For office use only	Team Control Number	For office use only
T1 ________________		F1 ________________
T2 ________________	7359	F2 ________________
T3 ________________	Problem Chosen	F3 ________________
T4 ________________	A	F4 ________________

2017HiMCMSummary Sheet.

Drone Clusters as Sky Light Displays

Since Intel Corporation achieved the Intel® Shooting Star ™ formation flight show and created a stunning nighttime aerial landscape using drones, the drone cluster, with its precise control of the graphic shave received more and more attention due to its wide applications. In response to the request of the mayor of the city, this article designed and implemented a drone control system to meet the needs of celebrating the upcoming New Year. In this design, the fleet of 528drones show the dynamic formation and rotation of the Sky Ferris wheel, the mighty night sky dragon dance, the night sky New Year's Eve countdown and finish the 10-minute show with "2018" word pattern in the air.

In the process of program design, this dissertation deeply studies the problems of fixed point analysis, flight path analysis, 3D trajectory analysis, and Anti-collision release safety analysis. The simulation of the dynamic lift-off and rotation of the wheel is achieved by using the gravity wheel and the observation wheel as the prototype, and calculating the coordinates of each point of the wheel and the trajectory of plane motion and the pre-path design; The dragon skeleton model, dragon skin attachment point model and three-dimensional rotation model are constructed to realize the movement of the dragon in two-dimensional space and three-dimensional space. With the pattern design of circular clock and "2018", and the creatively designed dense scatter diffusion algorithm to find the shortest possible time to calculate the path and avoid collision , the fast anti-collision switch of the patterns in the air can be achieved. Finally, on the basis of the performance indexes of UAV cluster, we planned and designed an operational air show program and made a high degree of simulation in Matlab to accurately calculate the number of UAVs required for the performance, the area of the air performance, The UAV launching path, the time interval between the launches, the initial position in the air, the pattern change and movement tracks, as well as the anti-collision security strategy, combined with the UAV rental price to give the basic budget, and combined with the layout of urban space to give recommendations of the location to put on the show..

***Keywords*:** UAV Aerial Show, Plane Motion Trajectory, 3D Motion Trajectory, Skeleton Model, Skin Attachment Model, Dense Scatter Diffusion Algorithm, Anti-collision pattern switch

Contents

Memo

To: Mr. Mayor

From: 2017 HIMCM Team #7359

Date: 2017/11/18

Topic: Report on possibility of drone light performance in New Year.

Dear Mr. Mayor:

You must have heard or seen the astonishing drone light show on the sky of Florida. In order to add unique festive atmosphere to the upcoming New Year of 2018, we suggest a drone light show as the best option. Please have a look at our report on the plan and possibility of such a light show below:

I. Plan and Design

Our general plan is to use a total of 420 drones arranged in a 20*22 order initially. They will take of in 1 minutes and form a vivacious and dynamic Ferris Wheel in the sky. With specific movement of the fleet of drones, the Ferris Wheel figure will start to spin. Three minutes later, a ferocious dragon will appear in the sky and perform a dance in front of the people. Finally, with a 5-second-count-down clock, a figure “2018” will appear in the sky to celebrate the New Year and put an end to the 10-minute drone light show. Below are the design for figures used in the light show:

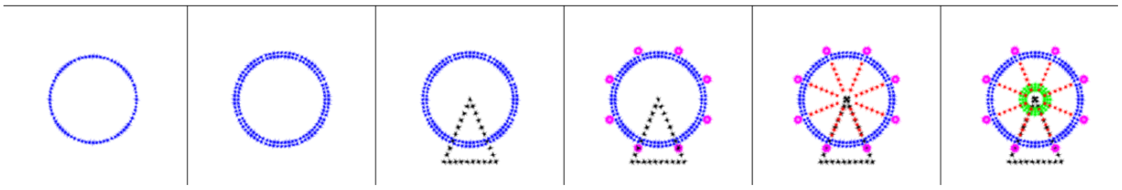


Fig1: Design of taking off and formation of Ferris Wheel

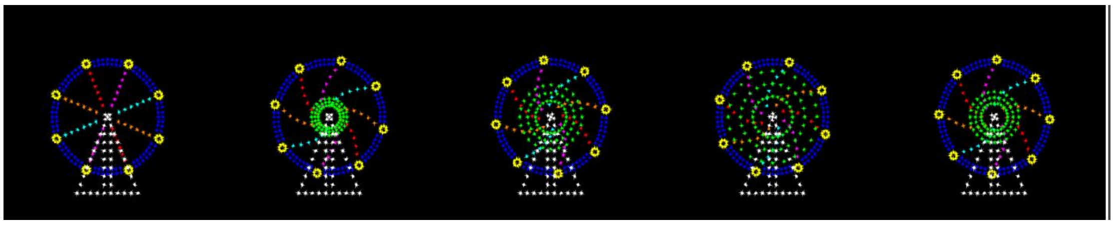


Fig2: Design of Rotation of the Ferris Wheel

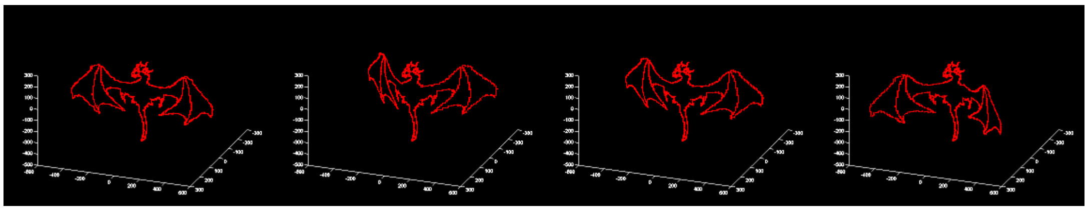


Fig3: Design of Dance of the Dragon

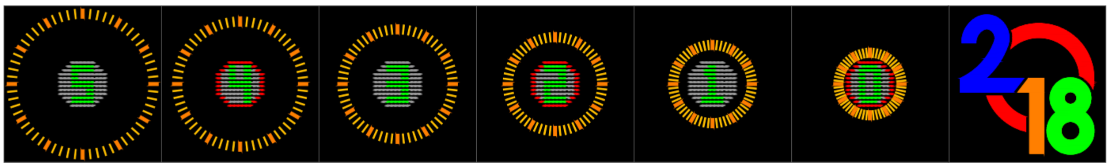


Fig4: Design of the Count-Down Clock

Mr. Mayor,We hope that our proposal has intrigued you. You might ask why use drones to do a light show and our answer is:

II. Reasons for Using Drones

First of all, a drone light show is a perfect combination of art and technology. Drones can control their trajectory precisely, and they are capable of showing various colors to create vivid and even moving figures. This will be a novel visual experience for the people and will thus bring a spirit of modern time to the city.

Also, a drone light show is environmentally friendly. Fireworks might be as well beautiful, but they produce huge amount of harmful gases such as Sulfur Dioxide, Nitrogen Monoxide, and Nitrogen Dioxide. There is no such problem with a drone light show and drones also can be recycled after performance.

Moreover, our team has already developed an advanced system to control the drones' flight. All you have to do is to provide an image, then we can finish off by designing, planning of trajectory, and determining number of drones.

Last but not least, we have also considered safety issues which you must have concerned with. We calculated taking of time for every drone according to its position in the sky and prevented collision issues during transition between two figure by designing a dense scatter diffusion algorithm. The possibility and safety of our plan is confirmed by simulations on the computer.

III. Plan for the Site for Performance

We will be using *Intel Shooting Stars* Drones in this performance. The size of each drone is 384*384*93 mm. 420 drones will be used in the performance and 10 for backup, so the total number of drones used will be 430. They will take off in a formation of 20*22. Assume that each drone takes 1m^2 of space, the area of the site for taking off should be at least around 430 m^2. We suggest the site to be located at the top of Times Square or a carrier boat on Hudson River in New York. We also have few suggestions for site for the audience:

1. Previous audience sites of firework shows: e.g. Times Square or Prospect Park, etc.
2. Open space near the Hudson river so figures in the sky can reflect: e.g. South Street Seaport, Williamsburg, Green point, and Dumbo, etc.

IV. Cost and Budget

We suggest to rent drones instead of purchasing them. For renting, every drone costs $30, and the total fee will be $12900. On the other hand, a similar Fireworks show will cost about $28500. As a result, a drone light show is also economically friendly.

We sincerely wish that the drone light show we designed could bring joy to the New Year and we hope to earn your support, thank you!

2017 HIMCM Team #7359

I. Introduction and Problem Restatement

1.1 Introduction

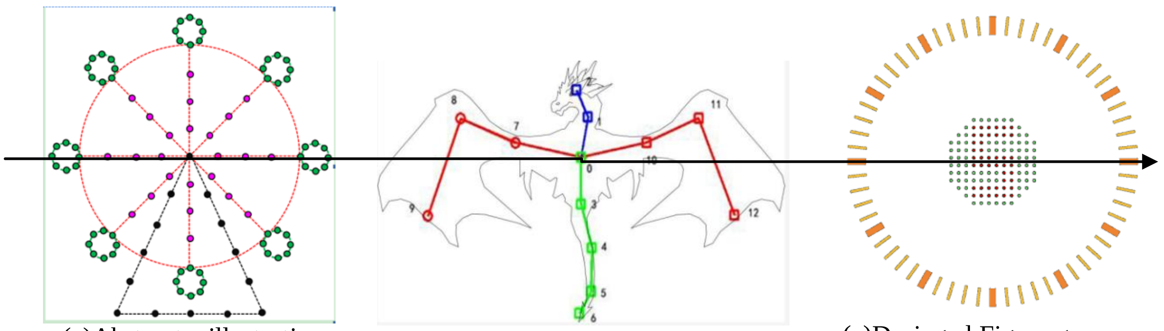
As Intel said, UAV sky light show[1] fully released the imagination of human-beings, bringing the infinite extension of the arts. The system's precise control of the graphics and artistic visual effects have given people a lot of surprises, and have been more and more used in urban air show. Our mission is to study the formation and movement of different patterns in the UAV sky show, whether it be rigid[2] or non-rigid[3] patterns, planar or three-dimensional motions. The design mathematically depict the flight path of the UAV to make the formation pattern lifelike in the air ,free to switch, and able to avoid collision in the process.

1.2 Problem Restatement

In late 2016, the “shooting star” project which took Intel and Disney two years to develop collaboratively, eventually took place. One programmer controlled 500 simultaneous UAVs through a laptop and performed a sky-light show over Florida. The visual effects stunned the audience. In order to increase the holiday atmosphere, in the upcoming 2018 New Year, the mayor considered using a UAV cluster to perform a beautiful air dance show and asked us to conduct an in-depth investigation and give an implementation plan. The goal is to create three possible patterns by using the UAV fleet. In addition to the designated Sky Ferris wheel and the dragon dance show, the third pattern is free to play. We need to determine the number of UAVs, the initial position of each UAV and UAV group in the sky, the flight path, The time and space required to complete the performance, while also taking full consideration of the budget, the launching area required and the related safety issues.

1.3 Problem Analysis

1.3.1 Fixed point analysis



(a)Abstract illustration of a Ferris wheel

(b)Design of Western dragon l

(c)Desinged Figure group （Countdown Happy New Year）

Fig 1. The preliminary design renderings of three types of patterns

A fixed point needs to be determined whether it is Aerial Ferris wheel, Western Dragon or self-designed pattern. At any time the coordinate position of the fixed point will not change .It is used to locate the static position and dynamic trajectory of the UAV groups. Figure 1 shows the preliminary design of three types of pattern. It is easy to see that for the Ferris wheel and New Year's countdown clock pattern, the fixed point can be set as the center of the circle, and the drone group refers to the center coordinates of the circle, determines its own position in the coordinate system according to its distance, and performs regular movements around the circle center in flight. According to the distance and direction to determine their own position in the coordinate system, as well as the target position when movement happens, to obtain the trajectory. For the western dragon, the position of the fixed point is rather complicated. Firstly, we select a set of key points according to the dance actions to be performed to form the motion skeleton, as shown in Figure 1 (b),the"0-13" mark points. Then determine a fixed point in this group of key points. In order to maintain the harmony and beauty of the action, we usually select the center of the dragon's body as the fixed point.

1.3.2 Flight path analysis

The flight path problem is closely related to the type of object and the mode of movement. From the design drawings, it can be seen that the motion of the Ferris wheel belongs to the rigid body motion of regular objects [2], and the motion of the dragon dance belongs to the non-rigid body motion of irregular objects [3]. For the Ferris wheel, the highest point on the triangle supporting stand, that is, the center of the concentric circle of the Ferris wheel body, is the fixed point of the rigid body. The wheel carousel, the wheel body frame, the points on the cabin are all making circular motion around it. The relative position of each point during the movement and remains the same. Therefore, the flight trajectory problem of UAV is transformed into the trajectory equation problem of three kinds of points making circular motion around the fixed point. In comparison, the dragon's air dance is more complex. The most significant difference is that during the dancing process, the relative position and shape of each point will change, with high degree of non-linear features. In order to solve this problem, we first selected a series of key points from the dragon body and their connections as skeleton of the dragon, and then use the link motion model[4] to model the movement of the dragon skeleton[5] to obtain the trajectory of the dragon skeleton, and finally determine the position of each. Skin attachment point of the skeleton[6](The drone that moves in synchrony with a certain skeleton is called the skin attachment point of the skeleton) and let them move in synchronization with the attached skeleton of the dragon to perform the dragon's flying show in the sky.

1.3.3 Three-dimensional motion trajectory analysis

Three-dimensional movement will make the dynamic effect of the air show more stunning. But the difficulty of the dynamic effect of three-dimensional drone flights imulation lies in how to calculate the coordinate points after the three-dimensional transformation[7]. In this model, we select the center line of dragon body as the rotational axis, and the wings make three-dimensional rotation around the axis. The new coordinate after the rotation of each point on the wings is calculated by using the three-dimensional rotation transformation, in order to realize the dragon's three-dimensional performance in the sky.

1.3.4 Flight collision safety analysis

Whether it is the rotation of the Ferris wheel, the air dance of the dragon, or the image switching of the sky clock, we have to consider the issue of the safety distance between the UAVs to avoid the collision of the UAVs during the flight. We avoid the collision by pre-designing the UAV flight path and strengthening the minimum safety distance strategy. For the Ferris wheel, we avoid the collision by distributing the points on the triangular brackets, the points on the Ferris wheel and the cabin points at three different air altitudes. For the dragon dance show, we avoid it by strengthening the minimum safety distance strategy, that is, if the theoretical distance between two UAVs is less than the minimum safety distance in the motion model, the actual relative distances between the two UAVs will not change until their theoretical distance is greater than the minimum safe distance. For the collision problem of pattern switching, We have creatively designed a dense scatter diffusion algorithm that avoids aircraft collision problems during switching process of patterns.

1.3.5 Other issues involved in the air show program

In addition to the issues above, we also need to decide the size of the pattern, the density of pixels under the budget, and then calculate the required number of UAVs, and the total area required for aerial performances. Besides, we need to determine the location of the air show to achieve the best performance effect, suitability for the launching of UAVs and proper space for audiences.

II. Definitions, Justifications, Assumptions and Variables

2.1 Assumptions:

1. All drones are of the same model and good quality.
2. Wind speed's influence to the drones is ignored
3. Drones accelerate constantly from rest. When the drones reach a certain velocity, they decelerate constantly to rest.
4. Every drone can be considered as a single point during calculations.

2.2 Variables

2.2.1 Variables Related to Drones:

$\mathbf{a}$: acceleration

$\mathbf{v}$: velocity

$\mathbf{t}$: time of fight

$\mathbf{s}$: distance of flight

2.2.2 Variables Related to Performance

S: total area used by performance

T: total time of performance

N: total number of drones

2.2.3 Variables Related to Ferris Wheel

$\mathbf{n_1}$: number of axes on the wheel

$\mathbf{n_2}$: The number of drones distributed on every spoke

$\mathbf{n_3}$: Numbers of the cabin

$\mathbf{n_4}$: The number of drones distributed on the cabin

$\mathbf{n_5}$: Number of inner rim

$\mathbf{n_6}$: The number of drones distributed on Inner rim

$\mathbf{m_1}$: The number of drones distributed on the left and right side of the triangle stand

$\mathbf{m_2}$: The number of drones distributed on the horizontal side of the triangle stand

R: radius of the wheel

$\mathbf{r}$: radius of one cabin

L: 1/2 horizontal length of the triangle stand

H: The distance between the horizontal stand of the triangle stand and the lowest point of the outer rim

β : angle between the base of the Ferris Wheel and the ground

$\gamma = \eta$: angle between two cabins; angle between two axis

θ : angle of the rotation of the Ferris Wheel

2.2.4 Variables Related to Dragon

φ : Largest angle between the wings and the body axis

III. Image preprocessing

In this paper, the purpose of image preprocessing is to remove the noise and edge glitches in the original input image so as to obtain clear image edge information and better scatter sampling at the edge of the image[8], and as well determine the position and quantity of the UAVs. The parts that need image preprocessing are the dragon pattern and a self-designed "2018" word pattern. The process is divided into the following steps:

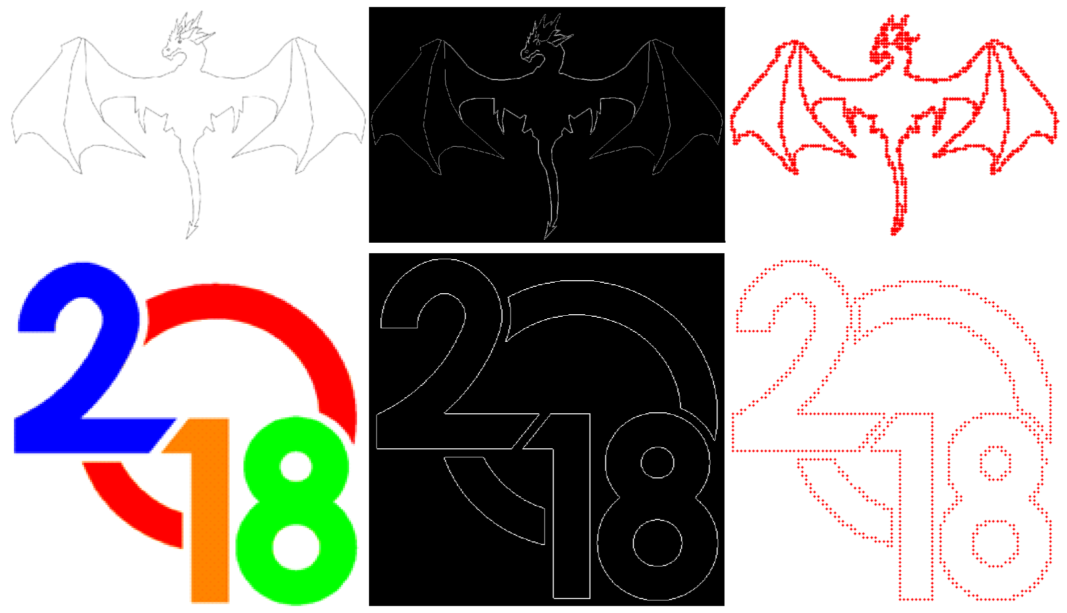
Step1: The input image is transformed into a grayscale image, and the binary image is further obtained by using the threshold transformation method[9], and then the binary image is inverted;

Step2: Use the linear spatial filter function[10][11] to filter the image to remove the noise point, while using the smoothing function[12] to remove the glitches at the edges of the image;

Step3: Gonzalez's boundary function [13][14] is used to extract the contour in a binary image;

Step4: The Gonzales bsubsamp function[15] is used to subsample the outline boundaries to get the scatter distribution of the image boundaries.

The result obtained by processing the output image through the above steps is shown in Fig. 2.



(a)The original input image (b) An image boundary map obtained by binarizing, inverting, and smoothing (c)Scatter plot obtained by subsampling the border

Fig 2. Effect of image preprocessing

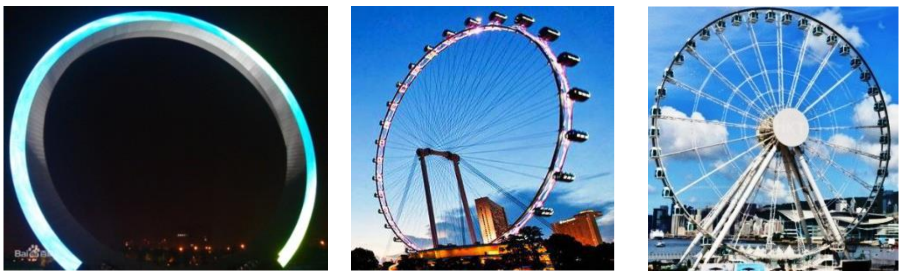
IV Mathematical Models

4.1 Aerial Ferris wheel

Aerial Ferris wheel is divided into three parts. First of all, we build a UAV path planning model, launch the UAV from the ground and generate the pre-designed pattern of the Ferris wheel in the air. Then, build the UAV's trajectory Model in the air according to the actual scenario of the Ferris wheel rotation, to achieve the effect of the Ferris wheel in the air. Finally, in order to enhance the effect of the Aerial show, we added flying postures of the UAVs to realize the dynamic display effect of the bottom supporting stand and the spoke of the Ferris wheel.

4.1.1 Graphic design

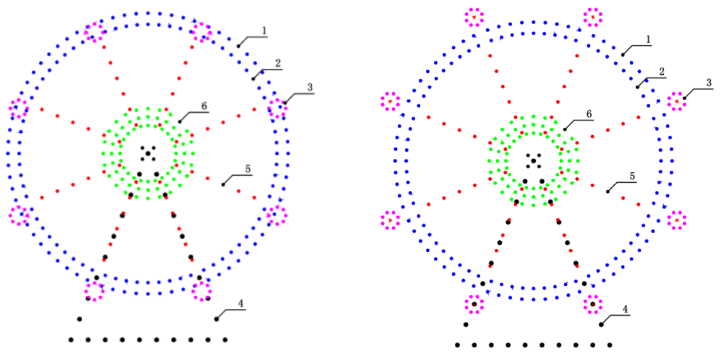
According to the different operating mechanisms, the Ferris wheel can be divided into three kinds: the non-spoke Ferris wheel (fig3(a)), the Gravity Ferris Wheel (fig3(b)) and the Observation Wheel (fig3(c)). Non-spoke Ferris wheel is like a ring, without any support in the middle, sightseeing cabin orbiting on a fixed track, this kind of Ferris wheel is relatively rare, it will not be considered in this design. Gravity Ferris wheel and Observation Ferris wheel structured similarly, the primary difference is between the cabin designs. Gravity Ferris wheel cabin hanging on the turntable to maintain the sliding axis of gravity; Cabins of observation Ferris wheel suspended outside the turntable, the angular speed of the cabin and turntable is the same, but in opposite direction, in order to maintain the level.



(a)Non-spoke Ferris wheel (b)Gravity Ferris Wheel (c) Observation Ferris Wheel

Fig 3. The actual Figure of three types of Ferris wheel

According to the actual Figure of the Ferris wheel, we designed the abstract structure diagram of the Gravity Ferris wheel and the Observation Ferris wheel, as shown in Figure 4. In the order of mark 1 to 6,they are the outer rim $\rightarrow$ the second rim $\rightarrow$cabin $\rightarrow$ lower triangular stand$\rightarrow$Spoke $\rightarrow$ multi-layer inner rim. It can be seen that the position of the center of the cabin determines the position of the other points on the cabin. Cockpit center of Gravity Ferris wheel is just below the rim, while cabin center of observation Ferris wheel is on the extension line of the spoke.



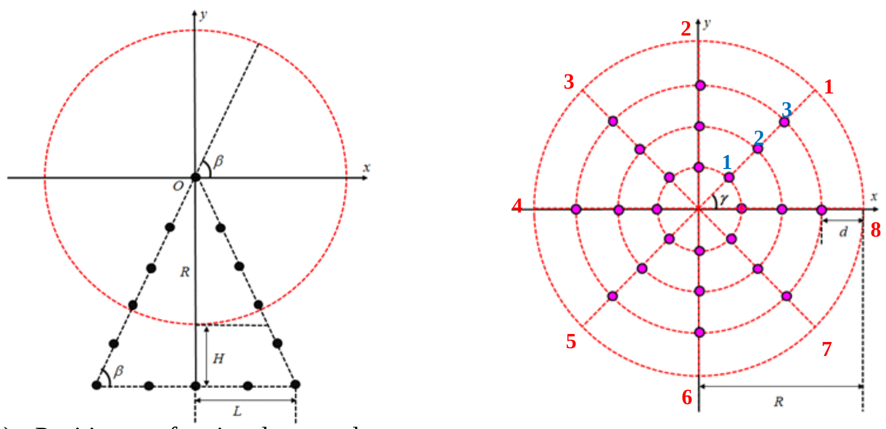
(a) Schematic diagram of gravity Ferris wheel (b) Schematic diagram of observation Ferris wheel

Fig 4. Schematic diagram of the abstract structure of two sorts Ferris wheel

4.1.2The calculation of various types of coordinates of Ferris wheel

As shown in Figure 5,the key points on the Ferris wheel are divided into three categories: points on the triangle stand, points on the turntable and the points on the cabins. The only difference between the gravity wheel and the observation wheel is the distribution of the cabin points. Therefore, this part includes the calculation of the coordinates of points on the triangle stand, the points on the turntable and the two types of cabin points.

The rectangular coordinate system is defined as: The center of the Ferris wheel is the origin, the horizontal right is the positive direction of X-axis, the vertical upward is the positive direction of Y-axis. The calculation of various key points is as follows.



(a) Positions of triangle stand pointsand related parameters

(b) Positions of rotary support shaft pointstand related parameters

Fig 5. Diagram of key points of triangle stand and turntable support shaft

1.Triangle stand points

The positions of the triangle stand in the coordinate system and related parameters are shown in Figure 5 (a).Then the linear equation of left stand can be expressed as:

$$y = x\tan\beta \qquad (1)$$

The right stand is symmetrical with the left bracket, whose line equation is:

$$y = -x\tan\beta \qquad (2)$$

The line equation for the bottom rim of the stand is:

$$y = -R - H, \quad -L \le x \le L \qquad (3)$$

Where $. L = (R+H)/\tan\beta$

Assuming that each of the left and right stand rims is evenly distributed with m_1 points (including the two ends of the stand), then coordinate of the left rim points are:

$$\begin{cases} x_i = \dfrac{(i-m_1)*L}{m_1-1} \\ y_i = \dfrac{(i-m_1)*(R+H)}{m_1-1} \end{cases} \quad i=1,\dots,m_1 \qquad (4)$$

In symmetry, coordinates of the right rim points are:

$$\begin{cases} x_i = \dfrac{(m_1-i)*L}{m_1-1} \\ y_i = \dfrac{(i-m_1)*(R+H)}{m_1-1} \end{cases} \quad i=1,\dots,m_1 \qquad (5)$$

Assuming that there are m_2 points on the bottom rim (including the two vertices), the coordinates of the bottom rim are:

$$\begin{cases} x_i = L - \dfrac{(m_2 - i)*2*L}{m_2 - 1} = \dfrac{(2*i - m_2 - 1)*L}{m_2 - 1} \\ y_i = -R - H \end{cases} \qquad i = 1, \ldots, m_1 \tag{6}$$

2.Coordinates of turntable support points

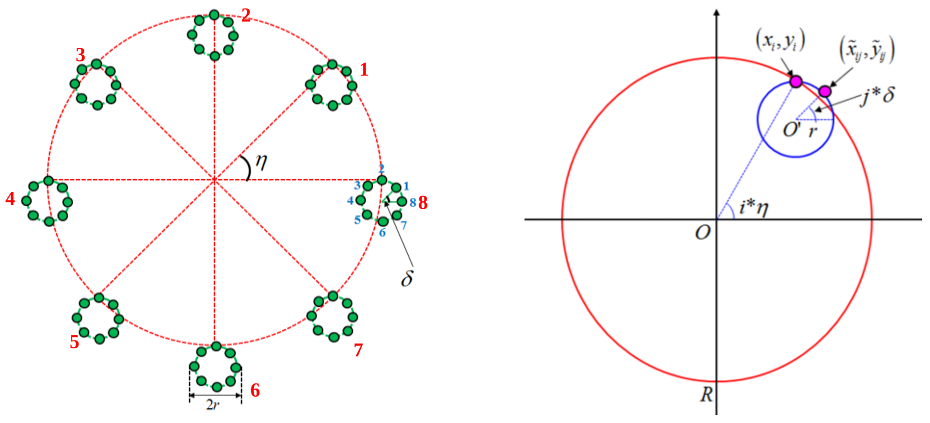
The position of the turntable support shaft in the coordinate system and related parameters are shown in Figure 5 (b). Assuming that the turntable evenly distributes n_1 support shafts, the angle between two adjacent support shafts is: $\gamma = 2\pi / n_1$

If each support shaft evenly distributed n_2 points, then the distance between two adjacent key points on the same bracket is: $d = R / n_2$

For convenience of calculation, mark the support shaft in the positive x-axis direction is the n_1 support shaft, and start marking the remaining support shafts in the order of anticlockwise rotation as 1st, ..., $n_1 - 1$, support shaft(as shown in Fig 5(b) red number logo).Note that the point closest to the origin on each support shaft is marked as the first point, and the remaining points on the support shaft are marked as 2nd, ..., n_2 (as shown in Fig 5(b) Color number logo).The Number **i** support shaft of the j-point coordinates can be expressed as:

$$\begin{cases} x_{ij} = (j*d)*\cos(i*\gamma) \\ y_{ij} = (j*d)*\sin(i*\gamma) \end{cases} \qquad i = 1, \ldots, n_1,\ j = 1, \ldots, n_2 \tag{7}$$

3.Gravity Ferris wheel cabin point coordinates：



(a) Cabin key point position diagram (b) Cabin key points related parameters diagram

Fig 6. Gravity Ferris wheel cabin key point location and related parameters diagram

Key points of Gravity Ferris wheel cabin and related parameters are shown in Fig 6. If n3 cabins are evenly distributed in the Ferris wheel, the angle between the adjacent cabin stands is expressed as: $\eta = 2\pi / n_3$

In general terms, **γ = η** ,it is assumed during the calculation that there will be a cabin at the end of each support shaft.

If n_4 points are evenly distributed on each cabin, the angle between the radii of two adjacent points is: $\delta = 2\pi / n_4$

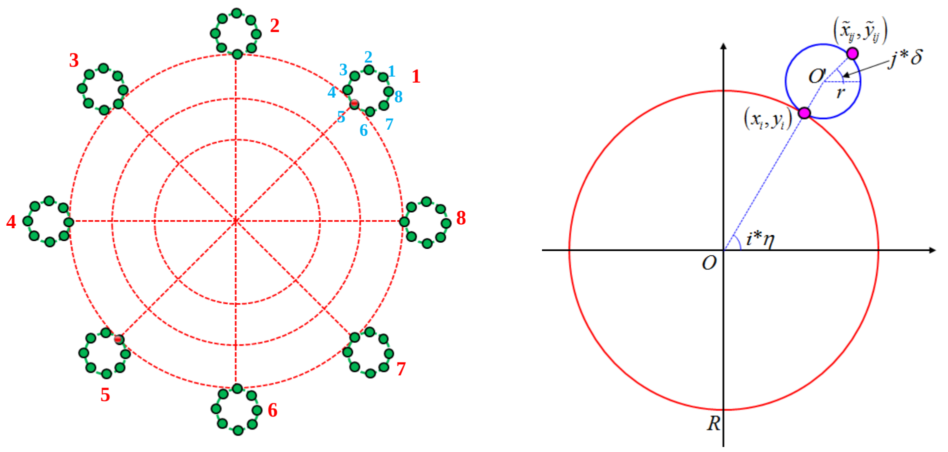
For convenience of calculation, mark the cabin in the positive x-axis direction of the apex as the first cabin, and the remaining cabins are numbered 1, ..., and n_3-1cabin in the order of anticlockwise (as shown in Fig 5 (a)). Mark the rightmost point of the cabin as the b point, and in the order of counterclockwise, mark the other points as 1th,..., n_4-1. We use (x_i, y_i) to represent the coordinates of the Number **i** cabin's apex, which represents the coordinates of the Number **j** point on the Number **i** cabin. The coordinates of the apex of the cabin are:

$$\begin{cases} x_i = R\cos(i*\eta) \\ y_i = R\sin(i*\eta) \end{cases}, \quad i = 1,\cdots,n_3 \tag{8}$$

Since the distance between center of the cabin and the apex is r, the center coordinate O' of the cabin Number **i** can be expressed as $(x_i, y_i - r)$ so that the coordinate of the **j** point of the Number **i** cabin is:

$$\begin{cases} x_{ij} = x_i + r\cos(j*\delta) \\ y_{ij} = (y_i - r) + r\sin(j*\delta) \end{cases} \quad i = 1,\cdots,n_3,\ j = 1,\cdots,n_4 \tag{9}$$

4.Coordinates of Observation Ferris wheel cabins



(a) Schematic diagram of coordinates of cabin key points

(b)Schematic diagram of Cabin key points related parameters

Fig 7. Schematic diagram of coordinates of Observation Ferris wheel cabin's key points and related parameters

As shown in Fig 7, the center O' of the cabin is on the extension line of the radius of the outermost circle with $O'O = R + r$. In the same situation with the calculation of the coordinates of the gravity wheel cabin, the coordinates of the center O' of the Number **i** cabin can be obtained as:

$$\begin{cases} x = (R+r)\cos(i*\eta) \\ y = (R+r)\sin(i*\eta) \end{cases} \tag{10}$$

Therefore, the coordinates of the **j** point in the Number **i** cabin can be expressed as:

$$\begin{cases} x_{ij} = (R+r)\cos(i*\eta) + r\cos(j*\delta) \\ y_{ij} = (R+r)\sin(i*\eta) + r\sin(j*\delta) \end{cases} \quad i = 1,\cdots,n_3,\ j = 1,\cdots,n_4 \tag{11}$$

4.1.3Model of UAV Path Planning

Path planning model calculates the takeoff time based on UAV's target coordinate in the air and time period needed to arrive at the coordinate. Take launching of the sky Ferris wheel as an example, as shown in Fig8Assuming that the UAV will be launched in six batches, in following order : the outer rim $\rightarrow$ the second rim $\rightarrow$ the cabin $\rightarrow$ the triangle stand $\rightarrow$ the spokes $\rightarrow$ the multi-layer inner rim

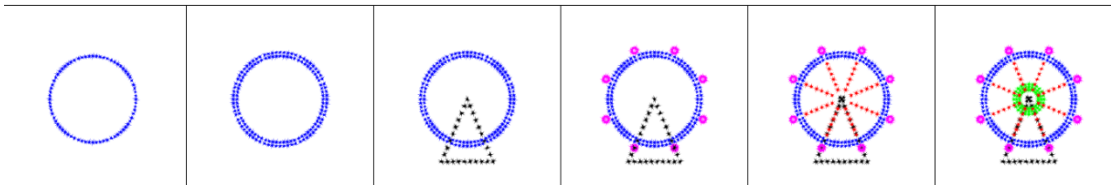


Fig8. Schematic diagram of the order of launching of the Ferris wheel

Assuming that the Number **i** batch of UAVs arrives in the air at time T_i, it can be expressed as : $T_i = (i-1)t + \frac{2R+H+h}{v}$ (12)

Then the takeoff time T_{ij} of Number **i** batch of UAV from the ground can be expressed as:

$$T_{ij} = T_i - \frac{D_{ij}}{v} \qquad (13)$$

In this case, D_{ij} is the distance from the ground of UAV **j** in batch **i.v** is the maximum rising speed of the UAV, **h** is the height of the bottom rim of the triangle stand above the ground.

4.1.4 Rotation model of Ferris wheel in the air

The problem of the rotation of the Ferris wheel in the air is actually to solve the motion trajectory problem of the three types of points such as the triangular stand points, the spoke points and the cabin points on the Ferris wheel. When the Ferris wheel is rotating, the points on the triangle stand are static, so it has no trajectory equation. The spoke points and cabin points rotate with the rotation of the Ferris wheel, which is a very typical rigid body motion in plane[16], Their trajectory is divided into the following three parts:

1.Trajectory of the spoke points

(x_{ij}, y_{ij}) represents the coordinates of the **j** point on the Number **i** spoke of the Ferris wheel, where the meanings of **i** and **j** are shown in Fig 5(b)(see 3.1.2 Calculation of the coordinates of various types of points on the Ferris wheel). Its trajectory equation is

$$\begin{cases} x_{ij} = (j*d)\cos(i*\gamma+\theta) \\ y_{ij} = (j*d)\sin(i*\gamma+\theta) \end{cases} \qquad (14)$$

That is, the spoke point (x_{ij}, y_{ij}) makes a circular motion with O as the center and with radius of **j*d**.

2.Trajectory of Gravity style cabin points

Cabin point to follow its center of the cabin O' to make a rotational movement, so the key is the trajectory of the center of the cabin O'. Mark (x_i, y_i) for the Number **i** cabin center coordinates, then the trajectory of the parameter equation is:

$$\begin{cases} x_i = R\cos(i*\eta+\theta) \\ y_i = R\sin(i*\eta+\theta) - r \end{cases} \quad (15)$$

Mark $(\tilde{x}_{ij}, \tilde{y}_{ij})$ as the coordinates of the **j** point on the Number **i** cabin of the Ferris wheel, where the meaning of **i** and **j** is shown in Fig 6(a), then the relation with (x_i, y_i) can be expressed by the following formula

$$\begin{cases} \tilde{x}_{ij} = x_i + r\cos(j*\delta) \\ \tilde{y}_{ij} = y_i + r\sin(j*\delta) \end{cases} \quad (16)$$

Take (14) into equation (13), we can get trajectory equation of $(\tilde{x}_{ij}, \tilde{y}_{ij})$

$$\begin{cases} \tilde{x}_{ij} = r\cos(j*\delta) + R\cos(i*\eta+\theta) \\ \tilde{y}_{ij} = r(\sin(j*\delta)\text{-}1) + R\sin(i*\eta+\theta) \end{cases} \quad (17)$$

That is, the trajectory of the cabin point $(\tilde{x}_{ij}, \tilde{y}_{ij})$ is a circle with the center of the circle $(r\cos(j*\delta),\ r(\sin(j*\delta)\text{-}1))$ and radius **R.**

3.View-style cabin point trajectory

Similar to the gravity type cabin point, the cabin point follows the center of the cabin O' for rotational movement, with the difference being that the position of the circle center O' is different. In the Ferris wheel, the parameter equation of the Number **i** cabin trajectory is

$$\begin{cases} x_i = (R+r)\cos(i*\eta+\theta) \\ y_i = (R+r)\sin(i*\eta+\theta) \end{cases} \quad (18)$$

Similarly, the coordinate of the **j** point on the Number **i** cabin is denoted by $(\tilde{x}_{ij}, \tilde{y}_{ij})$, then its relation with the center of circle O' is shown in (18). Therefore, the equation of motion trajectory is

$$\begin{cases} \tilde{x}_{ij} = r\cos(j*\delta) + (R+r)\cos(i*\eta+\theta) \\ \tilde{y}_{ij} = r\sin(j*\delta) + (R+r)\sin(i*\eta+\theta) \end{cases} \quad (19)$$

Therefore point $(\tilde{x}_{ij}, \tilde{y}_{ij})$ makes circular motion with $(r\cos(j*\delta),\ r\sin(j*\delta))$ as the center, $(R+r)$ as radius.

4.1.5 Models with Further Development

In order to make the Sky wheel more vivid and to enhance the dynamic display effect, we developed the following three kinds of enhanced actions in addition to simulating the rotation of the Ferris wheel: Programed the UAVs to make clockwise motion along the triangular stand, highlighting the dynamic contour of the triangle stand; Have the UAVs on the inner rim repeatedly move away from the center and move back closet to display the dynamic divergent effect of the inner rim. The spokes are curved during the rotation to show the dynamic rotating effect of the spokes. Enhanced actions involves coordinate calculation and trajectory calculation as follows.

1.coordinates of arcuate spoke and its motion track

As shown in the structural diagram 10, the **i** spoke of the Ferris wheel is changed from a straight line to a 60degree arcuate line with a radius of R. The center of the arc O_i' (x_i', y_i') is obtained by rotating the end point (x_i, y_i) of the i spoke counterclockwise by 60 degrees around the origin. The rotation track in Matlab can be rendered as Fig.9.

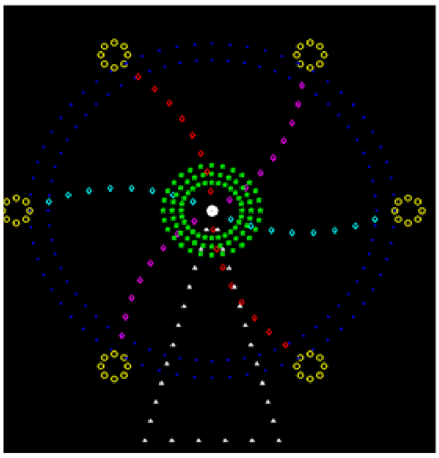


Fig 9. Arcuate spoke Matlab simulation

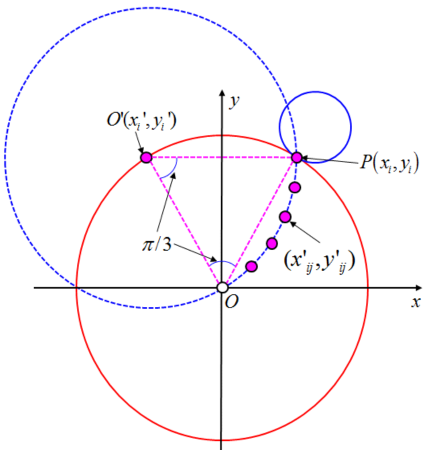


Fig 10. Ferris wheel arcuate spoke coordinate diagram

As shown in Fig 10, the relationship between $O_i'(x_i', y_i')$ and (x_i, y_i) is

in this case, .
$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = B \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{20}$$

$$B = \begin{bmatrix} \cos(\pi/3) & \sin(\pi/3) \\ -\sin(\pi/3) & \cos(\pi/3) \end{bmatrix}$$

The point on the arcuate spoke is the path of (x_i, y_i) rotated clockwise by $0 \sim \pi/3$ around the axis O_i'. If each spoke evenly distributed n_2 points (excluding the center of the circle), the coordinates of the **j** point (x'_{ij}, y'_{ij}) is the track of (x_i, y_i) rotated around O_i' clockwise by. $(j-1)\pi/(3n_2)$ Therefore, the relationship between them can be described as follows:

$$\begin{pmatrix} x_{ij}' \\ y_{ij}' \end{pmatrix} - \begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = C\left(\begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_i' \\ y_i' \end{pmatrix}\right) \tag{21}$$

in this case,.

$$C = \begin{bmatrix} \cos((j-1)\pi/(3n_2)) & -\sin((j-1)\pi/(3n_2)) \\ \sin((j-1)\pi/(3n_2)) & \cos((j-1)\pi/(3n_2)) \end{bmatrix}$$

take （ 20 ） into (21), $\begin{pmatrix} x_{ij}' \\ y_{ij}' \end{pmatrix} = (B + C - CB)\begin{pmatrix} x_i \\ y_i \end{pmatrix}$ (22)

Therefore, the motion track of (x'_{ij}, y'_{ij}) is a circle with O as its center and $(|B + C - C * B| * R)$ as it radius.

2.Dynamic rim coordinates and its motion track

There are m layers of inner rims evenly placed on the Ferris wheel and n_5 point is evenly distributed on each inner rim; the innermost rim has a minimum radius of r_1; the outermost rim has a maximum radius of r_2; the minimum distance between rims is s_1, and the maximum distance is s_2.When the Ferris wheel rotates a circle, the moving inner rims complete T_1 times of diffusion and contraction. The diffusion track can be shown as Fig 11.

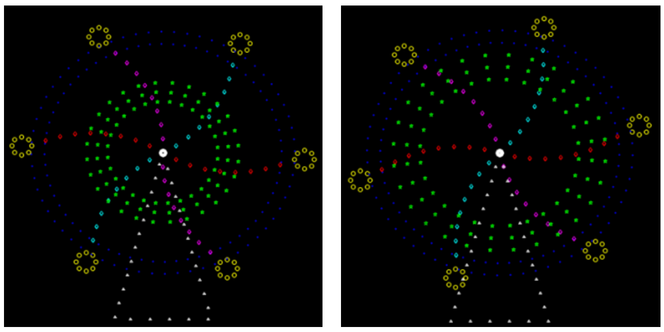


Fig 11. Simulation of the diffusion effect of the inner rim

If the shortest distance from the **i** inner rim to the center of the circle O is defined as d_i, and the longest distance is defined as $d_i^{'}$,then

$$\begin{cases} d_i = r_1 + (i-1) * s_1 \\ d_i^{'} = r_2 - (m-i) * s_2 \end{cases} \qquad (22)$$

If the Ferris wheel is rotated by q degree, the diffusion and contraction frequency of the inner rim is: $T_1 \dfrac{\theta}{2\pi}$ and the fractional part of $T_1 \dfrac{\theta}{2\pi}$ is e, namely

$$e = T_1 \frac{\theta}{2\pi} - \left\lfloor T_1 \frac{\theta}{2\pi} \right\rfloor \qquad (23)$$

And the integer part of $T_1 \dfrac{\theta}{2\pi}$ is $\left\lfloor T_1 \dfrac{\theta}{2\pi} \right\rfloor$

then the **i** spoke's distance from the center of circle is

(($$D_i = d_i^{'} - 2(d_i^{'} - d_i)\left|e - 1/2\right| \qquad (24)$$

In this case, the motion track of the coordinate (x_{ij}, y_{ij}) of the **j** point of the **i** spoke can be described by the following parameter equation

$$\begin{cases} x_{ij} = D_i \cos(j * \gamma + \theta) \\ y_{ij} = D_i \sin(j * \gamma + \theta) \end{cases} \qquad (25)$$

Namely, it is making a circular movement with O as its center of circle and D_i as its radius(with the rotation angle of θ).

3.Coordinates of the triangular support structure and their motion track

The initial coordinates of the key points on the triangular support structure can be found in Section **4.1.2** of this paper. That is, the initial coordinates of the key points on the left arm, the right arm and the base are respectively described by formulas (4), (5)

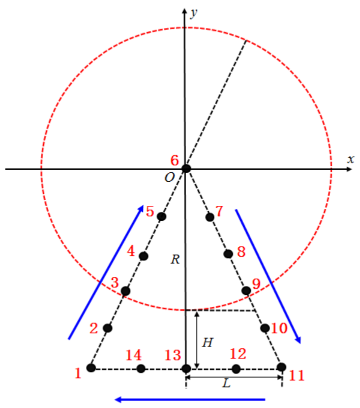


Fig12. Bottom triangular structure clockwise movement diagram

and (6). In this part of the paper, we discuss the coordinates of the key points of the support structure when the wheel are doing the clockwise movements around the triangular support structure (as shown in Figure 12).

We mark the leftmost key point on the base at the initial moment as point 1, and the remaining key points in the clockwise direction are numbered 2,..., and $(2m_1 + m_2 - 3)$, as shown in Fig 14. Assuming the Ferris wheel rotates one circle and each drone can complete a movement of T_2 units of distance (defining the distance between two adjacent key points as 1 unit of distance), when the Ferris wheel rotates θ degrees, each drone moves a distance of $\mathrm{T}_2\theta/(2\pi)$. If we define the distance of movement of the **i** key point from its initial position as $D_i^{'}$, then:

$$D_i^{'} = \mathrm{mod}\left(\frac{\mathrm{T}_2\theta}{2\pi} + i - 1, 2m_1 + m_2 - 3\right) \qquad (26)$$

In this case, mod is the remainder function. Therefore, the coordinates of the**i** key point are

$$(x_i, y_i) = \begin{cases} \left(\dfrac{(D_i^{'} - m_1)*L}{m_1 - 1}, \dfrac{(D_i^{'} - m_1)*(R+H)}{m_1 - 1}\right) & (0 \le D_i^{'} < m_1 - 1) \\ \left(\dfrac{(D_i^{'} - m_1 + 1)*L}{m_1 - 1}, \dfrac{(D_i^{'} - m_1 + 1)*(R+H)}{m_1 - 1}\right) & (m_1 - 1 \le D_i^{'} < 2m_1 - 2) \\ \left(\dfrac{(D_i^{'} - 2m_1 + 2)*L}{m_2 - 1}, \dfrac{(D_i^{'} - 2m_1 + 2)*(R+H)}{m_2 - 1}\right) & (2m_1 - 2 \le D_i^{'} < 2m_1 + m_2 - 3) \end{cases} \qquad (27)$$

Since $D_i^{'}$ is a function of θ, the above formula is the parameter equation of the track of the **i** key point (θ is a parameter). On the basis of the original rotation model, after an enhancement, the dynamic display effect of the Ferris wheel is simulated in MATLAB as shown in the figure 13:

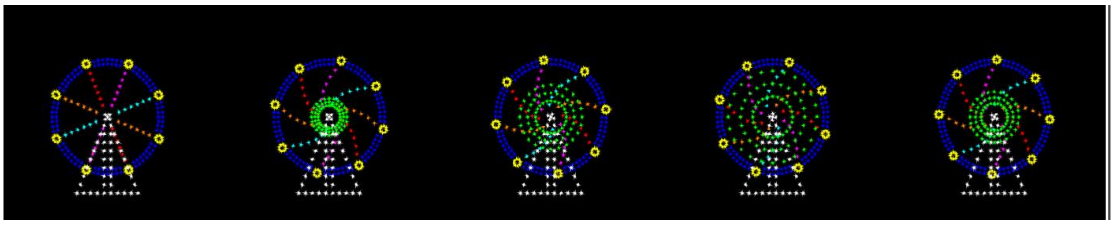


Fig13. Ferris wheel enhanced dynamic display simulation renderings

4.1.6 Ferris wheel modeling summary and analysis

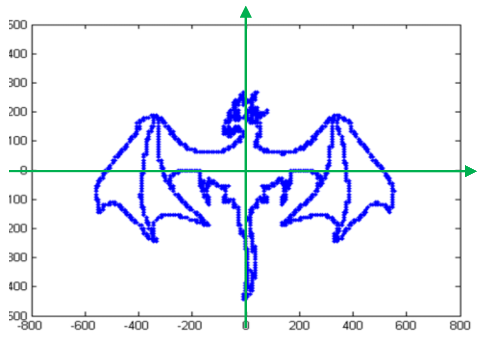
In the process of modeling the Ferris wheel, we can calculate the coordinates and motion tracks of all points, based on the structure of the wheels, in the gravity and sightseeing Ferris wheels so as to realize the turning effect of the Ferris wheel in the air. Meanwhile, it also innovatively designed the effect of the arcuate support arm, the contracting and extension movement of the inner rims and the triangular support frame moving clockwise to enhance the air show. The parameterized design concept of the model can be quickly completed and modified based on the user's needs, so it is very practical and applicable. Moreover, the established path-planning model can be applied not only to the launch of the drones but also to the safe and quick Figure changing when the drones are in the air.

4.2 Dragon air dance show

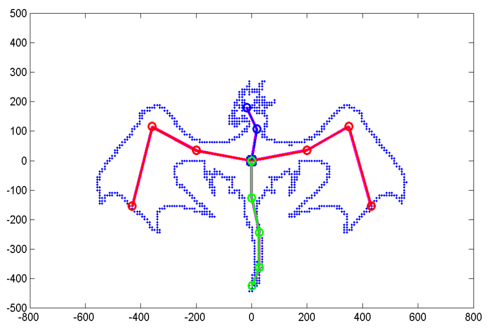
We designed the motion model of the dragon in both 2-dimension and 3-dimension model. The 2-dimension model can be divided into skeleton model and the out line model. Please refer to Chapter III for the graphic design of the dragon.

4.2.1 The Graphic Design of the Dragon and the Establishment of the 2-Dimensional Coordinate System

The drawing of the dragon was done by ourselves , After the graphic processing, the outline of the dragon as shown in Fig14 (c). In order to coordinate the dragon’s movement in the air properly, we took the graphic center of the dragon’s body as the fixed point. And we established the 2-dimensional coordinate system with this fixed point as the origin and each point in the outline is calculated and shown as in Fig 14(a).



(a) dragon's outline coordinates



(b) The key point of the dragon and its skeleton

Fig 14. Outline points and skeleton of the dragon

4.2.2 2-Dimensional Movement of the Dragon

Compared with the Ferris wheel, the dragon is an odd object. It has more complicated movement pattern, which is a typical non-rigid motion[2], due to its multiple joints. In order to solve this problem, we simplified the linkage of the dragon's bones and joints to a multi-link system. Therefore, the key issue is to selection of the key points in the skeleton and the establishment of the multi-link system.

4.2.2.1Dragon skeleton movement model

Step1：According to the dragon's coordinate chart, 13 key points are selected according to the head movement, the trunk movement, the wing movement and the tail movement, and the dragon skeleton coordinate chart is shown in Fig10(b). We marked the key points and recorded the key points involved in the movement of various parts of the dragon body, as shown in Fig 15and Table 1

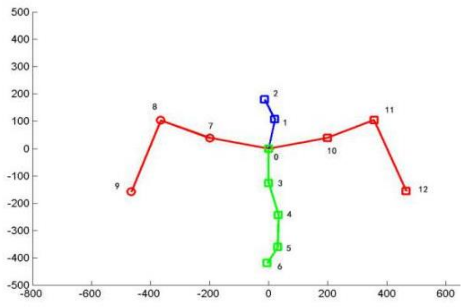


Fig 15. keel key point label

table 1. The sequence of key points involved in the movement of each component

Sports branch	Key point sequence
Head movement	0-1-2
Torso exercise	0-1,0-3
Left wing movement	0-7-8-9
Right wing movement	0-10-11-12
Tail movement	0-3-4-5-6

Step2: For each motion part in the dragon skeleton, its motion track should be rendered. Let's take the right wing for example. The detailed process is as follows:

As shown in figure 16, dragon's right wing has 4 key points A_0, A_1, A_2, A_3 which forms a three-joint link system, and takes A_0 as the fixed point. Firstly, A_1, A_2, A_3 makes a rotation around A_0 and this is defined as the movement Z_1; then, A_2, A_3 makes a rotation around A_1 which defined as the movement Z_2; finally, A_3 makes a rotation around A_2 which is defined as the movement Z_3; thus a new coordinates of A_1, A_2, A_3 are marked as A_1', A_2', A_3'. The mathematical relationship between them can be described as

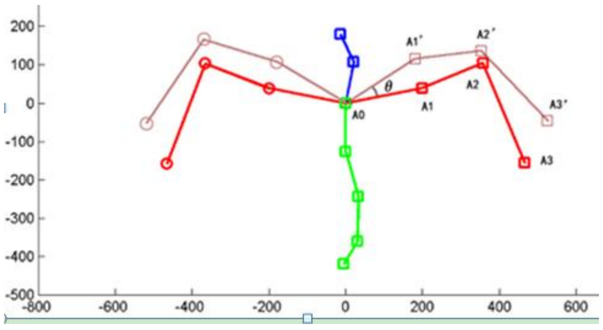


Fig 16.the rotative diagram of right-wing skeleton

$$\begin{cases} A_1' - A_0 = Z_1(A_1 - A_0) \\ A_2' - A_1' = Z_2 Z_1 (A_2 - A_1) \\ A_3' - A_2' = Z_3 Z_2 Z_1 (A_3 - A_2) \end{cases} \quad (28)$$

In this case, $Z_i (i = 1, 2, 3)$ is the rotation variation matrix, namely

$$Z_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \quad (29)$$

In this case, θ_i is a counterclockwise rotation angle. The motion track of the key points can be derived by (28)

$$\begin{cases} A_1' = Z_1(A_1 - A_0) + A_0 \\ A_2' = Z_2 Z_1 (A_2 - A_1) + Z_1(A_1 - A_0) + A_0 \\ A_3' = Z_3 Z_2 Z_1 (A_3 - A_2) + Z_2 Z_1 (A_2 - A_1) + Z_1 (A_1 - A_0) + A_0 \end{cases} \quad (30)$$

Step3 According to the pattern of Step2, the motion track of the head, trunk and wings are deduced one by one, and the motion track of the synthesized skeleton is obtained, as shown in Fig17.

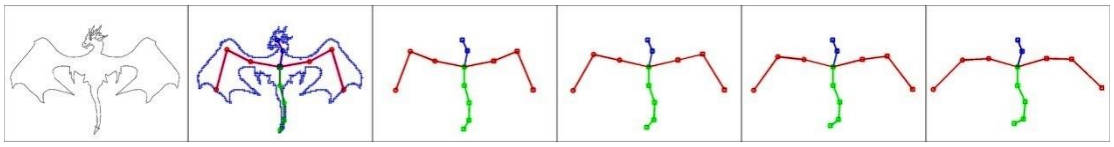


Fig 17. Dragon skeleton kinematic trajectory

4.2.2.2Model of the Dragon's Outline Attachment Points

We defined the drones which synchronize the movement of the skeleton as the outline attachment point of the skeleton, Since the movement of the skeleton and its outline attachment points are synchronized, setting attachment points for each skeleton will enable the movement of the outline. We also take the right wing as an example, the

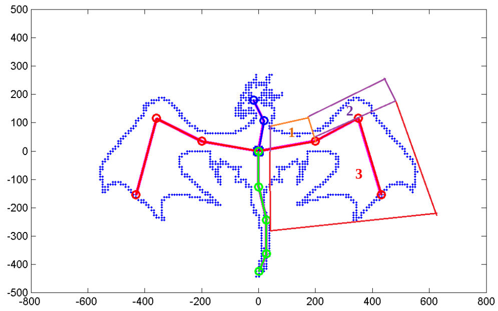


Fig 18. skin attachment point schematic

motion track of its outline can be shown.

According to the coverage of each skeleton, the right wing's our line attachment points are divided into three disjoint sets, as shown in Fig. 13.The sets of outline attachment points in the area 1, 2 and 3 are marked as S_1, S_2, S_3. The coordinates of the **j** attachment point (j = 1,2, ..., m$_i$, **i** = 1,2,3) in the set S_i are marked as S_1, S_2, S_3. The mathematical relationship between them is similar to the relationship between skeleton key point A_i and A_i', and it can be described by the following formula:

$$\tilde{x}_{ij} = \prod_{\tau=1}^{i} Z_\tau (x_{ij} - A_{i-1}) + \sum_{k=1}^{i-1} \prod_{\tau=1}^{k} Z_\tau (A_k - A_{k-1}) + x_0 \qquad (31)$$

4.2.3Three-Dimensional Motion Model of the Dragon

On the basis of 2-dimansionalmotion, we further extended the movement of the western dragon into three-dimensional space. Considering that the dragon image is relatively complicated, we split the dragon into body, left wing, right wing and tail. We define the plane where the dragon is located initially as the **yoz** plane, as shown in Figure 19. If we assume that the body of the dragon is limited the plane of **yoz**, the movement of its wings can be defined as rotational movements with **Z** as their axis. This 3-mimentinal movement can make this air show to be more vivid.

In the 3D motion model, we firstly need to convert the 2-dimensional coordinates of the outline points into 3-dimensional coordinates. Because the initial plane

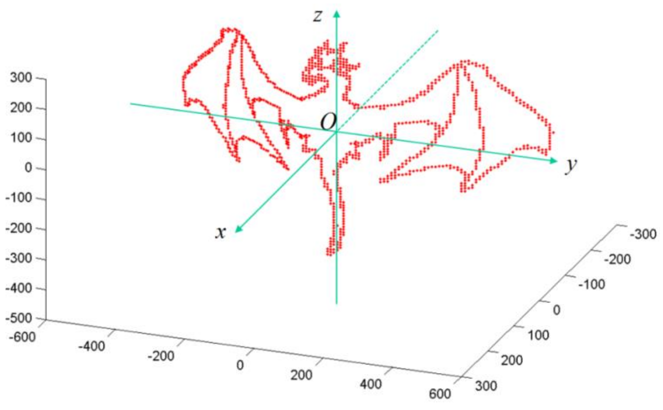


Fig 19dragon in three-dimensional space diagram

of the dragon is **yoz**, the 2D to 3D conversion can be achieved by the formula of:

$$\begin{gathered} R^2 \quad \rightarrow \quad R^3 \\ (x, y) \rightarrow (0, x, y) \end{gathered} \qquad (31)$$

Assuming that a random point $P(x, y, z)$ on the wings is rotated by θ degrees around z-axes and the coordinate is $\tilde{P}(\tilde{x}, \tilde{y}, \tilde{z})$, then the relationship between them can be expressed as

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (32)$$

According to (32), we can calculate the track of the dragon's wings during the movement. The simulation in MATLAB is shown in Fig 20:

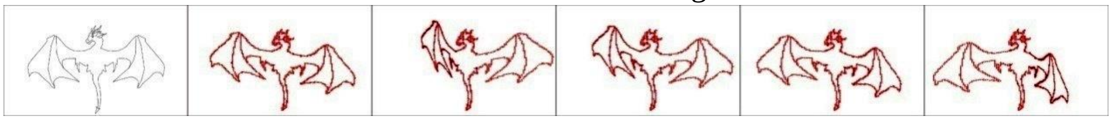


Fig20. Dragon in three-dimensional space motion trajectory dynamic diagram

4.2.4Summary and Analysis of Dragon Modeling

In the process of modeling the western dragon, we firstly get the sample result of the dragon's contour through image preprocessing, establish the dragons' scatter plot on the plane, and select the key point of the dragon skeleton and construct the dragon skeleton based on the multi-link rotating system Motion model; through the dragon skeleton to drive synchronous motion with the outline of the scattered points to complete the dragon's trajectory in the two-dimensional plane, and the application of three-dimensional rotation matrix, calculated on the dragon's wings scattered around the central axis of the dragon body rotation Transformation of the trajectory of the dragon to achieve the movement in three-dimensional space. The above model results are universal and can be conveniently applied to 2D and 3D movements of any non-rigid graphic.

In the modelling of the dragon, we firstly outline points by graphic processing. Then we established the coordinates graph of the outline points of the dragon and set the key points of the skeleton of the dragon, and the skeleton motion model was establish based on the multi-link system. The 2-dimentinoal motion was achieved by the synchronization of the dragon’s skeleton and its outline. The 3-dimentinal movement was achieved by applying the 3-dimensinal rotation matrix so as to calculate the motion track of points on the wings of the dragon moving around the body of the dragon as their axis. The above model is universally applicable, so it can be applied to any 2-dimensional or 3-dimensional movement of non-rigid graph.

4.3New Year Welcoming Sky Clock

In response to the requirement of the question, in order to increase the atomsphere of the following festival, we further designed a celestial clock, which performed New Year countdown in the sky as our self-designed dynamic show. And the process is shown in Fig.21

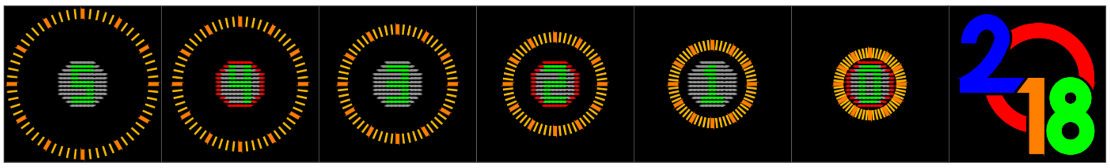


Fig21. The Renderings of the New Year Welcoming Sky Clock

4.3.1 Program design

1.The display of the countdown clock.

In the first six figures, the UAVs on the outer rim perform uniform rectilinear motion to the center of the circle, emphasizing the countdown effect to the audiences; the UAVs in the inner rim form a regular octagonal lattice and follow a clockwise route, while the lights on the UAVs are revealing the numbers

2.Shifting graphs to art characters.

We designed the “dense scatter diffusion algorithm”, to realize the moving locus of scatter from the clock (dense dot pattern) to the squiggle(sparse dot pattern) in a short time. Meanwhile, during the moving process, cross-collision situations are avoided, ensuring the safety of the UAVs;

3. The representation of the "2018"art characters.

The representation of "2018" achieves an artistical and dynamic effectvia variety of light colors, since the UAVs can fly clockwise at a uniform speed, along the outline of the characters.

4.3.2Problems to be solved

In the first part, the "Countdown Clock Display" is achieve by a method similar to a Ferris wheel Sky Show, since it consists of rigid-body movements of regular objects.. In the third part, we firstly extract the scatters on the outline by the same method of realizing the dragon figure. Therefore, the key problem to be solved is the second part "art characters". The aim is to find a route that allows UAVs safely complete the movement in the minimum time. We design an algorithm named "dense scatter diffusion model", which solves the problem of UAV image transformation without collisions, and calculates the sequences when UAVs are shifting images in regard to the principle of reaching targets simultaneously.

4.3.3Dense scatter diffusion model

The algorithm is developed from the consine law. As shown in Figure 22, assuming there are two UAVs, K and E. UAV K starts from point A and movesalong the direction of vector CA.UAV E starts from point B and movesalong the direction of vector CB.Both moves at a speed of **v.**

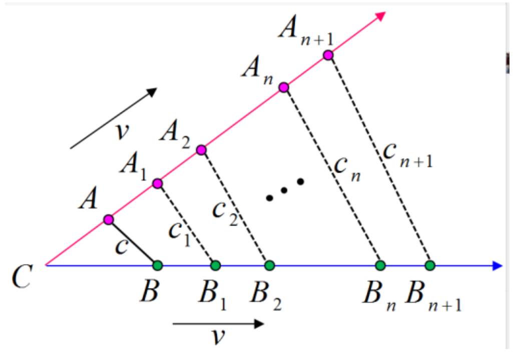
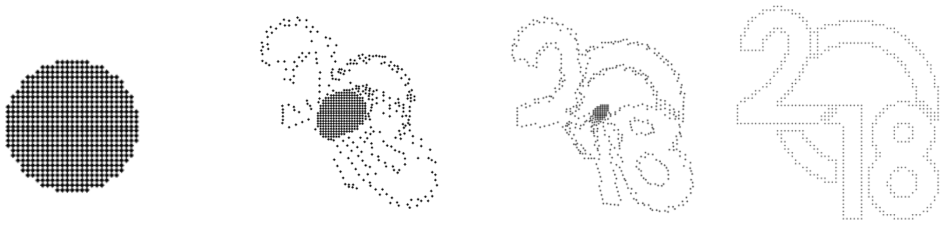


Fig22. Fixed starting points

When the flight time $t=1,\dots,n,n+1$, the positons of UAV K is, $A_1,\dots,A_n,A_{n+1}$ and the positons of UAV E is, $B_1,\dots,B_n,B_{n+1}$ the distances between K and E is, $c_1,\dots,c_n,c_{n+1}$ Therefore, we obtain:

$$c_n^2 = CB_n^2 + CA_n^2 - 2CB_nCA_n\cos\angle C$$
$$= (a+n*v)^2 + (b+n*v)^2 - 2(a+n*v)(b+n*v)\cos\angle C$$
$$c_{n+1}^2 = (a+(n+1)*v)^2 + (b+(n+1)*v)^2 - 2(a+(n+1)*v)(b+(n+1)*v)\cos\angle C$$
$$\Rightarrow c_{n+1}^2 - c_n^2 = 2v(1-\cos\angle C)(a+b+v+2*n*v) \qquad (33)$$

from the formula 33, we can see that when $\angle C = 0°$, $c_{n+1}=c_n$; otherwise, $c_{n+1}>c_n$. In other words, if the two UAVs start from fixed points and diffuse at a uniform linear speed, the distance between them will be farther away or remains constant. Therefore, collisions will never occur.



(a) original (b)(c) Intermediate state (d) target

Fig23. Dense scatter diffusion

In the algorithm, we define Figure 23-a as a dense scatter (original), and 23-d as a sparse scatter (target).In fact, since the dense scatter takes up far less space than sparse scatter, we approximate the dense scatter as a point, which is equivalent to point C in formula 33.The scatter points start at different times and moves to the target map, so that they can reach the designated locations simultaneously. Therefore, UAV collisions in the graphic shifting can be effectively solved via the ‘dense scatter diffusion algorithm’. Specific details are illustrated in the following sections:

4.3.3.1The matching algorithm of corresponding points

Suppose there are N points in the scatter figure, and p is the point outside the figure. Then we mark the distance between point p and the ith point as p_i, then we obtain: $d_i = \min_{i=1,\cdots,N} p_i$ where d_i is the distance from p to the target map.

Since the core of the algorithm is "arriving simultaneously", the farthest point must moves first. In consequence, the scatter matching problem between the original figure(fig.23-a) and the target map(fig.23-d) is converted to the problem of finding the farthest point from the target map. This algorithm can be explained more specifically by the following steps:

Step1:conforming that the original figure and the target map have the same geometric center, which is the origin O in this algorithm. Based on that, acoordinate system is established. Set A is initialized as an array formed by the coordinates of all points on the original figure , where the ith element is denoted as (x_i, y_i); and set B is initialized as an array of coordinates of all the points on the target figure, where the jth element is denoted as $(\tilde{x}_j, \tilde{y}_j)$;

k = 1;

Step 2: Calculate the Euclidean distance between all points in A and B, and save them in matrix D, where:

$$D_{ij} = \sqrt{(x_i - \tilde{x}_j)^2 + (y_i - \tilde{y}_j)^2} \qquad (34)$$

Step 3: Mark The number of elements in set A is M, $M = |A|$; let, $d_i = \min_{j=1,\cdots,M} D_{ij}$ and $s = \max_{i=1,\cdots,M} d_i$ $i_0 = \arg\max_{i=1,\cdots,M} d_i$ $j_0 = \arg\min_{j=1,\cdots,M} D_{i_0 j}$,

so element in set A is the farthest point from set B, and the distance from element (x_{i_0}, y_{i_0}) to (x_{i_0}, y_{i_0}) set B is the distance to the element $(\tilde{x}_{j_0}, \tilde{y}_{j_0})$. Our algorithm controls (x_{i_0}, y_{i_0}) original figure to move to the point $(\tilde{x}_{j_0}, \tilde{y}_{j_0})$ in target map.

Step 4: Let matrix $E(k,:) = (x_{i_0}, y_{i_0}, \tilde{x}_{j_0}, \tilde{y}_{j_0}, s)$, $A = A \setminus \{(x_{i_0}, y_{i_0})\}$ and $B = B \setminus \{(x_{j_0}, y_{j_0})\}$. Delete the the ith row and the jth column in distance matrix D;

Step5: If $A = \varnothing$, the matching work is ended. Otherwise, let k=k+1, return to Step 3.

4.3.3.2Matching algorithm optimization

The time complexity of the current matching algorithm is $O(N^3)$, so the larger the N, the longer the operation time. for the sake of improving the computational efficiency, we optimize the algorithm by using the greedy algorithm. We select several farthest points away from the center in target map (e.g. the first 32 points in matrix B) and the outermost points in original figure (where there are $8\sqrt{N/\pi}$ points in total in each row and column in original figure), and almost obtained the same result. However, the time complexity is decreased to $O(N^{3/2})$ and the running speed is greatly increased.

4.3.3.3Determine the departing delay time

To reach the purpose of reaching the target at the same time, it is required to move the farthest UAV as the first departure, and the other drones need to set off with different delay times. According to the calculation in **4.3.3.1**, the moving distance of the first UAV is $E(1,5)$ and the moving distance of the kth UAV is . $E(k,5)$

If the flight velocity of the UAV is **v,** the delay of the kth UAV departure time can be calculated by . $(E(k,5)-E(1,5))/v$

4.3.3.4Anti-collision optimization

Suppose thereare two last points a_1, a_2 need to be transferred, andthey are expected to move to b_1, b_2 respectively, $|a_1b_1|>|a_1b_2|>|a_2b_1|>|a_2b_2|$, as shown in figure **, in regard to step 2 in 4.3.3.1,it is possible to get the distance matrix D:

$$D=\begin{bmatrix} |a_1b_1| & |a_1b_2| \\ |a_2b_1| & |a_2b_2| \end{bmatrix} \quad (35)$$

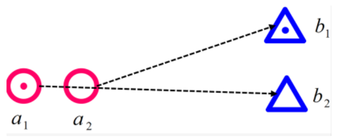


Fig23. Anti-collision optimization

According to the algorithm, a_2 should move to b_1, and a_1 should move to b_2.Since $|a_1b_2|>|a_2b_1|$, a_1 must firstly departure, so a collisions may occur.

The solution is: if the rear point should move farther, then the departure time is synchronized with the front one, which means a_1 and a_2 start at the same time. In the simulative trajectory program, this optimization significantly reduces the occurrence frequency of collisions.

4.3.5 The conclusion and analysis of Pattern shifting model

The pattern switching model, is implemented via simulation by computer. It is proven that the model can not only effectively prevent collisions of UAVs, but also quickly shift between different images. Therefore, the scheme is operational for UAV group formation and can be promoted to the following conditions:

1. The source map is a sparse lattice graph, and the target figure c is a solid circle: this case is the inverse operation of the *'dense scatter diffusion algorithm'*, which only needs to switch the starting and reaching points, and starting at the same time;

2. The source map and the target map are sparse-dot matrix graphs: at first we need to generate a dense solid dot matrix graph from the source map, then convert it to the target map. Therefore, two random 2D patterns can be shifted.

3. The source map and the target map are sparse lattice 3D graphs, which only need to replace the distance equation by the 3DEuclidiandistance equation. A dense solid dot matrix graphic then can be generated by the d*ense scatter diffusion algorithm*, which is as the same as achieving the shift of any two 3D patterns.

V. Sky light display project design and Simulation effect display

Assume adopting Intel shooting stars drones in the project[1], we researched and found that the size of the drone is 384*384*93（mm）, with a maximum rising velocity of 5m/s, a maximum horizontal flight speed of 65km/h(18m/s), a maximum flight duration of 27 minutes without wind, and a maximum flying height no more than 400 feet (120 meters). In order to guarantee both the display effect and safety of the sky light show, we all agree to set the maximum speed of the drones to 5m/s, with a minimum safety space between each drone of 1m*1m*0.5m.

5.1 Overall planning of the project

The overall planning needs to determine the performing area, number of drones, performing time and budget. Based on the information and the result of the proposed model, we would give the implementation details of Ferris Wheel, Western Dragon and New year clock projects, including the initial location and moving path of drones, and afterwards do the simulation on MATLAB, in order to determine the complete Sky light display project plan.

5.1.1 Sky performing area suggestion

In order to improve the ornament of the sky light display, we suggest to take the performing area and watching area in Figure 24. It's split into three parts: The first part is the performing area, which is the vertical space of 80m*80m in the front, with 5m between the ground the lowest point; The middle part is an empty space of 100m*100m; and the back part is a watching space between 50m*100m, the looking up perspective angle for the most front row audience is 24degrees, the maximum looking up perspective angle is 40 degrees, the maximum looking perspective angle of the last row audience is 30 degrees. This kind of designing project, can make the audience watching in a small distance without getting exhausted.

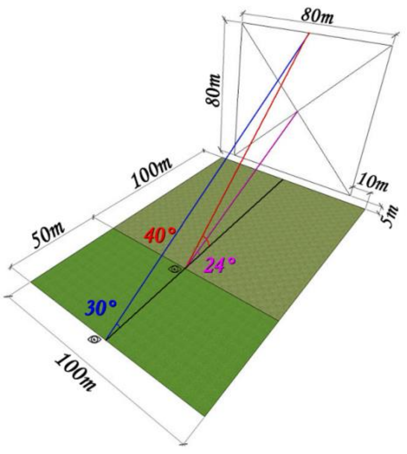


FIG24: The distribution of drones of the clock pictures

5.1.2 Minimum numbers of drones

There were four images to be displayed in the air, which are Ferris wheel, western dragon, timing clock and '2018' in artistic character. We choose the timing clock image as standard to set the minimum number of drones, other images may increase the number of drones depending on budget. The more drones included in the display, the sharper the images would be, and as well more vivid.

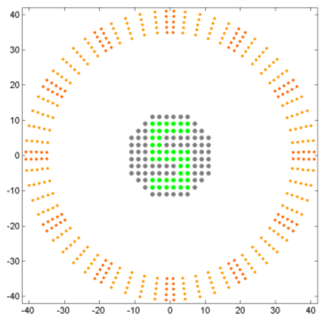


FIG25: drones numbers of "clock"

The reasons for choosing the clock Figure as the standard is listed as follows:

1. The center of the circle is an image with condense pixels, in order to show the digital timing completely, the number of drones to be used is relatively fixed.

2.The number of drones distributed on every circle are relatively fixed, i.e. 60 drones represent the clock scale. According to this principal, the minimum number of drones need for the project we calculated are listed below:

$$N=60*5+(6+10)*3+6*12=420$$

5.1.3 Minimum landing field area

Based on the minimum 420 drones for performance, with 10 reserving drones, 430 drones are arranged in a matrix of 20*22 on the ground. We set the arranged space for each drone as 1m*1m, then the minimum landing field area would be 430 $m^{2.}$

5.1.4 Maximum performance duration

Based on the Intel shooting stars drone parameter, the maximum flight time under a static air condition is 27 minutes. In our performance plan, drones are subject to frequent speed changes and sometimes full speed status is reached, therefore for safety reason, we set the duration of the performance to be no more than 15 minutes.

5.1.5 Minimum Budget

The Intel Shooting star drones could be either bought or rent, we recommend to rent the drones for economic consideration. Based on the unit price of 30 dollars/drone, we calculate that the minimum equipment cost is 12,600 dollars. Other costs, such as design costs, site preparation, human resource and administrative costs are not considered here.

5.2 Detailed Project Implementation

5.2.1 Ferris Wheel Performance Project

1. Parameter setting

With referencing to the structure of the Ferris Wheel, under the constraint of the project planning, we calculated and calibrated the parameter for the Ferris wheel displaying, and gain information of the distributed location of drones of each part, sky performance area, launching time of drones and performance duration.

Table2 Ferris Wheel Parameter setting table

Names of Structures	parameters	Definition of the parameters	Values	S.T.
Outer rim	R_1	Radius of the Outer rim	30m	Performance area
	n_7	The number of drones distributed on Outer rim	76	$\le 2\pi * R_1 \approx 188$
Second rim	R_2	Radius of the Second rim	28m	$\le R_1 - 1 = 29$
	n_8	The number of drones distributed on Second rim	76	$\le 2\pi * R_2 \approx 176$
Inner rim	n_5	Number of inner rim	3	Effect design
	n_6	The number of drones distributed on Inner rim	30	$\le 2\pi * r_1 \approx 31$
	r_1	Minimum radius under motion state of Inner rim	5	$R_1 / 6$
	r_2	Maximum radius under motion state of Inner rim	26m	$\le R_2 - 1 = 27$
Spoke	n_1	number of axes on the wheel	8	$= n_3 = 8$
	n_2	The number of drones distributed on every spoke	8	$\le R_2 - 1 = 27$
Triangle stand	L	1/2 horizontal length of the triangle stand	9m	$= \tan(\pi/12) * (H + H') \approx 9$
	H	The distance between the horizontal stand of the triangle stand and the lowest point of the outer rim	5m	> r
	H'	The height of the horizontal side of the triangle	5m	<80-2*R1-H=15m

		stand from the ground		
	m1	The number of drones distributed on the left and right side of the triangle stand	20	<sqrt(81+(H+ R1)^2)=36
	m2	The number of drones distributed on the horizontal side of the triangle stand	13	<2 L=18
cabin	r	Radius of the cabin	3m	Effect design
	n3	Numbers of the cabin	8	Effect design
	n4	The number of drones distributed on the cabin	8	<2pi* r=18

Based on the table above, and total number of the drones as 420, we can get the number of drones in every part, and meanwhile calculated the following:

(1) The area of performance **S** in the sky (The minimum rectangular area of the Ferris Wheel)

$$S=2（R1+r）*（2\ R1+r+H）=66m*68m=4488m^2$$

(2) Launching time for drones:

Assume that the drone is doing a uniformly accelerating motion with acceleration of **a** from static to speed **v**. After which it starts to fly in uniform speed, then do a uniformly decelerated motion with acceleration of –**a** before it arrive the destination. For distance **S**, it could be divided into 3 parts below for calculation:

Beginning stage: Doing a uniformly accelerated motion from static to speed **v**,

$$t_1=\frac{v}{a} \qquad S_1=\frac{vt_1}{2}=\frac{at_1^2}{2}=\frac{v^2}{2a}$$

Middle stage: Doing a uniform speed motion $S_2=vt_2$

Last stage: Doing a uniformly decelerated motion from speed v to static:

$$t_3=\frac{v}{a} \qquad S_3=\frac{vt_3}{2}=\frac{at_3^2}{2}=\frac{v^2}{2a}$$

Therefore, the relationship between distance s and time t is shown by the expression below:

$$t=\begin{cases} t_1+t_2+t_3=\dfrac{aS+v^2}{av} & \text{if } S>v^2/a \\ \mathrm{t_1+t_3}=\sqrt{\dfrac{2\mathrm{S_1}}{\mathrm{a}}}+\sqrt{\dfrac{2S_3}{a}}=2\sqrt{\dfrac{S}{a}} & \text{otherwise} \end{cases} \qquad (36)$$

Based on formula(36), the calculation of the time for drone to finish the first image is:

If : $a=1(m/s^2)$ $v=5(m/s)$ $S=2\mathrm{R}_1+H+H'=70(m)$

Then: $t=\dfrac{aS+v^2}{av}=\dfrac{70+5^2}{5}=19(\mathrm{s})$

Set: Interval $\Delta t=5(s)$,then the time T can be expressed as: $T=t+5\Delta t=44(s)$

(3) Wheel Turning Time in the Air:

The minimum time of Ferris Wheel turning one round in the sky is equal to the minimum flight time of the outmost drone to finish a full circle, it can be expressed as:

$$\mathbf{t}=2*pi*(R1+2r)/v=45s$$

2. Effect Simulation of the Project

We have simulated the Ferris wheel project in MATLAB, and get the Figure below as the result. Figure26 is the dynamic simulation effect, Figure 27 is the simulation effect of the Ferris wheel when it is turning2. Effect Simulation of the Project

We have simulated the Ferris wheel project in MATLAB, and get the Figure below as the result. Figure26 is the dynamic simulation effect, Figure27 is the simulation effect of the Ferris wheel when it is turning.

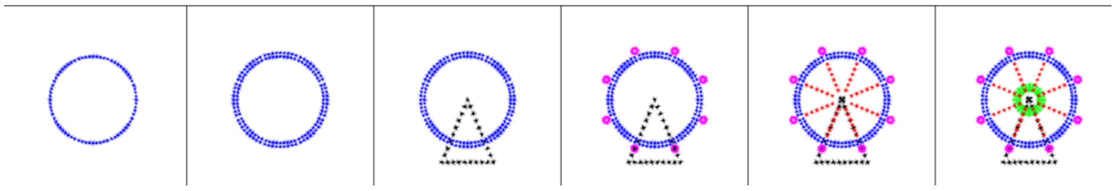


Fig26. drones launching

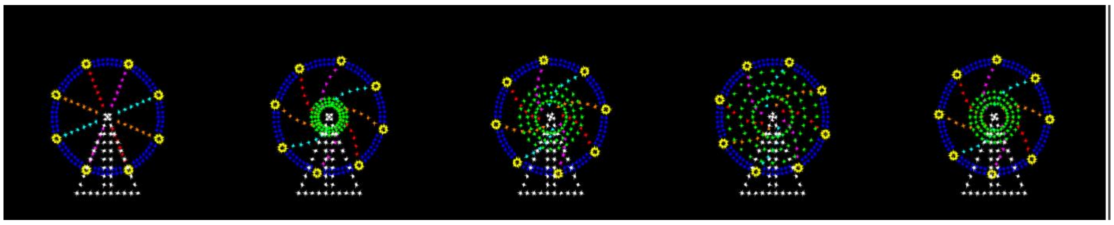
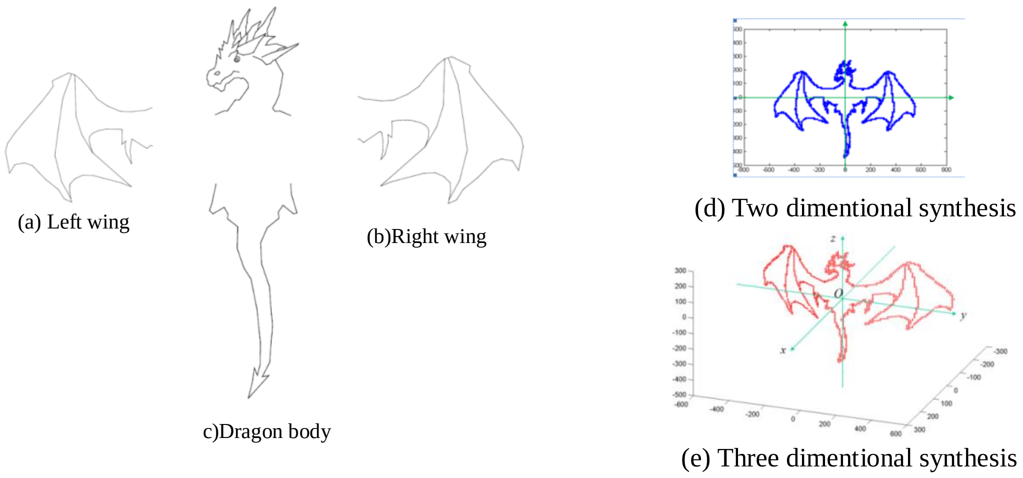


Fig27. Ferris wheel turning

5.2.2 Western Dragon Dancing show project

1. Parameter setting

Assume that the number of drones put into the Western Dragon Dancing show is 420, using pattern changing method to change the Ferris Wheel Figure into the initial Figure of Western Dragon, (See 5.3 about the pattern changing method). As the Dragon Figure is relatively complicated, first we split the dragon into three parts: Body, Right Wing and Left Wing(Shown in figure28).Then we draw the scatter points on the boundary separately for each part, and combine the scatter points aftwewards. Eventually we find the parameters below:



(a) Left wing
(b)Right wing
c)Dragon body
(d) Two dimentional synthesis
(e) Three dimentional synthesis

FIG28: Structure Design of Western Dragon and Synthesis

Table3 Western Dragon Parameter setting table

parameters	Definition of Parameters	Values	S.T.
n9	Number of drones on the dragon's body	126	Scatter figure
n10	Number of drones on the left wing of the dragon	147	Scatter figure
n11	Number of drones on the right wing of the dragon	147	Scatter figure
W	Width of the 2-D dragon	80m	Two dimentional synthesis scatter figure
H	Height of the 2-D dragon	60m	
θ	Largest angle between the wings and the body axis	$\pi/6$	

Based on the table above, and total number of drones as 420, we can get the number of drones of every part from scatter calculation, and meanwhile get:

(1) Performing area S in the sky (rectangular area of Dragon figure)

S=W*h=80m*60m=4800m2

(2)Time of Figure Changing

Including the time t1 that is need for changing the Ferris Wheel into a sphere as intermediate state, then the time t2 that is need for changing the sphere into western dragon.

Assume that the drones of sphere is distributed in a 1m*1m space, **r** as radius of sphere, then: $r=\sqrt{420/\pi}\approx 11$

Set the maximum distance of Changing Ferris Wheel to filled circle is S_1, then:

Set the maximum $S_1=(R_1+H)/\cos(\pi/12)-r\approx 25$ distance of transforming sphere to Western Dragon is S_2, then: $S2=w/2-r=29$

According to formula 36, set the whole transforming period to t, then:

Then: $$T=t_1+t_2=\frac{aS_1+v^2}{av}+\frac{aS_2+v^2}{av}=\frac{25+5^2}{5}+\frac{29+5^2}{5}\approx 21(s)$$

(3)Performance Duration

The minimum time t need for Dragon performing a set of actions in the air equal to the minimum time of the drones on the outermost part of the dragon wings rotate around the center of the body by angle of 4θ, the calculation formula is list as below:

$$4\theta*(w/2)/v=4*\frac{\pi}{6}*60/5\approx 25(s)$$

We set the time duration for each action of the Dragon Perfromance to be 30 seconds, based on total of 4 actions we have in plan, it requires2 minutes to play the entire group of actions.

2.Effect Simulation of the Project

We simulated the animation design project in MATLAB, the figure of effect of Dragon dance project is presented below:

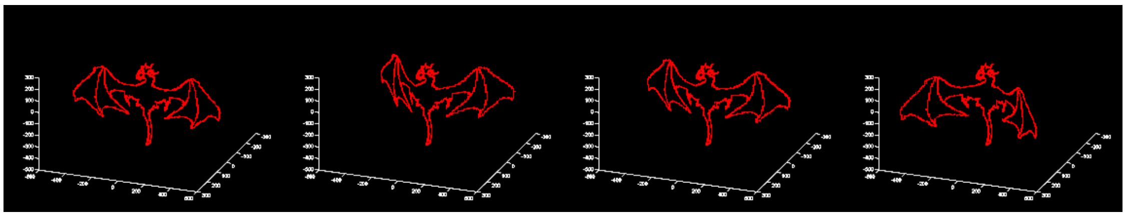
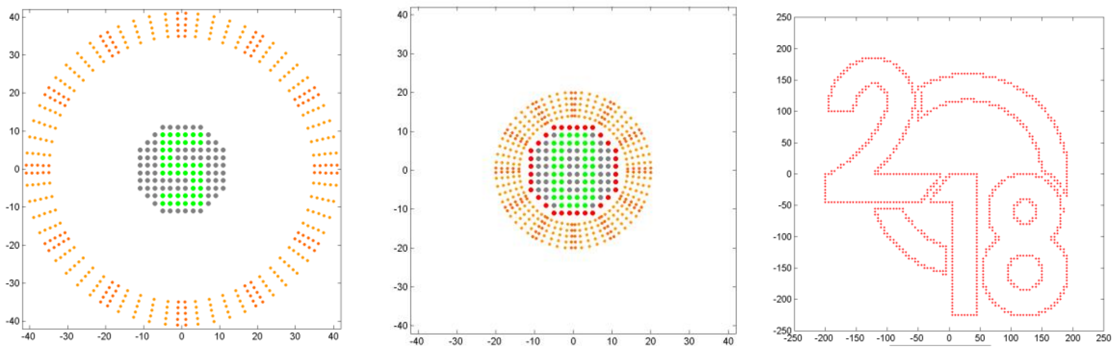


Figure29 Western Dragon dancing show simulation effect

5.2.3 New Year Celebration Plan – "Clock in Sky"

1.Attribute settings of the "Clock"

The design we made for the "clock" is shown in Fig. 30. The attributes we used on the pattern of the clock was inspired by the method we utilized in the Ferris Wheel program. The presentation of the figure of "2018" was inspired by the Western Dragon program. The specific attributes settings we used are shown in Chart 4. We leaned heavily on using the attribute chart to calculate the time of transition and ultimately calculate the time needed to complete this performance.



（a）Initial Clock （b）Last Figure of Clock （c）Figure after the Change

Fig30. The Different Steps of the Performance

Table4 “clock” Parameter setting table

Components	Variables	Meaning of Variables	Values	S.T.
Clock Circles	m3	Number of Outer circles	5	Result Design
	R31	Initial radius of outermost circle	30 m	Size of land needed for performance
	R35	Initial radius of innermost circle	26 m	<=R31-4=26
	n12	Number of drones needed for each circle	60	Show the time
	R31’	Final radius of outermost circle	20	> R35’+4 =15
	R35’	Final radius of innermost circle	16	>60/2*Pi=11
Inner digital display	n13	Number of drones needed	120	8-bit pixel Figure
	d	Distance between drones	2 m	>1m Safe distance

According to the chart above, we can calculate the area and time we would need for the performance:

(1)Area: S=π* R31* R31=2826 m^2

(2)Time for Transition:

Similar to the calculation for the transitions of the Western Dragon, we used a variable – S_2 (S_2 = 19m) to control the longest distance when changing from the intermediate sphere into the “counting-down” figures. (The variable we used in Western Dragon was S_1 = 29m （$< v^2/a = 25m$）). Assume the transition time is T, then:

$$T = \frac{aS_1 + v^2}{av} + 2\sqrt{\frac{S_2}{a}} = \frac{29+5^2}{5} + 2\sqrt{\frac{19}{1}} \approx 20(\mathrm{s})$$

(3)Duration for each phase: 5 sec for “counting-down”; 5 sec for “Performance time”

2.The Attributes for the Artistic Letter “2018”

Table5 “2018” Artistic Letter Parameter setting table

Variables	Definitions	Values	S.T.
W	Width of the figure	60m	Area needed
h	Height of the figure	60m	Area needed
n14	Number of drones for “2018”	420	Standardized value

(1)Performance Area: S = 3600 m^2

(2)Time for Transition:

From “counting-down” figure “0” to a intermediate sphere would require the longest distance of S_1 = 9m. From the intermediate sphere to the letters “2018” would require the longest distance of S_2 = 19m. Assume transition time as T, according to formula (43):

$$T = 2\sqrt{\frac{S_1}{a}} + 2\sqrt{\frac{S_2}{a}} = 2\sqrt{\frac{19}{1}} + 2\sqrt{\frac{9}{1}} \approx 15(\mathrm{s})$$

(3)Duration of performance: 30 sec. (In the air).

3.Simulated Result of Transition of the Figures (Reference Fig23)

5.4 Conclusion for Drone Performance Plan

This performance would require 420 drones and 10 spare ones. The landing zone would require a minimum area of 430m^2. The maximum area needed for the presentation of the Western Dragon would be 4800m^2.The minimum area needed for the “counting-down” performance would be 2826m^2. The schedule as table6,the whole performance would require 10 minutes. Since it costs $30 to rent a single drone, the entire operation would cost approximately $12600.

Table6 Drone Performance Plan

program	task for Drones	Limit time	Suggested time
Ferris Wheel	Launch and display	44s	60s
	spinning	45s/cycle	240s
Dragon	display	21s	30s
	Dragon show	30s/cycle	150s
“clock”	display	20s	30s
	counting-down	5s	10s
“2018”	Shift	15s	20s
	display	30s	60s
Total time		210s	600s

VI. Strengths and Weaknesses

6.1 Strengths

1. We met the problem’s requirement and completed the Ferris Wheel, Western Dragon’s static and dynamic model. Besides that, we also came up with our own plan of the “counting-down” show for the celebration of the new year.
2. We utilized Matlab in order to build both the figures and how the drones would behave in both 2-dimensional and 3-dimensional space. Thus, we illustrated our capability of doing simulation with Matlab and design ingpractical flight plans.
3. The program we built has certain level of universality. The skeletal structural program and skin adhering program could be used to design transition of any figure in both 2D and 3D space.
4. We creatively designed an algorithm that controls how dots spread in a clustered shape, hence effectively solved the collision problem for the drones that we had to deal with. This algorithm helped us to implement the transition between two different figures.
5. The program was designed so that customers could simply change the values assigned to each variable in order to satisfy their own needs. It is simple enough for people to take a few glances and figure out what each line means.

6.2 Weaknesses

In regard to the time limitation, it is comparablely simple when realizing the three-dimensional in the non-rigid movement, however, the biological features like wagging effect[18] in the movement failed to be considered in the paper. Furthermore, when calculating the conversion of skin attachment points, we merely think of the rotating conversion rather than the non-rigid like conversion for the skin itself.

VII. Further Work

1. Fix the flaws in our program and model.
2. Improve the ways to solve collision problems. We could implement “collision-detect” algorithms to prevent such thing from happening in order to reach a higher security level.
3. We could combine the capability of the drones, the controlling system with our mathematical model in order to improve how our show is being presented.

VIII. Reference

[1] "Intel® Shooting Star™." Intel, www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html. Accessed 18 Nov. 2017.

[2] Shapiro, Joel. *Rigid Body Motion*Https://www.physics.rutgers.edu/~shapiro/507/book5.pdf ed., New Jersey, Rutgers, 2010. 12 vols.

[3]Dill, Andreas R., and Martin D. Levine. Non-Rigid Body Motion. Http://graphicsinterface.org/wp-content/uploads/gi1985-55.pdf ed., Montreal. Quebec, McGill University.

[4]R., Merzouki, and Samantaray A. K. *Models of Actuators*. PDF ed., London, Springer London, 2012.

[5]Mithraratne, K., and T. Wu. "Accurate Modeling of Finger Kinematics Using a Statistically-Reduced Rigid Body Model."*IFMBE Proceedings*, vol. 43, pp. 303-21.*GoogleBooks*,books.google.com/books?id=ITBBAAAQBAJ&pg=PA303&dq=rotation+link+motion+model&hl=zhCN&sa=X&ved=0ahUKEwjU9O3po8fXAhXCNJQKHbJFAxcQ6AEINzAC#v=onepage&q=rotation%20link%20motion%20moel&f=false. Accessed 18 Nov. 2017.

[6]Weblog post. *SimplyMaya*, 10 Dec. 2002, simplymaya.com/forum/showthread.php?t=4405. Accessed 18 Nov. 2017.

[7]J, Ramsay, and Silverman B. W. *Visualizing the results*. PDF ed., New York, Springer New York, 2006.

[8]Agustin. "How to Plot the Sample Mean in a Scatter Plot?"*MathWorks*, 4 Aug. 2016, www.mathworks.com/matlabcentral/answers/266667-how-to-plot-the-sample-mean-in-a-scatter-plot? Accessed 18 Nov. 2017.

[9]Driessche, P.van Den, and James Watmough. "Reproduction Numbers and Sub-threshold Endemic Equilibria for Compartmental Models of Disease Transmission."*Reproduction Numbers and Sub-threshold Endemic Equilibria for Compartmental Models of Disease Transmission*, digital ed., pp. 1-21.

[10]Esser, Ernie. "UCI Interdisciplinary Computational and Applied Mathematics Program."*Linear Spatial Filtering*, edited by SungjinAhn, PDF ed., UCI, 2014. Excerpt originally published in *ICAMP*, pp. 1-4.
[11]Fletcher, Tom, editor. "Spatial Filtering" ["Image Processing Basics"]. *Coe.utah.edu*, Utah, 31 Jan. 2012, www.coe.utah.edu/~cs4640/slides/ Lecture5.pdf. Accessed 18 Nov. 2017.
[12]Finnigan, Tom. "Blending Function."*Exploring Blending Function*, PDF ed., pp. 39-44.
[13]"Function B = Boundaries(BW, Conn, Dir)."*Fourier.eng.hmc*, 9 Mar. 2015, fourier.eng.hmc.edu/e161/dipum/boundaries.m. Accessed 18 Nov. 2017.
[14]Bultheel, Adhemar, and Erik Hendriksen. "Gonzales's Boundaries Function." *Orthogonal Rational Functions*, 4th ed., Cambridge, Cambridge University, 2007, pp. 155-72.
[15]"Function B = Bsubsamp(A, Conn, Dir)."*Fourier.eng.hmc*, 9 Mar. 2015, fourier.eng.hmc.edu/e161/dipum/boundaries.m. Accessed 18 Nov. 2017.
[16]"Rigid Body Kinetics with Plane Motion (Part 1) - Engineering Dynamics." *YouTube*, structurefree, 8 June 2010, www.youtube.com/ watch?v=PZGR-lYxiv8. Accessed 18 Nov. 2017.
[17]E., Lagnese j., et al. *Splitting of eigenvalues*. Boston, Birkhäuser Boston, 2012. 34 vols.
[18]Coren, Stanley. "What a Wagging Dog Tail Really Means: New Scientific Data."*Psychology Today*, 5 Dec. 2011, www.psychologytoday.com/blog/ canine-corner/201112/what-wagging-dog-tail-really-means-new-scientific-data. Accessed 18 Nov. 2017

IX .Appendix

Program Code

[Code 1]: Spinning ferris wheel in MATLAB

```
% This script is aiming to plot the dynamic record, by saving vast picture of each
%movement, and representing the overall process of one single cycle of the ferris wheel.

clear; clc; close all;
R1 = 30;    % Radius of the Outer rim
R2 = R1-2;    % Radius of the Second rim
r = 3;    % Radius of the cabin
beta = 5*pi/12;    % the angle of left-stand to the horizontal horizon.
H = 10;    % The distance between the horizontal stand of the triangle stand and the lowest
           % point of the outer rim
L = (R1+H)/tan(beta);    %1/2 horizontal length of the triangle stand

m1 = 20;    % The number of drones distributed on the left and right side of the triangle stand
            %(including the origin)
m2 = 13;    % The number of drones distributed on the horizontal side of the triangle stand

n1 = 8;      % number of axes on the wheel
n2 = 8;      % The number of drones distributed on every spoke
n3 = 8;      % Numbers of the cabin
n4 = 8;      % The number of drones distributed on the cabin
n7 = 64;    % The number of drones distributed on Outer rim
n6 = 30;    % The number of drones distributed on Inner rim

r1 = R1 / 6;    % Minimum radius under motion state of Inner rim
r2 = R1 - 6;    % Maximum radius under motion state of Inner rim
n5 = 3;         % the number of lines if the inner rim
minnr = 2;    %the minimum distance from each inner rim, the distance
              %between when suppressing into small loop
maxnr = 4;    %the maximum distance between each outer rim, the distance between
               %when expanding to the maximum loop
T1 = 6;         %the change of time of inner rim radius per rotation period
picnum = 60;    % the number of picture produced per rotation period
tr = 2 * picnum / T1;    %the number of picture corresponded to the change time of inner rim
cn=6;           % the correspond number of picture produced when one point on the triangle stand
move to the next point

num = 0;      % the number of produced picture
for theta = 0:pi/picnum:2*pi    %every angle it will generate one picture
    num = num+1;
    figure;
    hold on;
    axis([-R1-2*r-3 R1+2*r+3 -R1-H-3 R1+r*2+3]);
    axis square
    % sketching black lines represented the nighttime sky
    for x=-R1-2*r-2 : 2 : R1+2*r+2
        line( [x x], [ -R1-H-3, R1+r*2+3], 'LineWidth', 23, 'Color', 'k');
    end;

    etatemp = 0:2*pi/n3:2*pi; % the angle of triangle stand where the cabin is held
    % plot the cabin of the ferris wheel
    for t = 1:length(etatemp)
        deltatemp = 0:2*pi/n4:2*pi;
        xtemp = (R1+r)*cos(etatemp(t)+theta)+r*cos(deltatemp);
        ytemp = (R1+r)*sin(etatemp(t)+theta)+r*sin(deltatemp);
        plot(xtemp,ytemp, 'yo','Linewidth',1,'Markersize',5);
    end
    % plot the cabins of gravitational ferris wheel
    %   for t = 1:length(etatemp)
    %       deltatemp = 0:2*pi/n4:2*pi;
    %       xtemp = R1*cos(etatemp(t)+theta)+r*cos(deltatemp);
    %       ytemp = R1*sin(etatemp(t)+theta)-r+r*sin(deltatemp);
    %       plot(xtemp,ytemp, 'yo','Linewidth',2,'Markersize',4);
    %   end

    % plot the outer rim
    theta0 = 0:2*pi/n7:2*pi;
    x = R1*cos(theta0+theta);
    y = R1*sin(theta0+theta);
    plot(x,y,'b.','Linewidth',4,'Markersize',8);
    %plot the second outer rim
    x = R2*cos(theta0+theta);
    y = R2*sin(theta0+theta);
    plot(x,y,'b.','Linewidth',4,'Markersize',8);

    %using stars to represent the plotting of the arc dragline
    co = ['r' 'c'];
    B = [cos(pi/3), -sin(pi/3);+sin(pi/3),cos(pi/3)];
    for i = 1 : n1
        for j = 1 : n2-1
            angle_j = j*pi/3/n2;
            C = [cos(angle_j),sin(angle_j);-sin(angle_j),cos(angle_j)];
            A = B * R1 * [cos(2*pi*i/n1+theta-pi/12),sin(2*pi*i/n1+theta-pi/12)]';
            xy = ((B+C-C*B)*A)';
            plot(xy(1),xy(2),[co(rem(i,length(co))+1) 'd'],'Linewidth',1,'Markersize',4);
        end;
    end
    %plot the inner rim
    theta0 = 0:2*pi/n6:2*pi;
    for t = 1:n5
        tempr = (r2-(t-1)*maxnr) - abs( rem(num-1,2*tr)-tr )/tr*((r2-(t-1)*maxnr) ...
- (r1+(n5-t)*minnr));
        x = tempr*cos(theta0+theta);
        y = tempr*sin(theta0+theta);
        plot(x,y,'gp','Linewidth',1,'Markersize',4);
    end;

    % plot the triangle stand
    % plot the left-side of the triangle stand
    xtemp = -L+L/(m1-1)*rem(num-1,cn)/cn : (L/(m1-1)) : 0;
    ytemp = tan(beta)*xtemp;
    plot(xtemp,ytemp,'w^','Linewidth',2,'Markersize',3);
    % plot the right-side of the triangle stand
    xtemp= L/(m1-1)*rem(num-1,cn)/cn : (L/(m1-1)) : L;
```

```
        ytemp=-xtemp*tan(beta);
        plot(xtemp,ytemp,'w^','Linewidth',2,'Markersize',3);
        % plot the horizontal line of the triangle stand
        xtemp = L-2*L/(m2-1)*rem(num-1,cn)/cn : -2*L/(m2-1) : -L;
        ytemp=-tan(beta)*L*ones(size(xtemp));
        plot(xtemp(1:end),ytemp(1:end),'w^','Linewidth',2,'Markersize',3);

        plot(0, 0, 'wo', 'Linewidth',3,'Markersize',6);    %plot the origin
        set(gcf,'color','k');

        % save the pictures to the file
        saveas(1,[cd '/pic/FerrisWheel' num2str(num) '.png'],'png');
        close;
end
```

[Code 2]: Dance of dragon in MATLAB.
[Code 2-1]:f_GetScatter.m

```
%% The image preprocessing function
function scatter = f_GetScatter( picfile )
% utilize the Gonzalez digital image processing technology, extracting the profile
%of the digital image and sampling the sactters.
%    Input: picfile, the file for laying all the produced figures with
%full route, e.g. 'c:\Himcm2017\pic\dragon_body.png'
%    Output: scatter, representing by matrix where reserving the pixel coordinates of scatters.

  % binarization parameter, where the larger the parameter, the more the slecting scatters
alpha = 0.99;
% load the pics.
x = imread(picfile);
% transferring multicolor to grayscale
y = rgb2gray(x);
% Threshold segmentation: binarization processing
f = im2bw(y,alpha);
g= ~f;
% pic size, M: the number of columns, N: the number of rows
[M,N] = size(g);
%the processing of image edge
B = boundaries(g,8,'cw');
bim = 0;
scatter = 0;
for i = 1:length(B)
        b = B{i};
        if size(b,1)>20
                xmin = min(b(:,1));
                ymin = min(b(:,2));
                bimtemp = bound2im(b,M,N,xmin,ymin);
                bim = bim+bimtemp;
                %2nd time for sampling the edge
                [s,su] = bsubsamp(b,8);
                % transmiting the edge to binary graphic
                gtemp = bound2im(s,M,N,min(s(:,1)),min(s(:,2)));
                scatter = scatter+gtemp;
        end
end
end
```

[Code 2-2]: dragon.m

```
%%% the simulative session of dragon
% this model manages to extract the pixel points from the original picture and simulate the
% where we separate into two parts: the body and the wings, and respectively extract
%their profiles and then combind them

clc; clear;    close all;
%input the library routine of “Gonzalez digital image processing”
addpath(genpath('pictoolbox'));
pix2cm = 1;

% Step 1 input the left wing1 of the wetern dragon
gg1 = f_GetScatter('dragon_leftwings1.png');
% Step 2 input the left wing of the wetern dragon
gg2 = f_GetScatter('dragon_leftwings.png');
gg_leftwings = logical(gg1+gg2);
% Step 3i nput the body of the wetern dragon
gg3 = f_GetScatter('dragon_body.png');
gg_body = gg3;
% Step 4 input the right wing of the wetern dragon
gg4 = f_GetScatter('dragon_rightwings.png');
% Step 5 input the right wing 1 of the wetern dragon
gg5 = f_GetScatter('dragon_rightwings1.png');
gg_rightwings = logical(gg4+gg5);

% obtain the size of the pics
[M N] = size(imread('dragon_rightwings1.png'));
% this is the coordinate of the origin
cpix = [633 342];
xtemp = ([1:N] - cpix(1))*pix2cm;
ytemp = ([1:M] - cpix(2))*pix2cm;
ytemp = -ytemp;
XX = ones(M,1)*xtemp;
YY = ytemp'*ones(1,N);

% Step 6 setting the yoz axises where the body lies, establishing the three- dimensional coordinate
system
xdot_body = XX(gg_body==1);
ydot_body = YY(gg_body==1);
body_3d = zeros(3,length(xdot_body));
body_3d(2:3,:) = [xdot_body';ydot_body'];
rotation1 = min(body_3d(2,:))-9;
rotation2 = max(body_3d(2,:))+9;

% calculating the three-dimensional coordinates of the wings
xdot_leftwings = XX(gg_leftwings==1);
ydot_leftwings = YY(gg_leftwings==1);
leftwings_3d = zeros(3,length(xdot_leftwings));
leftwings_3d(2:3,:) = [xdot_leftwings';ydot_leftwings'];
```

```
xdot_rightwings = XX(gg_rightwings==1);
ydot_rightwings = YY(gg_rightwings==1);
rightwings_3d = zeros(3,length(xdot_rightwings));
rightwings_3d(2:3,:) = [xdot_rightwings';ydot_rightwings'];

% Step 7 plot the three-dimensional coordinates of the body and wings in the original condition
figure(1);
plot3(body_3d(1,:),body_3d(2,:),body_3d(3,:),'r.');
hold on;
plot3(leftwings_3d(1,:),leftwings_3d(2,:),leftwings_3d(3,:),'r.');
plot3(rightwings_3d(1,:),rightwings_3d(2,:),rightwings_3d(3,:),'r.');
axis([-300 300 -600 600 -500 300]*pix2cm);
az = 110.5;
el = 34.0;
view(az,el);
saveas(1,[cd '\pic\dragon' num2str(1) '.png'],'png');
close;

% Step 8 make wings move, so that the scatters on the wings can rotate with the intersecting
%line of y=rotation1 and x=0
angleStep = 21;
theta1 = linspace(0,-pi/6,angleStep);
theta2 = linspace(-pi/6,pi/6,2*angleStep-1);
theta3 = linspace(pi/6,0,angleStep);
theta_left = [theta1 theta2(2:end) theta3(2:end)];
theta_right = -theta_left;
angleStep = length(theta_left);
leftwings_3dCell = cell(1,angleStep);
leftwings_3dCell{1} = leftwings_3d;
leftwings_dotNum = size(leftwings_3d,2);
rightwings_3dCell = cell(1,angleStep);
rightwings_3dCell{1} = rightwings_3d;
rightwings_dotNum = size(rightwings_3d,2);

for i = 2:angleStep
    figure(1);
    % Step 8.1 plot the three-dimensional coordinates of the body
    plot3(body_3d(1,:),body_3d(2,:),body_3d(3,:),'r.');
    hold on;
    % Step 8.2 calculating the three-dimensional coordinates of the body
    Atemp_left = [cos(theta_left(i)) -sin(theta_left(i)) 0; ...
        sin(theta_left(i)) cos(theta_left(i)) 0 ; 0 0 1];
    leftwings_3dtemp = Atemp_left*(leftwings_3dCell{1}-[zeros(1,leftwings_dotNum); ...
        rotation1*ones(1,leftwings_dotNum);zeros(1,leftwings_dotNum)])...
        +[zeros(1,leftwings_dotNum);rotation1*ones(1,leftwings_dotNum); …
        zeros(1,leftwings_dotNum)];
    leftwings_3dCell{i} = leftwings_3dtemp;
    plot3(leftwings_3dtemp(1,:),leftwings_3dtemp(2,:),leftwings_3dtemp(3,:),'r.');
    % the rotation matrix of reeling y-axis in the angle of theta
    Atemp_right = [cos(theta_right(i)) -sin(theta_right(i)) 0; ...
        sin(theta_right(i)) cos(theta_right(i)) 0; 0 0 1];
    rightwings_3dtemp = Atemp_right*(rightwings_3dCell{1}-[zeros(1,rightwings_dotNum); ...
        rotation2*ones(1,rightwings_dotNum);zeros(1,rightwings_dotNum)]) ...
        +[zeros(1,rightwings_dotNum);rotation2*ones(1,rightwings_dotNum); …
```

```
            zeros(1,rightwings_dotNum)];
        rightwings_3dCell{i} = rightwings_3dtemp;
        plot3(rightwings_3dtemp(1,:),rightwings_3dtemp(2,:),rightwings_3dtemp(3,:),'r.');
        view(az,el);
        axis([-300 300 -600 600 -500 300]*pix2cm);
        saveas(1,[cd '\pic\dragon' num2str(i) '.png'],'png');
        close;
end
```

[Code 3]: Pattern shifting in MATLAB

```
function f_dsda(inputfile,savefile)
%f_dsda: the function of dense scatter diffusion algorithm
%       inputfile: input the name of the document, including the route
%             and Filename Extension, the format is 8-code bmp document.
%                         e.g.：  'c:\himcm2017\FerrisWheel.bmp'
%       savefile: save the file, including the route and the previous part of
%             the file, e.g.：  'c:\himcm2017\png\fw2cir', automatically ,
%           the system will create the png file with sequential number.
%      e.g. the name of the no.100 file is : 'c:\himcm2017\png\fw2cir100.png'

%read the bmp file
fbuf = imread( inputfile );
%255represents while, which is the pixel of the picture that we are
% aiming to obtain.
[Y,X] = find( fbuf==255 );
% for calculating the center point of the figure,
%    and the coordinate is approximately rounding.
ccxy = [floor(0.5+(min(X)+max(X))/2), floor(0.5+(min(Y)+max(Y))/2)];
X = X - ccxy(1);
Y = ccxy(2) - Y;

%for calculating each coordinate of every point in the circle
dotnum = length(X);
r = ceil( (dotnum/pi)^(1/2) );
xx = reshape( ones(2*r+1,1)*[-r:r] ,[(2*r+1)^2,1] );
yy = reshape( [-r:r]'*ones(1,2*r+1) ,[(2*r+1)^2,1] );
D = xx.^2 + yy.^2;
[ord, ind]=sort(D,'ascend');
%arrange the points regarded to the distance from the center of the circle.
xy(:,1) = xx( ind(dotnum:-1:1) );
xy(:,2) = yy( ind(dotnum:-1:1) );

sxy = xy;
D = X.^2 + Y.^2;
[ord, ind]=sort(D,'descend');
%arrange the pixel points of the figure by from the nearest
% to the center to the farthest.
dxy = [X(ind(:)), Y(ind(:))];

dcmpnum = 32;
for i = 1 : dotnum
num = size(sxy,1);
```

```
%optimized algorithm: the farthest loop of the original figure,
%8*r represents one loop of the points
send = min( num , 8*r );
%optimized algorithm: fetch out 32 farthest points,
%find the maxium of the distance in a loop.
        dend = min( num , dcmpnum );
clear Dds;
%calculate the square of the distance of points in the far layer of
%the figure that aiming to the far layer points of the original figure.
      for j = 1 : dend
            Dds(:,j)=(sxy(1:send,1)-dxy(j,1)).^2 + (sxy(1:send,2)-dxy(j,2)).^2;
End
% pick the smallest distance in each line.
minDds = min(Dds);
% pick the max of the smallest distance of each line in the matrix.
tempd = max ( minDds );
dn = find(minDds==tempd);
% find one specific point’s “coordinate” in the matrix,
% and then select the first one
      sn = find( Dds(:,dn(1))==tempd );
%transfer the data reserve, for the following sequence:
% X-axis of the original figure, Y-axis of the original figure,
% X-axis of the figure that aimed, Y-axis of the figure that aimed,
%the distance, the time of delaying.
Trans(i,:) = [ sxy(sn(1),1), sxy(sn(1),2), dxy(dn(1),1), …
  dxy(dn(1),2), tempd^(1/2), 0 ];
% delete the fit points in the original figure.
sxy( sn(1), : ) = [];
% delete the fit points in the figure that aimed.
      dxy( dn(1), : ) = [];
end

maxd = Trans(1,5) + Trans(1,6);
%adjust the delaying time of setting off, so that every point can
% be able to reach their correspond point in the sky simultaneously.
Trans(:,6) = Trans(:,6) + Trans(1,5) - Trans(:,5);
minTrans6 = min(Trans(:,6));
if minTrans6 < 0
      Trans(:,6) = Trans(:,6) - minTrans6;
end;
%optimized algorithm: if the distance is longer than the previous time,
% adjusting the delaying time of this turn means adjusting the
% waiting time for setting off.
for i = dotnum : -1 : 2
      if Trans(i-1,5) < Trans(i,5)
            Trans(i,6) =    Trans(i-1,6);
      end
end

%calculate the route of movement,
% and then save the file to ‘savefile + serial number + .png’
for i = 0 : ceil(maxd)
      figure(1);
      axis([min(X)-2 max(X)+2 min(Y)-2 max(Y)+2]);
```

```
        axis equal;
        hold on;
        dd = min(i,maxd);
        mx = Trans(:,1) + (Trans(:,3)-Trans(:,1)).*(Trans(:,6)<=i) ...
  .* (dd-Trans(:,6))./(maxd-Trans(:,6));
my = Trans(:,2) + (Trans(:,4)-Trans(:,2)).*(Trans(:,6)<=i) ...
  .* (dd-Trans(:,6))./(maxd-Trans(:,6));
        plot( mx, my, 'k.');
        saveas(1,[savefile num2str(i) '.png'],'png');
        close;
end
```