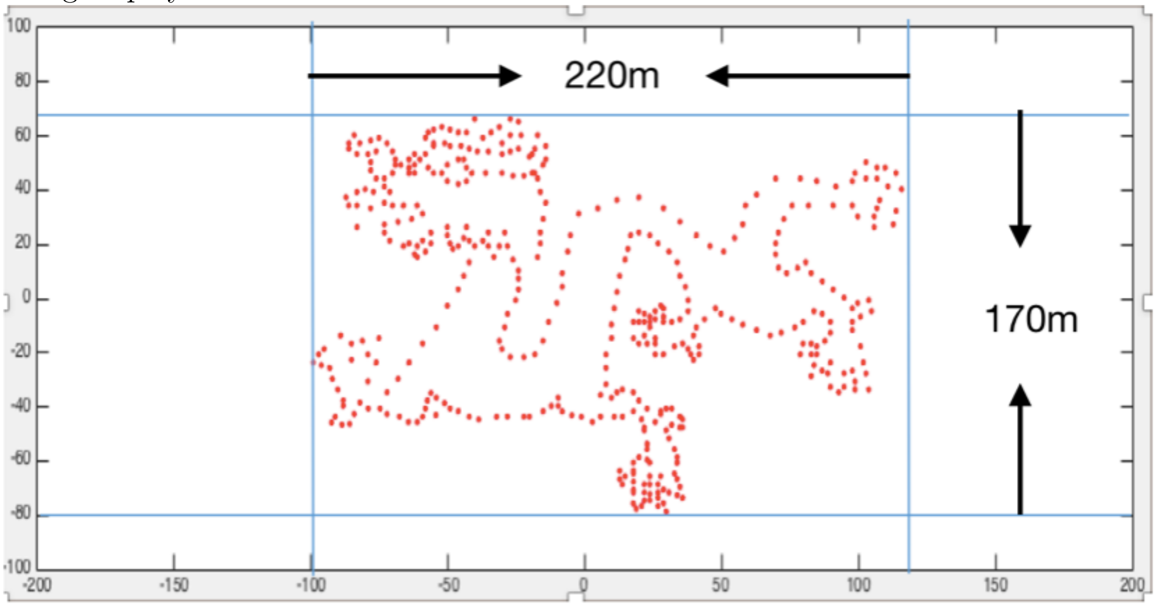
5.1 Modeling the Display of Dragon

The following is an image of the traditional Chinese dragon in Ming Dynasty and Qing Dynasty in ancient China. It is also the most popular and acceptable type of dragon all over the world.

We made a graph of the dragon by hand and selected key points from the graph. There are 505 dots in the initial hand-made graph of our dragon. Considering the safety distance between drones, and other displays had lower requirement of the number of drones, we decided to reduce the number of dots in the dragon figure without too much losing quality. Finally, we deleted 79 dots and leaving 426 dots in the fire. The actual xy-coordinates of each dot is measured and recorded. The center of mass of the figure is selected as the origin$(0, 0)$ of the coordinate system. The dragon figure was displayed using Matlab, which looks very good.



The total dimension of the dragon is 220 m × 170 m, and the distance between each pair of two drones is at least 3 m. We are going to use 426 drone in each of the following display.



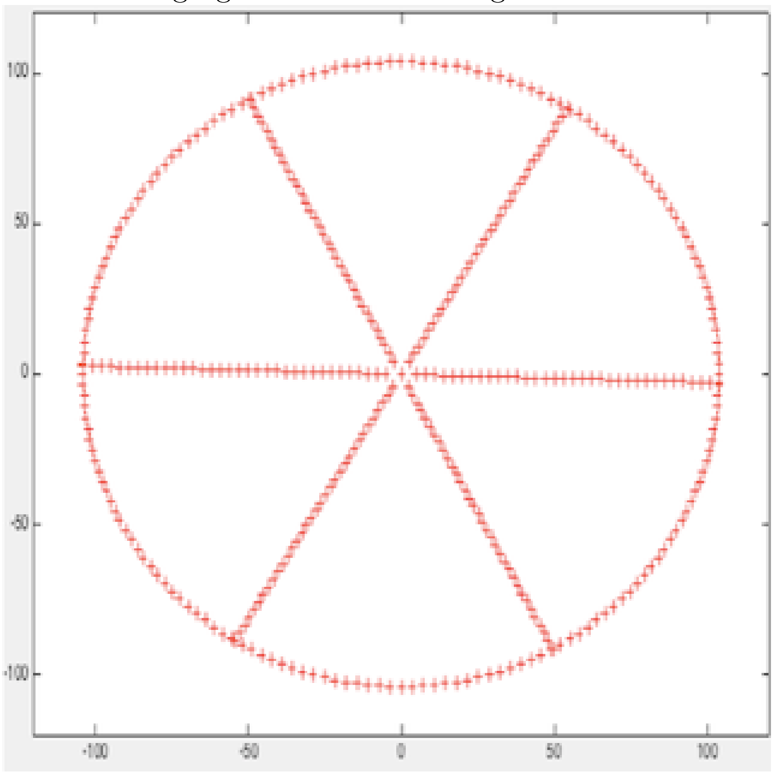
5.2 Modeling the Display of Ferris Wheel

The ferris wheel can be creating using computer program instead of drawing by hand.

In this part, we are going to determine the radius, the number of spokes, the number of drones on the circumference, and the number of drones on the spokes in the Ferris Wheel.

We decided to have a Ferris wheel Display with a diameter about the same as the length of the dragon, so the radius should be about 110 m and the distance between drones is 3 m. The circumference of the circle is $2\pi r$ which is about 690 m, so about 230 drones are required on the circumference. For each spoke in the ferris wheel with a length of 110 m, about 36 drones are required. With the remaining 196 drones, about 6 spokes are needed to arrange the 196 drones. With a minor adjustment, we determined to use 220 drones on the circumference, and 34 drones on each spoke, and two drones near the center of the wheel. The total number of drones is $220 + 34 \times 6 + 2 = 426$.

The following figure is the final image of the ferris wheel.

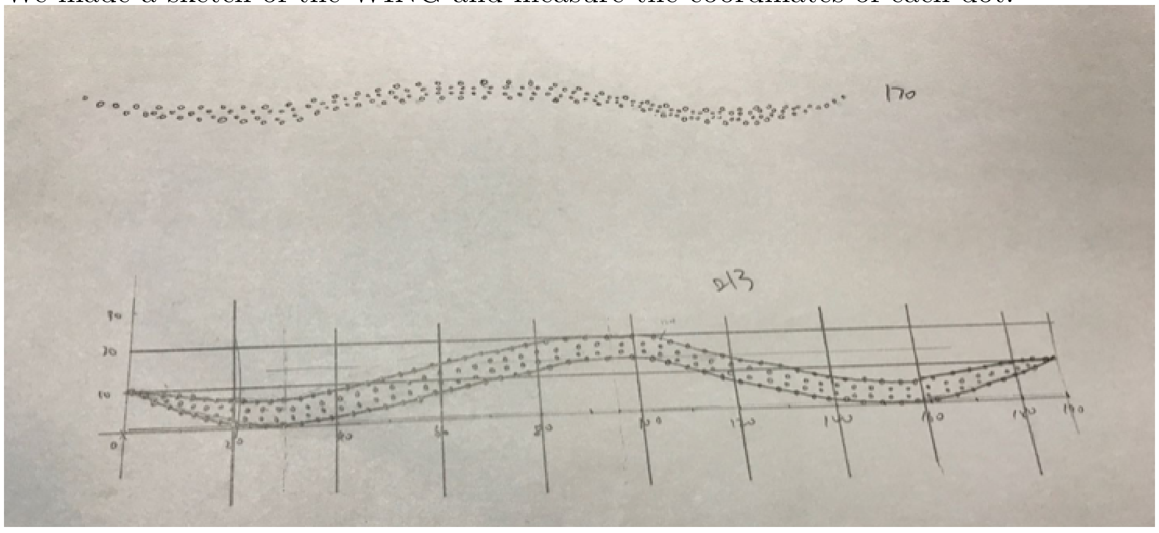


5.3 Modeling the Display of Shenzhen Citizen Center

We get the idea of our third graph by the landmark of our city: Shenzhen Citizen Center.

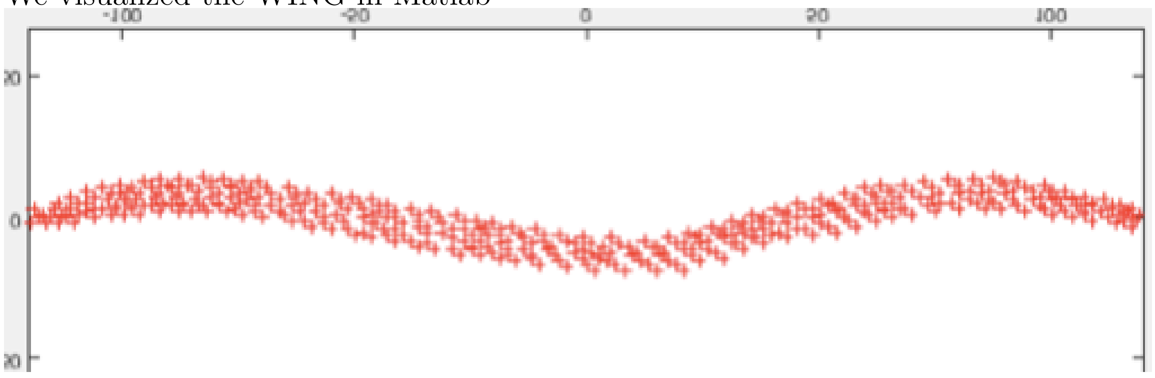
Here is a photograph of the building. It is a large fabulous bird waving its wing and it represents the hope and improvement of the city.

We made a sketch of the WING and measure the coordinates of each dot.



In order to use up 426 drones, the WING should have a dimension about 300 m × 30 m with two layers with different z values close to each other.

We visualized the WING in Matlab

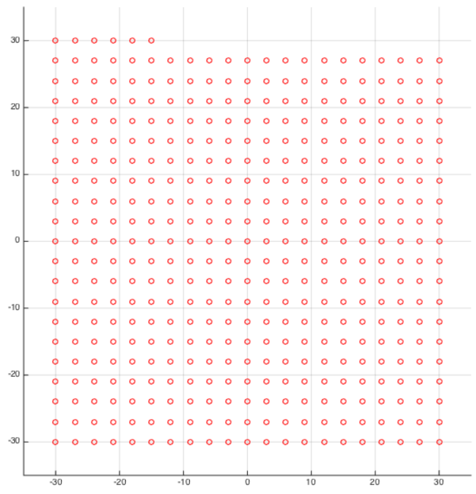


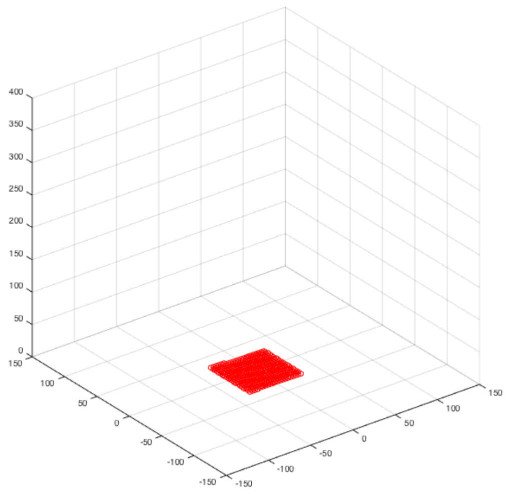
In order to create a more vivid 3-D display, we need to properly set the z coordinate for each drone, which may take a much longer time. So in this paper, the initial images of ferris wheel and dragon were created as 2-D images, while the WING is created as a two layer 3D image For all the three displays, we convert the x and y coordinates centered at $(0, 0)$.

5.4 Requirement of the Launch Area in total

Since we have 426 drones, we plan to arrange them into a 21 × 21 square grid with 441 vacant spaces. The distance between each grid is designed as the minimum safe distance of 3 m, and some necessary margin and work area are also needed. The the required launching area should be about 70 m×70 m = 4900 m^2.

The following graphs show a 2D view of the launch area including the initial arrangement of drone and a 3D view of the launch area in the same scale as the flying zone.





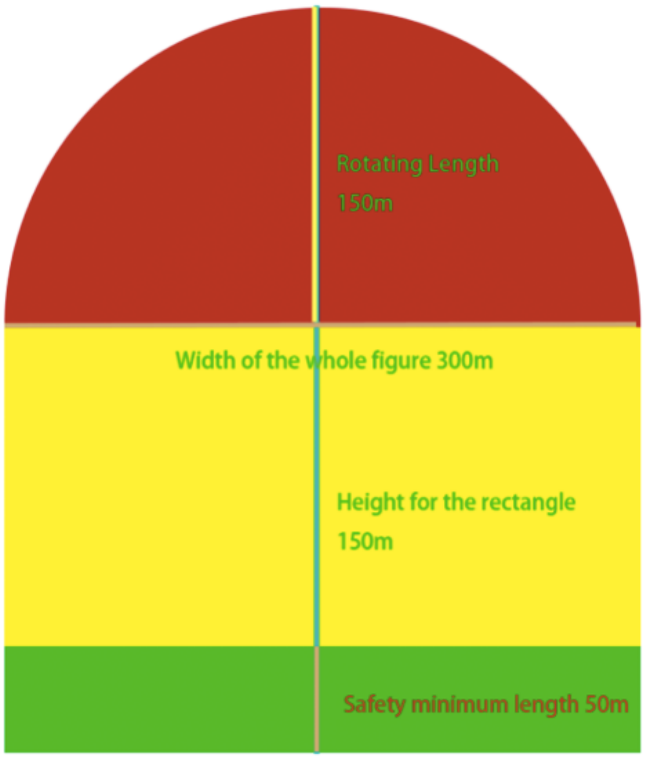
5.5 Requirement of the Total Aerial Space

During the air show, the space that the drones may reach should be clear of any object. To guarantee the safety of the show, we also set a 50 m for the minimum height for drones during the show, except during the launching.

The air space consists of three parts:

1. Cylindrical region with Height below 50 m and radius of 150m: a reserved safety zone,
2. Cylindrical region with Height between 50 m and 200 m and radius of 150m: region for the center of the display.
3. Hemisphere region: Centered at height of 50 m with a radius of 150 m: The region for the rotation of the displays.

The maximum height of the required space is 350 m.



That's the 2-dimension picture of the semi-sphere and the cylinder. The cylinder has a height of 200 meters which equal the height of the safety zone and the actual region of the shape, and the semi-sphere's radius is 150 meters.

6 Model Verification

6.1 Time Duration for Each Action

The total time spent could be calculated in several parts depending on the schedule of the displays.

1. Launching to form Ferris Wheel: We set the initial height of ferris wheel as 200 m, considering the drones are not going directly upward, the total distance that drone travel should be larger than 200 m. Consider the drones on the circumference of the ferris wheel, the distance it travels is about $\sqrt{110^2 + 200^2} = 230$ m, so we estimated the time for the launching to form ferris wheel to be 77 seconds.

2. Launching to form dragon: Similarly, we set the initial height of the dragon as 200m, the maximum distance a drone flies is the same. So we estimated the time for the launching to form dragon to be 77 seconds.

3. Launching to form WING: Similarly, we set the initial height of the WING as 200m, the maximum distance a drone flies is about $\sqrt{150^2 + 200^2} = 250$ m, so we estimated the time for the launching to form dragon to be 84 seconds.

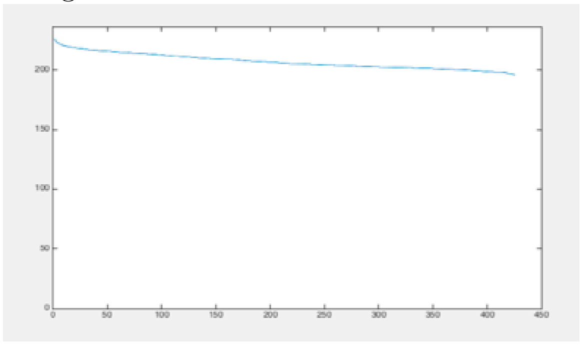
4. Landing from Ferris wheel: same as launching:

5. Landing from Dragon: same as launching:
6. Landing from WING: same as launching:
7. Rotation of Ferris Wheel: With a radius of 110 m, the circumference is about 690 m, so it takes about 230 seconds to finish
8. Rotation of Dragon: same as launching: With a radius of 110 m, the circumference is about 690 m, so it takes about 230 seconds to finish
9. Rotation of WING: same as launching: With a radius of 150 m, the circumference is about 942 m, so it takes about 314 seconds to finish

For the transition between images, we used computer program to emulate the transition and calculate the longest displacement a drone may travel:

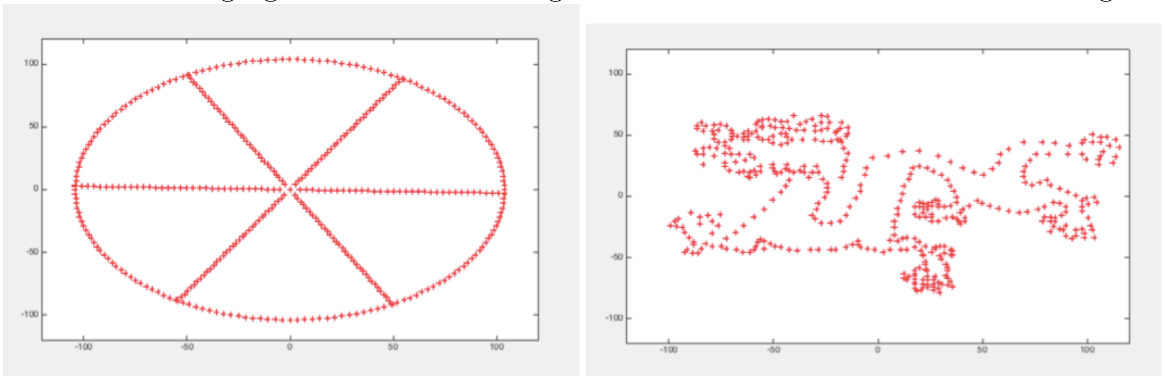
1. Transition between Ferris Wheel and Dragon: The maximum distance a drone travels is about 99 m, which requires 33 seconds to finish.
2. Transition between Ferris Wheel and WING: The maximum distance a drone travels is about 143 m, which requires 48 seconds to finish.
3. Transition between WING and Dragon: The maximum distance a drone travels is about 146 m, which requires 49 seconds to finish.

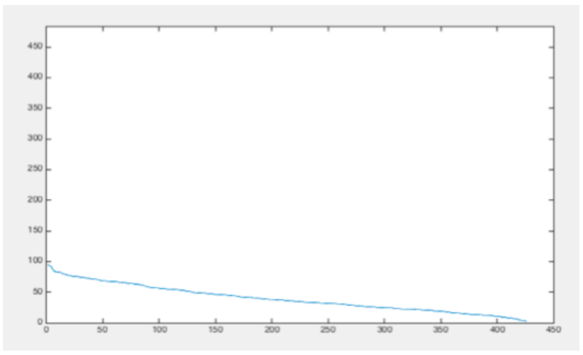
The following figure shows the timing for the launching of drones to form a WING at a height of 200m.



The total distance travel by all the drone were sorted in descending order. The distance for the drone traveling the most is draw on the left end of the graph. The travel distances traveled by drone are close to each other, since the distance are from the ground to a height of 200 m, which does not have too much variance.

The following figure shows the timing of a transition from ferris wheel to dragon.





The total distance travel by all the drone were sorted in descending order. The distance for the drone traveling the most is draw on the left end of the graph. The travel distances by drone spread to a relatively larger distance, because during a transition, the route for each drone may differ significantly.

In summary, the time for each action is listed below:

Action	Distance	Time
Launching to form Horizontal Ferris Wheel	230	77
Launching to form Horizontal Dragon	230	77
Launching to form Horizontal WING	250	84
Landing from Horizontal Ferris Wheel	230	77
Landing from Horizontal Dragon	230	77
Landing to form Horizontal WING	250	84
360° Rotation of Ferris Wheel	690	230
360° Rotation of Dragon	690	230
360° Rotation of WING	942	314
Transition between Ferris Wheel and Dragon	99	33
Transition between Ferris Wheel and WING	143	48
Transition between WING and Dragon	146	49

Since the total battery time of a drone is 20 minutes, we can make combinations of displays and actions to fill the time. A possible combination may be:

Action	Approximate Time
Launching to form a Ferris Wheel	77 seconds
Rotate the Ferris Wheel to a near vertical direction (90°)	60 seconds
Rotate the Ferris Wheel in vertical direction for 180°	115 seconds
Transfrom from Ferris Wheel to a Dragon	33 seconds
Rotate the Dragon in vertical direction for 180°	115 seconds
Transfrom from Dragon to WING	49 seconds
Rotate the WING in vertical direction for 180 °	157 seconds
Transfrom from WING to Dragon	48 seconds
Transfrom from Dragon to Ferris Wheel	33 seconds
Landing from Ferris Wheel	77 seconds
Total Time	765 seconds

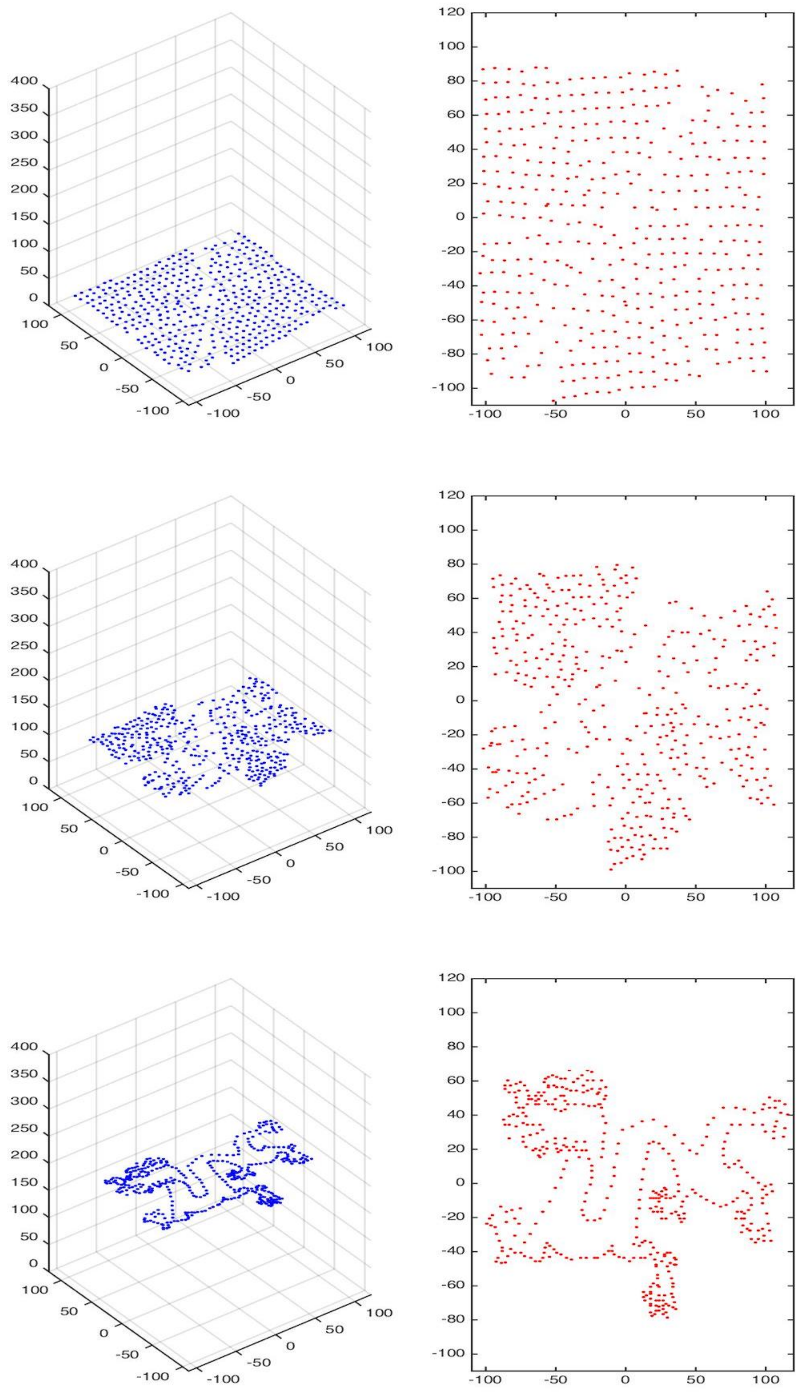
The table above is a sample arrangement of the displays for about 13 minutes. We can add a few seconds between two displays to make better impress to the audience.

For safety reasons, we do not want to schedule the displays to a full 20 minutes.

The actual arrangement of the displays can be determined before the air show with the help of the timing table above.

6.2 Testing the Launching to Form a Image

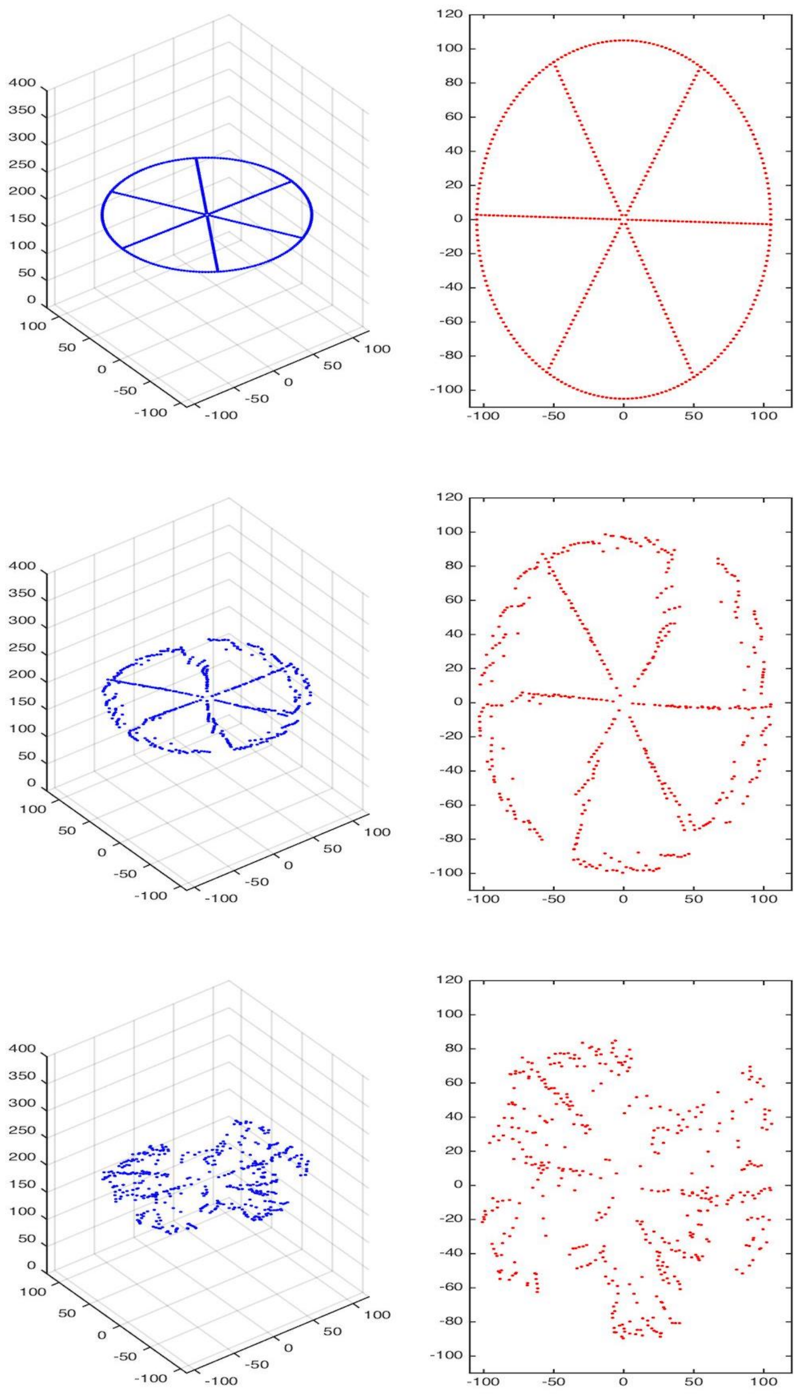
The following pictures illustrated the process of launching drones from ground to form a dragon in the sky.

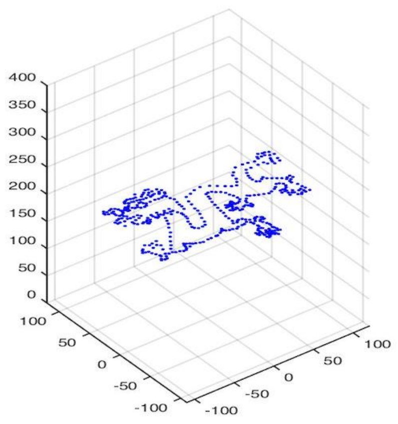


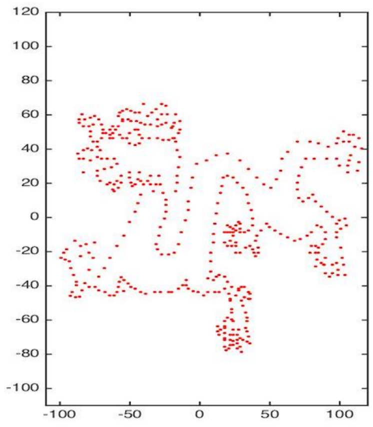
6.3 Testing the Image Transforming method

The follo w.i

Lng pictures illustrated the process of image transforming from a ferris wheel to a dragon.

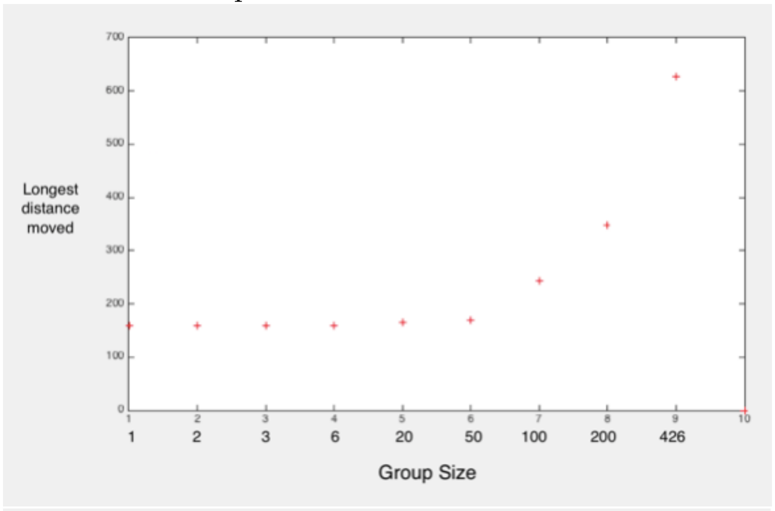


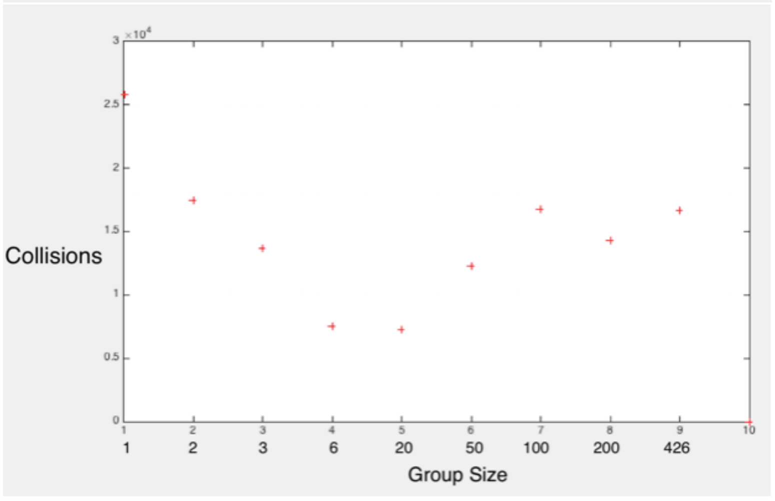




6.4 Testing the Influence of the Group Size in the Image Transforming method

In the image transforming method, we can use different group size for the drones and compare the maximum distance traveled and the number of potential collisions. We test one transformation and collected the maximum distances a drone may travel and the total number of potential collisions.





The grouping size of 1 is equivalent to the simple angle based method without considering the r values of each drone. The group size of 426 is the same as only considering the r values of drones in the matching.

The figure above shows the maximum travel distance and number of potential collisions that may happen during the transition. From the figure we can see that when grouping size is small, the maximum distances are almost the same, but the number of collisions is decreasing. When the group size becomes larger, the maximum travel distance becomes significantly larger, because in this case, there may be drones that fly over a very large angle to its destination. The number of collisions is also becoming larger for larger group size, since the larger group size makes the drone fly over a larger range and in a near random manner. The travel distance and the number of collision were near the best value for a group size between 6 and 20, so we plan to choose group size to be 10 in the further test.

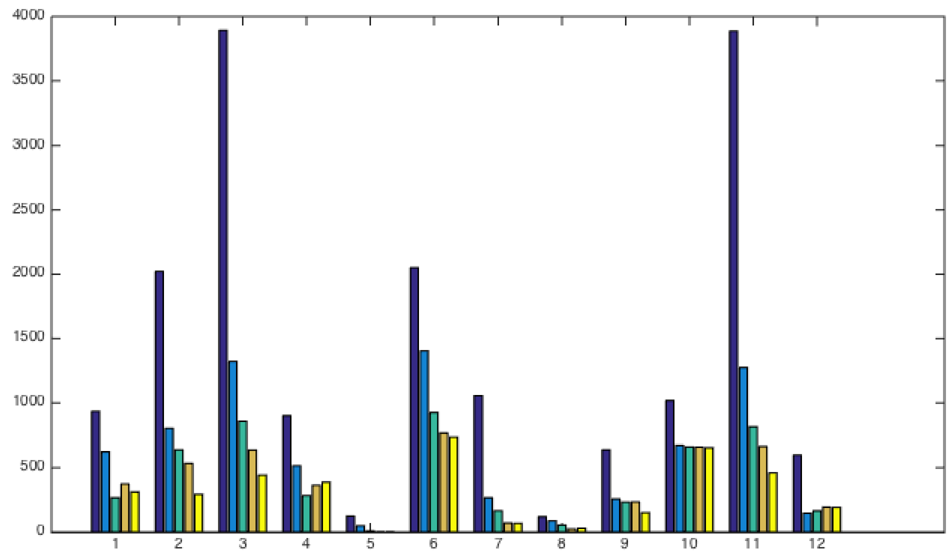
6.5 Testing of the Collision Avoidance Method

We also tested our collision avoidance method by using difference number of layers. During each step of the transitions, we count the number of pairs of drones whose distance between each other is too close to each other. We use 1 m as a threshold to count the number of collisions. We may allow the drones to be less than the safety distance from each other temporarily. We tested the launching, landing, and transition between each pair of images facing the same direction and centered at the same position with 426 drones in the image. The transitions were implemented using 100 steps, and the distance between each layer is 2 meters. The used 3, 5, 10, 20 layers, respectively. For example, when 3 layers are used, 426 random numbers are selected at random from $\{0, 1, 2\}$. If the number of 0, the corresponding drones did not move. If the number of 1, the corresponding drones will move towards the normal direction of the image plane for 2 meters. If the number is 2, the corresponding drones will move towards the normal direction of the image plane for 4 meters, etc. Thus, the drone were separated into 3 layers, with 2 meters between each layer.

The results are listed in the following table.

Table 1: The number of collisions during each action

Index	Start	Target	No Avoidance	3Layers	5Layers	10Layers	20Layers
1	Ground	Ferris Wheel	934	623	266	373	311
2	Ferris Wheel	Dragon	2022	802	638	533	292
3	Ferris Wheel	WING	3892	1322	857	637	442
4	Ferris Wheel	Ground	934	515	283	363	387
5	Ground	Dragon	123	48	9	2	3
6	Dragon	Ferris Wheel	2051	1404	925	766	733
7	Dragon	WING	1055	266	165	69	67
8	Dragon	Ground	119	86	54	23	29
9	Ground	WING	638	256	232	233	150
10	WING	Dragon	1018	671	660	659	654
11	WING	Ferris Wheel	3886	1275	814	664	459
12	WING	Ground	598	146	164	193	191



From the results above, we can see that our collision avoidance method reduced the number of possible collision greatly. In most of the cases, with 5 layers, the number of possible collisions is reduced to less than 2 for each drone, and some times, the number of collision is reduce to less than 1 for each drone. The actual number may be even smaller since some collisions may be counted more than once.

We do not have time to test the time delay method, which will even reduce the number of possible collisions further.

The collision avoidance method can be improved by a better drone matching method.

7 Safety consideration

1. Propeller guards:

   Considering the possible collisions made between drones and between birds and drones, we installed propeller guards on all the drones involved in the show so that the propellers won' t be damaged and the drones won' t fall over due to collision.

2. Routine check: Also, in order to ensure that there is no malfunction in each drone, we test all the drones before the show and do routine check to make sure the drones are in good operation.

3. Launching platform: Having a launching platform could ensure the no resistance such as grass or mud interference which disrupts the launching. And it also makes sure a calm and flat launching space is obtained to launch each drone fairly.

4. Obeying the maximum height decided in the local law: Due to the permission of flight stated in the law, the maximum flight height for drones is 400 m. Considering this law, our maximum height for flight is 350 m. If the some drones fly over 400 m high, our technicians will immediately maneuver these drones to lower than 400 m.

5. Setting up the Prohibition zone: Since there might be accident of drone falling over because of hitting or malfunctioning, we established a circular prohibition area according to the Horizontal Projectile Motion (the furthest distance of falling) formula. So that we can calculate the horizontal falling of the highest drone in its maximum speed (means that it goes the furthest in horizontal projectile).

   According to the formula of Horizontal Projectile Motion:

   $$\begin{aligned} H &= \frac{1}{2}gt^2 \\ S &= V_0 t \end{aligned}$$

   where S is the Horizontal distance travelled, V_0 is the initial speed, t is the time needed for the drone to fall to the ground, H is the vertical height, g is the gravitational constant. The equations can be simplified to

   $$S = V_0\sqrt{\frac{2H}{g}}$$

   Using cruising speed $V_0 = 3$ m/s, the maximum height of drones $H = 350$ m, and $g = 9.8\text{m}/s^2$, we can calculated the value of S as 25.355 m. So the the prohibition zone for watching should have a larger radius than the flying region, with an addition 26 m in radius. The distance from the zone for the audiences far more than this distance, so the accidental landing of drone would not post potential harm to the audiences.

8 Advantages and Disadvantages

Our model is easy to be used to model 3D drones and make 3D movements and transformations. The transforming from one image to another image is implemented in a simple but elegant manner. The transition method has included the consideration of relative position of each drone in the center of mass system to match the drone proper. Experiment shows that by proper grouping, the maximum travel distance of the number of collisions can both be close to a lower level.

Our collision avoid method further reduce the number of collisions to a minimal level to reduce the potential harm to the drone. By using our two collision avoidance methods together, it is possible to reduce the number of collisions to near 0.

Our modeling and implementation of the show can make to show in a fluency manner and may generate some very attractive effects.

Also, by doing safety consideration and giving a regulation of appropriate watching area, we make sure the customers or audience are in the best vision in this show.

Moreover, our fanciest designs of the three patterns involved in the show could attract a lot of people to watch our show, so that the economy effect of our show is huge.

However in our design, we did not consider to automatic process of creating 3D images other than the laying in the collision avoidance method. If images can be created to show more 3D effect, the show may even be more exciting.

In our model, we have assumed to weather is good and the wind does not blow fast. So, we assume the accuracy of the positions of the drones.

We also assumed that all drone would fly normally. In reality, we should consider the possible consequence of accidence on the drones. The collision avoidance method should still be improved on transitions between complicated images.

Most importantly, the whole air show are predetermined. There is no flexibility in the order of images, actions, etc. Everything is determined. In a much stronger model, drones should cooperate between each other without a central control, and drone should be able to make new displays with no pre-calculation.

In a nut shell, bringing all the advantages together, the show we designed will make not only huge income but also great word-of-mouth of this typical show. Therefore, after this first tryout of the show, we could improve the drones and plan another round of show next year, to generate continuous profit.

9 Memo to the Mayor

Dear Mayor,

We are honored for the opportunity you gave us to investigate the idea of using drones to create three displays in the 40th anniversary of the city. We are very pleased to report our plan of the design of the show and hope it will enrich the festival and impress the audience involved in the show.

During our research and investigation, by judging the availability of drones, spaces and resources and the aesthetics of the show, we designed three iconic patterns for the show: the Ferris Wheel, the Dragon, and the WING of a large fabulous bird which represents romance, gallantry and prosperity, an image from our city center, which represents exactly our city spirit.

After our mathematical analysis and calculation, we determined that we may need 426 drones for the show to come to reality, because we found that 426 is a compromise of the lower limit for a vivid show and the economic considerations. The land we needed for launching is about 4900 m^2 so that the drones can launch in a good order and collisions with each other could be avoided while launching. Also, the air space we needed in the sky is roughly a 300×300 with maximum height of about 350 m, so that audience can experience the majesty of the show. It seems that our city do have sufficient condition for all these requirements. We can set the launch area in the Stadium near the sea, and to present the air show there. The audience could stand or sit on the grass land near the stadium, or on the beach, which is in the best watching area. With the skyline of our city as the background, the show much become a tremendous memory for every audience.

Although our air show consists of so many drones flying at the same time, you do not need to worry about the control of these drones. Because all drones are centrally controlled by only one pilot - the computer program. We only need a single person to operate the program to control all the drones to exhibit all different wonderful displays. You probably will wonder that is it safe to have so many drones flying in the sky, will it be a mess up? I would tell you that, all the path of the drones were pre-determined, which means, we know exactly where are the drone at any time, and the possible collisions between drone has been avoided by our very nice methods. The drones will not be a mess up, instead, they will be more like a wonderful dancer dancing in the sky with beautiful lights flashing.

As to the arrangement of the show, you will received a menu from us, showing the timing of each action. These actions include launching from ground, forming images, translating image into your preferred direction, rotating the image for any number of degrees, transition between your selected images, landing, dancing in the sky, etc. You can also design your lighting scheme to control the lights of the drone to make more variety of imaging effects to make the show more of fun. You are able to choose from those options to create a sequence of displays, actions, and transition effects, at your preferences, to make a show for up to 15 minutes, just like ordering food in a restaurant. We are going to present you with the emulated air light show similar to a movie, which is very close to the true effects with our powerful emulating software, so that you can refine the effects of the air show. It is just this easy. We have tested all possible displays and actions to guarantee the show to come into reality.

Mr. Mayor, I am so excited that our city could have such a wonderful air show to celebrate our city's 40th anniversary. The terrific show would be able to create a unique image of our city in the world. You know, we are the central city of high technology in China, with so many prosperous new ideas coming true, with Dajiang as the best example. Dajiang is now the top company in the world in designing and manufacturing unmanned flight vehicles, which have been applied to all aspects in our life. So a successful air light show would definitely make the company and our city more exposed to the whole world. With these ideas in mind, I am even more excited, especially, when think about that our plan could help our city to gain more focus.

My Mayor, I am here, waiting for a dream air light show to come true, come true as soon as possible. I do not want to wait for two years, I want to have it tomorrow.

Sincerely, Team #8028

10 appendix

10.1 References

1. Intel shooting star fact sheet

   https://newsroom.intel.com/wp-content/uploads/sites/11/2017/07/Intel-Shooting-Star-Tech-Fact-Sheet-073117-1.pdf

2. Coordinate transformation

   https://en.wikipedia.org/wiki/List_of_common_coordinate_transformations

3. Center of drones

   http://www.mathwords.com/c/centroid_formula.htm

4. Safety consideration

   http://www.dji.com/cn/flysafe?site=brandsite&from=insite_search

5. Horizontal Projectile Motion–https://baike.baidu.com/item/%E5%B9%B3%E6%8A%9B%E8%

10.2 Program

The emulation was implemented using Matlab. The code is attached below:

```
clear x2;
clear y2;
clear z2;

R = 120;
H = 400;

me = mean(dragon);
dragon(:,1) = dragon(:,1) - me(1);
dragon(:,2) = dragon(:,2) - me(2);

me = mean(DPZC);
DPZC(:,1) = DPZC(:,1) - me(1);
DPZC(:,2) = DPZC(:,2) - me(2);

clear x0;
clear y0;
clear z0;

x0(1:426)=0;
y0(1:426)=0;
z0(1:426)=0;

%Creating Start positions
```

```
for i=0:425
    x0(i+1) = 5 * mod(i,21)-50;
    y0(i+1) = 5 * floor(i/21)-50;
    z0(i+1) = 0;
end

%Ferris wheel
DrNum = 180;
R1 = 104;
x2 = R1*cos( 2*pi/DrNum * ( 1:DrNum ) );
y2 = R1*sin( 2*pi/DrNum * ( 1:DrNum ) );
z2 = 200 * ones(1,n);

x3(1:204) = 0;
y3(1:204) = 0;
z3(1:204) = 0;

index = 1;

FuTiao = 6;
for i=1:FuTiao
    for r=5:3:R1
        x3(index) = r * cos( 2*pi/FuTiao * i - pi/120);
        y3(index) = r * sin( 2*pi/FuTiao * i - pi/120);
        z3(index) = 200;
        index = index + 1;
    end
end

x2 = [x2 x3 zeros(1,426-size(x2,2)-size(x3,2))];
y2 = [y2 y3 zeros(1,426-size(y2,2)-size(y3,2))];

x = x0;
y = y0;
z = z0;

n=10;
%take off
[x, y, z] = trans2(x0,y0,z0,x2,y2,z2,[-R, R, -R, R, 0, H], 0.01, 100, n);
%Rotate 1
angle = pi/180 * 0.5;
A = [ 1 0 0 ; 0 cos(angle) -sin(angle); 0 sin(angle) cos(angle)];
[x, y, z] = rotate(x,y,z, A, [-R, R, -R, R, 0, H], 180, 0.02 );
%Rotate 2
angle = pi/180;
A = [  cos(angle) 0 -sin(angle); 0 1 0; sin(angle) 0 cos(angle)];
[x, y, z] = rotate(x,y,z, A, [-R, R, -R, R, 0, H], 180, 0.05 );
```

```
%to Dragon
index = 1;
sumDist(1:10) = 0;
ccount(1:10) = 0;
[x, y, z, ~,~] = trans2(x,y,z,dragon(:,1)',dragon(:,2)', 200*ones(1,426),
                                  [-R, R, -R, R, 0, H], 0.01, 100, n);

%to DPZC
dz = floor( 3 * rand(1,426)) * 2 - 4;
z = z + dz;
[x, y, z, sum1, cc] = trans2(x,y,z,DPZC(:,1)',DPZC(:,2)',200*ones(1,426) + dz,
                                  [-R, R, -R, R, 0, H], 0.01, 100, n);
sumDist(index) = sum1;
ccount(index) = cc;

%LANDING
[x, y, z, sum1, cc] = trans2(x,y,z,x0,y0,z0,[-R, R, -R, R, 0, H], 0.001, 100, n);
 8.3 Function 1(Translation):
function [x2, y2, z2, maxs, ccount] =
            trans2( x1, y1, z1, x2, y2, z2, ax, delay, m, groupsize )

    selected = [ 1 ];

    n = size( x1,2 );
    ccount = 0;

    r1 = sqrt( x1.^2 + y1.^2);
    theta1(1:n) = 0;

    for i=1:n
        if( x1(i) ~= 0 )
            theta1(i) = atan( y1(i)/x1(i) ) +
                pi*(x1(i) < 0) + 2*pi * ( (x1(i) >= 0) * ( y1(i) < 0 ) );
        else
            if( y1(i) > 0 )
                theta1(i) = pi/2;
            else
                theta1(i) = 3*pi/2;
            end
        end
    end

    tt(1:n) = 0;
    for i=1:n
        tt(i) = n - sum( theta1 > theta1(i));
    end
    r2 = sqrt( x2.^2 + y2.^2);
   theta2 = atan( y2 ./ x2 ) + pi * ( x2 < 0) + 2 * pi * ( (x2 >= 0) .* ( y2 < 0 ) );
    for i=1:n
        if( x2(i) ~= 0 )
            theta2(i) = atan( y2(i)/x2(i) ) +
                        pi*( x2(i) < 0) + 2 * pi * ( (x2(i) >= 0) * ( y2(i) < 0 ) );
        else
            if( y2(i) > 0 )
                theta2(i) = pi/2;
            else
                theta2(i) = 3*pi/2;
            end
        end

    end

    tt1 = [ theta1' r1' z1' (1:n)'];
    tt2 = [ theta2' r2' z2' (1:n)'];

    tt1 = sortrows( tt1, 1);
    tt2 = sortrows( tt2, 1);

    theta_target(1:n) = 0;
    r_target(1:n) = 0;
    z_target(1:n) = 0;


    for i=1:ceil(426/groupsize)
        N = groupsize;

        if( N * i <= 426 )
            tt1s = tt1( N*i-N+1: N*i,:);
            tt2s = tt2( N*i-N+1: N*i,:);
        else
            tt1s = tt1( N*i-N+1:426,:);
            tt2s = tt2( N*i-N+1:426,:);
        end

        tt1s = sortrows( tt1s, 2);
        tt2s = sortrows( tt2s, 2);

        for k=1:size(tt1s,1)
            theta_target( tt1s(k,4) ) = theta2( tt2s(k,4) );
            r_target( tt1s(k,4) ) = r2( tt2s(k,4) );
            z_target( tt1s(k,4) ) = z2( tt2s(k,4) );
        end
```

```
    end



    r = r1;
    t = theta1;
    z = z1;

    sum2(1:426) = 0;

    x0 = r .* cos(t);
    y0 = r .* sin(t);
    z0 = z;

%     ratio = 0.9;

    for i=1:m

%        r = ratio * r + (1-ratio) * r_target;
%        t = ratio * t + (1-ratio) * theta_target;
%        z = ratio * z + (1-ratio) * z_target;

       r = r + (r_target-r1)/m;
       t = t + (theta_target-theta1)/m;
       z = z + (z_target-z1)/m;

       x = r .* cos(t);
       y = r .* sin(t);

       delta = [x-x0; y-y0; z-z0];
       dist = sqrt( sum(delta .^2 ) );
       sum2 = sum2 + dist;
       x0 = x;
       y0 = y;
       z0 = z;

       subplot(2,2,1);
       scatter3( x(selected), y(selected ), z(selected ), 'b', '.' );
       hold on
       axis(ax);

       subplot(2,2,2);
       plot( x, y, 'r+');
       hold off
       axis(ax([1 2 3 4]));

       subplot(2,2,3);
```

```
%       plot( x, z, 'r+');
%       hold off
%       axis(ax([1 2 5 6]));
        plot( i, ccount, 'b.');
        hold on
        axis([0 m 0 2000]);

       subplot(2,2,4);
%       plot( y, z, 'r+');
       hold off

%       axis(ax([3 4 5 6]));

        plot( sum2);
       axis([0 450 0 300]);

       pause(delay);

       ccount = ccount + numOfCollision(x,y,z);
    end

    sum2 = sort(sum2, 'descend' )
    plot( sum2);
    axis([ 0 450 0 max(sum2) + 10]);

    x2 = r .* cos(t);
    y2 = r .* sin(t);
    z2 = z;
    maxs = max(sum2);
    pause(3);
    subplot(2,2,1);
end

function [x, y, z] = rotate(x,y,z,A,ax, m, delay)

    w = [x; y; z];

    for i=1:m
        c = mean(w,2);

        w(1,:) = w(1,:) - c(1);
        w(2,:) = w(2,:) - c(2);
        w(3,:) = w(3,:) - c(3);

        w = A * w;

        w(1,:) = w(1,:) + c(1);
        w(2,:) = w(2,:) + c(2);
        w(3,:) = w(3,:) + c(3);

        subplot(2,2,1);
        scatter3( w(1,:), w(2,:), w(3,:) );
        hold off
        axis(ax)

        subplot(2,2,2);
        plot( w(1,:), w(2,:), 'r+' );
        hold off
        axis(ax([1 2 3 4]));

        subplot(2,2,3);
        plot( w(1,:), w(3,:), 'r+' );
        hold off
        axis(ax([1 2 5 6]));

        subplot(2,2,4);
        plot( w(2,:), w(3,:), 'r+' );
        axis(ax([3 4 5 6]));
        hold off

        pause(delay);
    end
    x = w(1,:);
    y = w(2,:);
    z = w(3,:);

end

function count = numOfCollision( x, y, z)
    n = size(x,2);
    count = 0;

    for i=1:n
        for j=1:n
            if( i ~= j )
               d = [x(i) - x(j);y(i) - y(j);z(i) - z(j)];
               dist = sqrt(d'*d);
               if( dist < 1 )
                   count = count + 1;
               end
            end
        end
    end
end
```