The 110th academic year of the Department of Information Engineering, Tamkang University

Special Experimental Results Report

Topic name:

Imitation Game – Facial Expression Similarity Challenge

Team:

Leader: Ruyi Xu

Xiehan Chen

Chengqian　Li

Bohan Lin

Zhuoying Zhou

Jingxiong Wang

Table of contents

Motivation and purpose

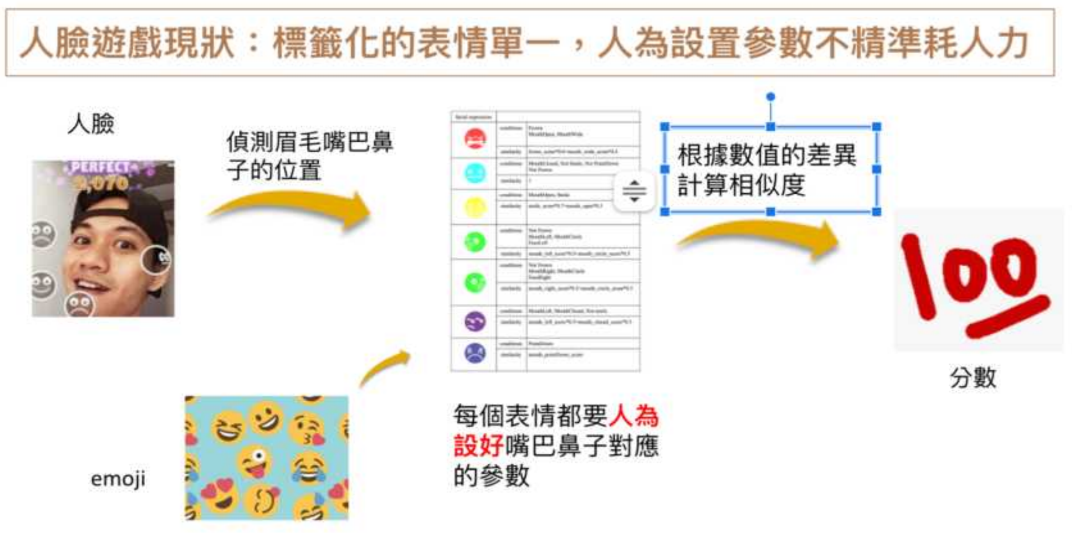
1. Research Background

Artificial intelligence, as a technology born in the 1950s, has ushered in its second spring due to the explosion of data and the continuous maturity of the underlying technology in recent years, and has gradually begun to show its strong potential in various fields. As a computer-related technology, its ability to solve problems and its wide range of applications are all amazing. Today, it exists in the medical industry, entertainment industry, and financial industry. It is also effective in the field of Go, which is equally mysterious in the eyes of the public. It can also make achievements in the field of debate that people think AI cannot touch.

Image processing, as an important branch of artificial intelligence, has important value in the research field and our life. With the continuous maturity of corresponding technologies, a large number of different and unique AI derivative applications and technologies are also emerging. And this powerful tool is naturally used by curious people to explore the human body itself. Among them, as the most varied part of the human body and one of the important ways to express human emotions, the human face and its related facial expressions have naturally received high attention. With the efforts of all parties to explore and research, the technology of face recognition and emotion recognition has become more and more mature, resulting in a variety of different kits and technical achievements on the market. Not only that, with the continuous maturity and improvement of basic face recognition technology, related application products are also emerging, such as the common expression game on the market is one of them. It's a pity that most of the expression games on the market are rigid matches for a few fixed expressions, and the application of face recognition in more significant medical applications is not abundant, so further research and attempts in related fields It is also meaningful and interesting.

2. Research motivation

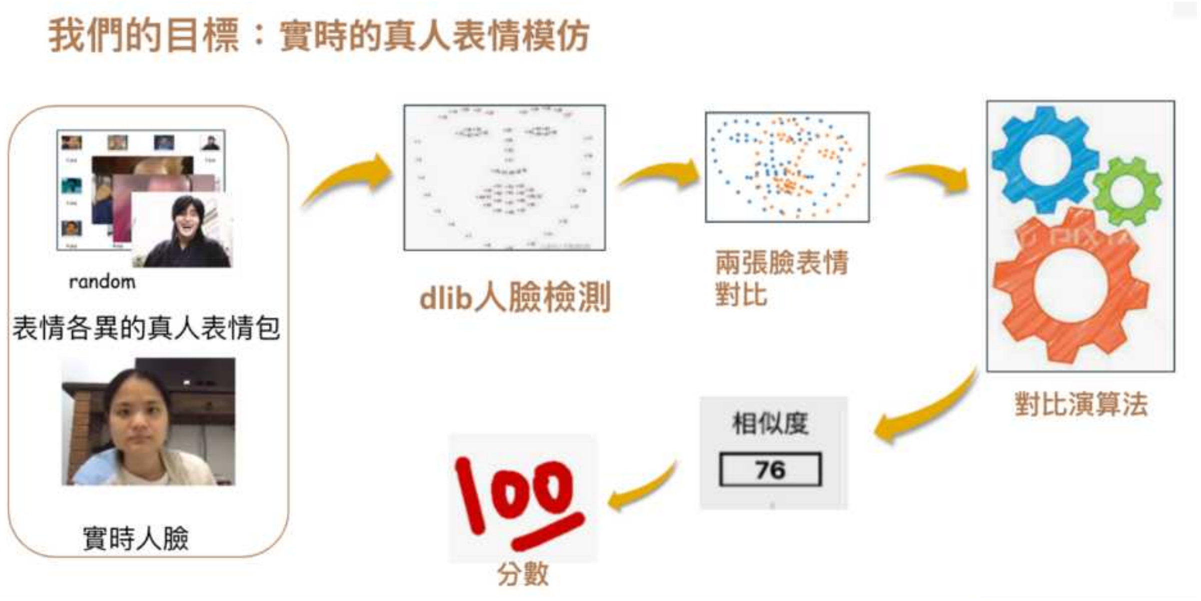
After a certain period of time to understand and study the basic knowledge of AI, we have the idea of developing and researching AI-related applications, among which face recognition and emotion recognition first entered our field of vision. After doing a lot of data searching, we found it quite interesting to make a small game of facial expression recognition and similarity matching.


人臉遊戲現狀：標籤化的表情單一，人為設置參數不精準耗人力
人臉
偵測眉毛嘴巴鼻子的位置
根據數值的差異計算相似度
100
分數
emoji
每個表情都要人為設好嘴巴鼻子對應的參數

We found that the face matching games on the market mostly match a few fixed facial expressions, and most of them are played by one player, which seems too boring. So we thought of researching an expression game that can break away from the fixed expression template, and develop a two-player mode. In this way, its playability and fun can be greatly improved, and it can be more attractive to players, so that patients can have more motivation to persist and get better positive feedback by playing with patients or doctors.

3. Research purpose

Make a game based on facial expression similarity matching, and hope to use our game to help people with facial paralysis and people with poor emotion recognition. Through the game, let them understand what the expressions of the general public represent in an easier way Meaning, by the way, they can use the scores to let them know whether they have made progress while playing the game to achieve the effect of rehabilitation



Tool

1. OpenCV :
   The full name of OpenCV is Open Source Computer Vision Library , which is a cross-platform computer vision library that can be used to develop real-time image processing and pattern recognition programs. This project uses opencv to open the camera, capture the camera screen, process pictures and other image processing functions.

2. Dlib :
   Dlib is a set of libraries including machine learning, computer vision, and image processing. This project applies dlib 的 the face recognition module, and uses the face 68 feature point model (shape_predictor_68_face_landmarks.dat.bz2) officially trained

by dlib to detect face feature points.

3. PyQt5、Qt Designer
   PyQt is a GUI programming solution for the Python language , and QtDesigner is a visual UI designer included in Qt , which uses drag and drop operations to design graphical interfaces. While designing, you can also directly preview the final form effect. Qt Designer can save a lot of code when the form is complex or the whole program needs a lot of forms . This project uses QtDesigner assist in designing UI and generating python program code.

4. The numpy data structure, this project refers to numpy to generate matrices and transform matrices.
5. SciPy math tool, this project refers to the built-in function of this tool to calculate Euclidean distance and cosine distance.

Functional description

It is a small game of facial expression recognition. We combine facial expression recognition with games to make a small game that can exercise facial muscles.

The game is divided into single-player mode and double-player mode:
in single-player mode, players can challenge three levels in total, and make corresponding expressions according to the expression pictures generated by the system, and then the system will make corresponding expressions according to the similarity of the expressions made by players. Make a score.

In the two-player mode, players can choose who imitates whose expression, the time is 15s, and the final picture will be scored according to the similarity.

The program features we need to address include:

1. Make a question – then take out an emoticon package from the library

2. Image capture – opencv captures picture frames

3. Obtain and draw face feature points

4. Calculate the similarity

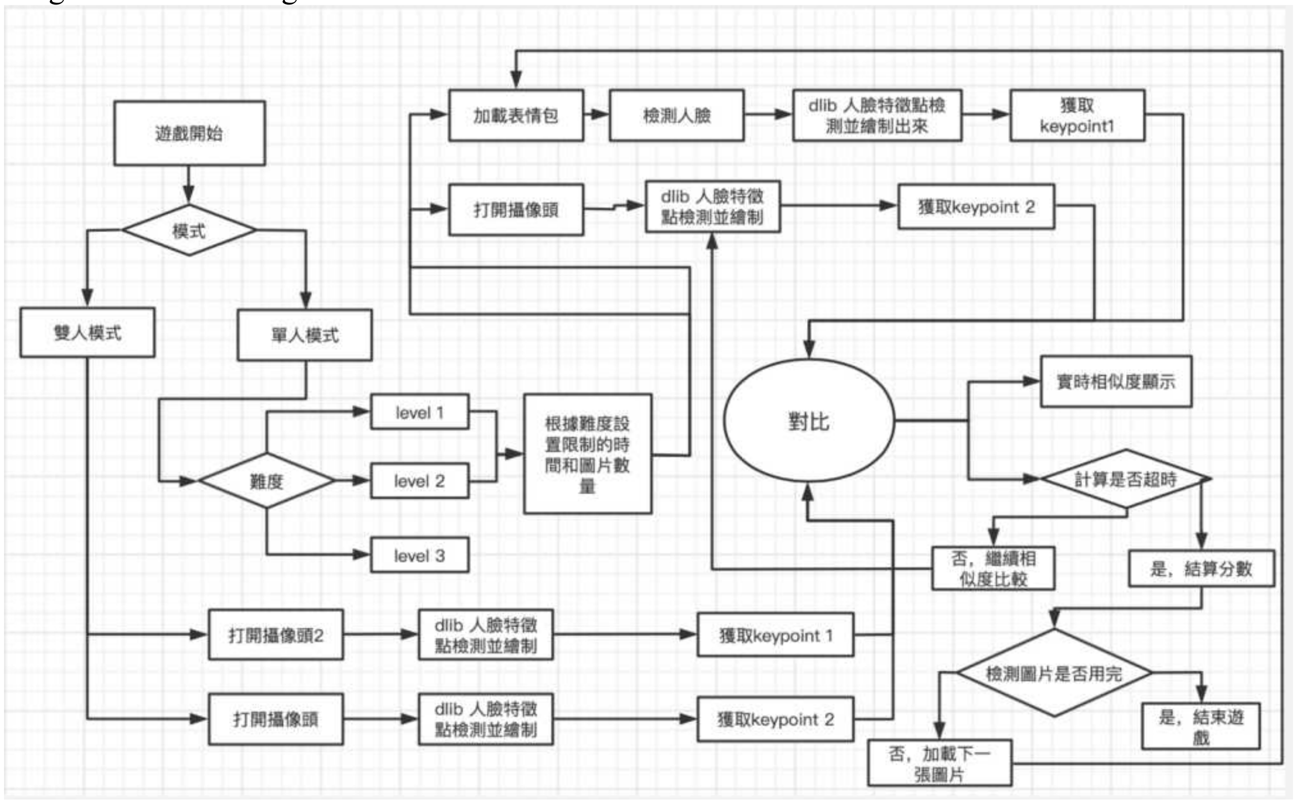
5. Set the countdown and the number of pictures according to the difficulty

6. Judge the countdown time and calculate the score

7. The player selects the game mode, pauses/starts the music, returns to the selection page, ends the game, – pyqt5 builds button events

Architecture diagram

Program structure diagram:



Experimental stage

The first stage: data collection, determine the topic, determine the gameplay

At this stage, we determined the direction of our research - expression recognition.

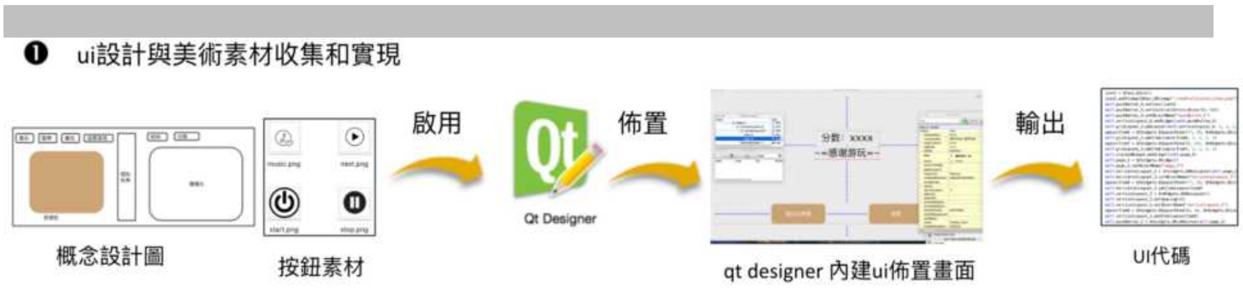
For this general direction, we first learned about some applications and technologies used in facial expression recognition on the market. Basically, the features of the face are obtained through opencv technology and its suite. After a general understanding and learning of the usage of opencv, we determined our topic through discussion within the group.

According to the title of the topic, we also learned about the small games related to emoticons that can be found on the Internet. Collected and summarized the gameplay of this type of game, and finally discussed the gameplay of our game and what the core mechanism of the game is. Then we began to divide and cooperate. Some people planned the gameplay and game values; some collected the emoticons needed for the game and built an emoticon library;

The second stage: design game system framework and core algorithm research

1. Write the game framework

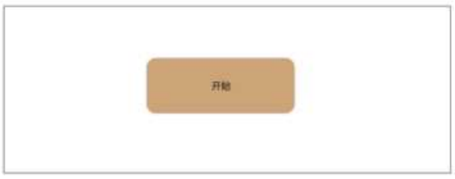
1) UI interface

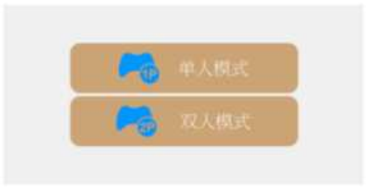


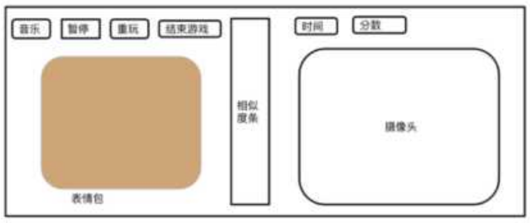
Writing UI interfaces in PyQt can be implemented directly through code, or through Qt Designer. The design of Qt Designer conforms to the MVC architecture, which realizes the separation of view and logic, thus realizing the convenience of development. The operation mode in Qt Designer is very flexible. You can view the effect of the control at any time by placing the control by dragging and dropping. The .ui file (essentially an XML format file) generated by Qt Designer can also be converted into a .py file by the pyuic5 tool.

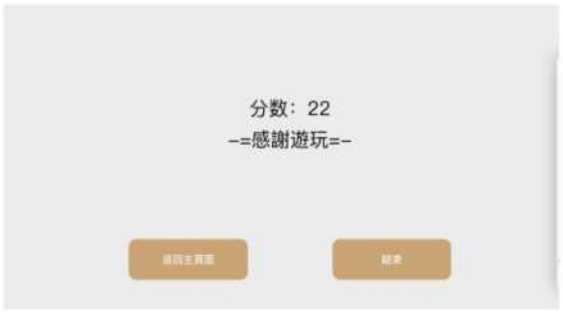
QtDesigner is installed together with the PyQt5-tools package, and its installation path is under "Python installation path\Lib\site-packages\pyqt5-tools".

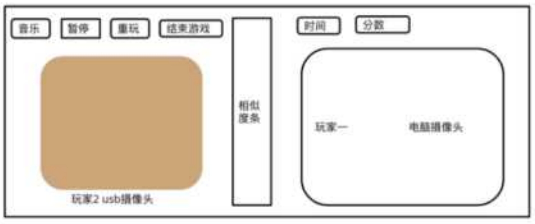
UI concept design:



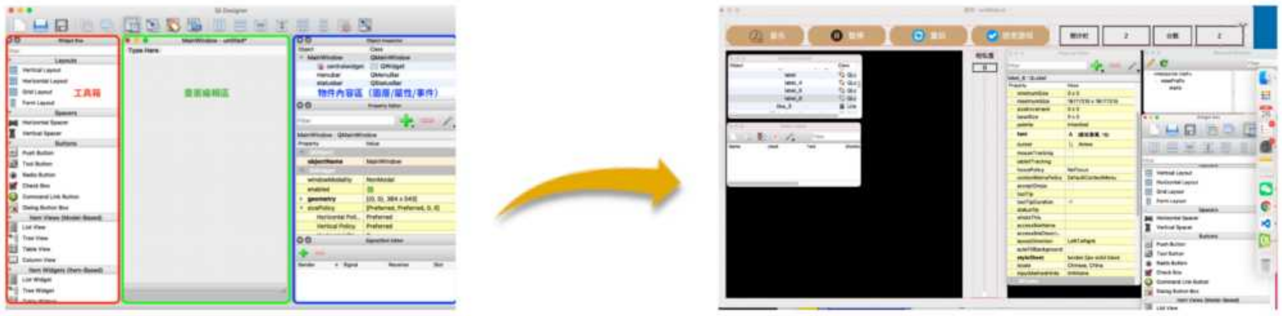




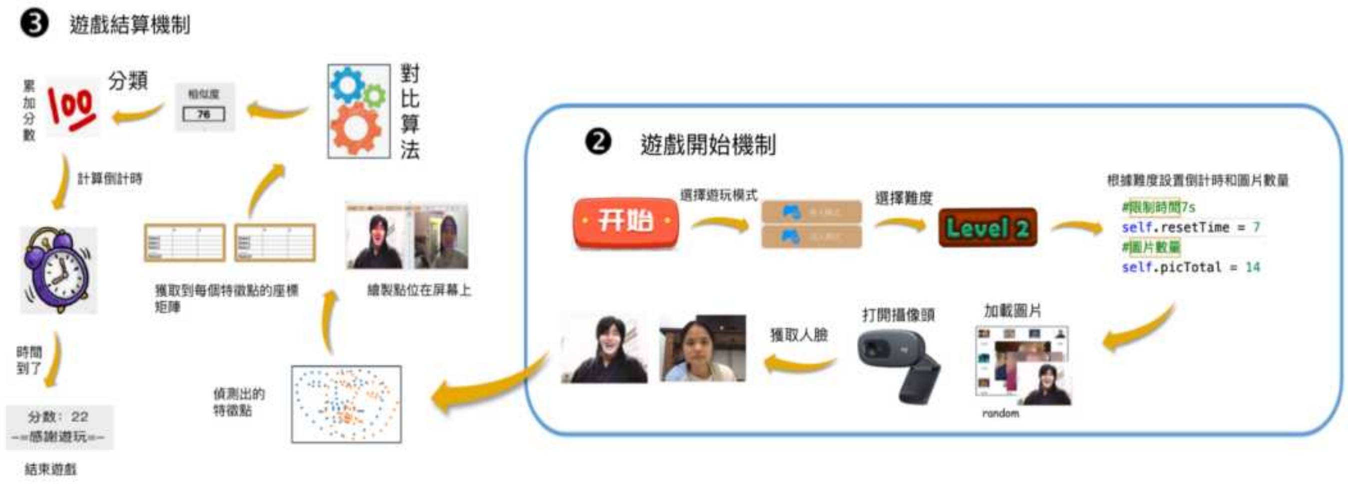




Qt designer operation
1. Toolbox: There are various objects available (containers, input and output windows, interactive buttons, etc.)
2. Screen editing area: the designed screen
3. Object content area: Contains object layer tree view, object property window, and object event editing window

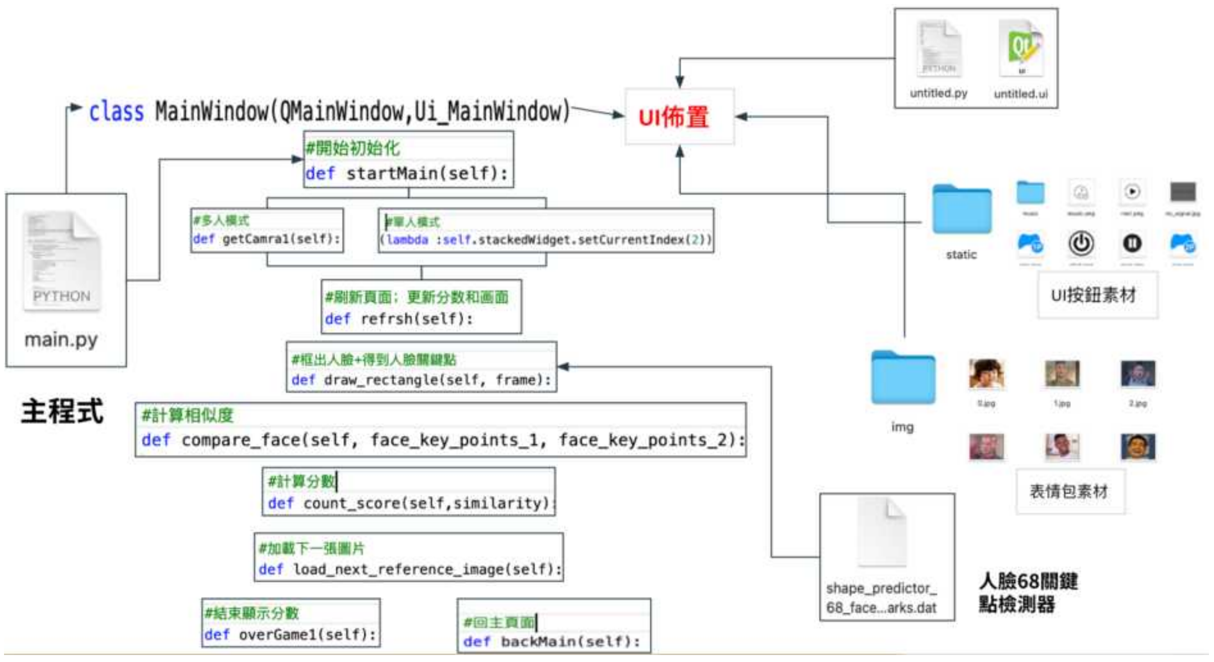


2) Game mechanism design:



```
#限制時間7s
self.resetTime = 7
#圖片數量
self.picTotal = 14
```

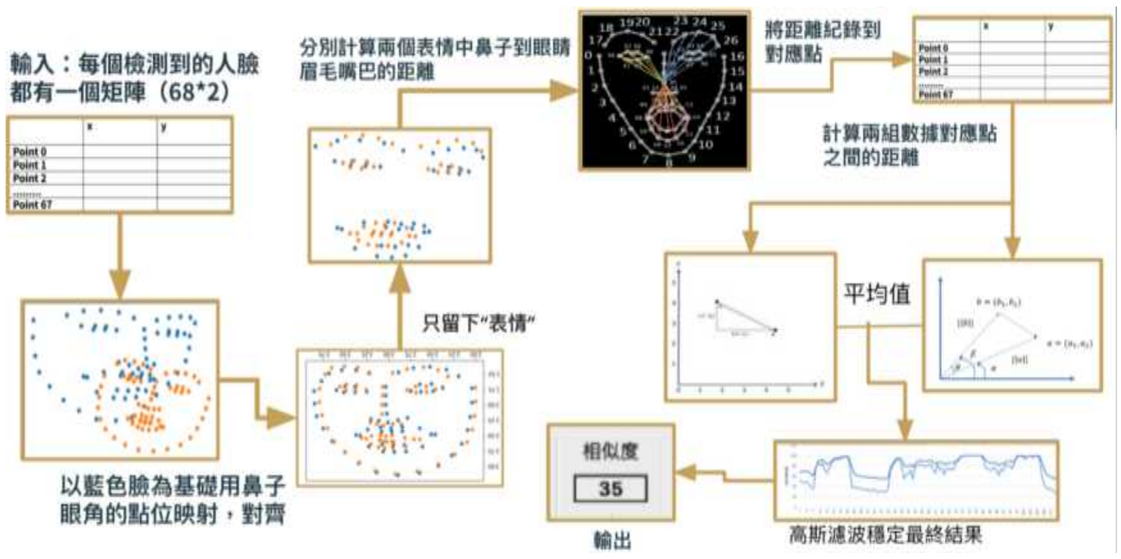
Functional Architecture Diagram of Game Mechanism:



```
class MainWindow(QMainWindow,Ui_MainWindow)
#開始初始化
def startMain(self):
#多人模式
def getCamra1(self):
#單人模式
(lambda :self.stackedWidget.setCurrentIndex(2))
#刷新頁面；更新分數和畫面
def refrsh(self):
#框出人臉+得到人臉關鍵點
def draw_rectangle(self, frame):
#計算相似度
def compare_face(self, face_key_points_1, face_key_points_2):
#計算分數
def count_score(self,similarity):
#加載下一張圖片
def load_next_reference_image(self):
#結束顯示分數
def overGame1(self):
#回主頁面
def backMain(self):
```

2. Expression similarity algorithm idea :

Idea map:


輸入：每個檢測到的人臉都有一個矩陣（68*2）
Point 0
Point 1
Point 2
Point 67
以藍色臉為基礎用鼻子眼角的點位映射，對齊
只留下"表情"
分別計算兩個表情中鼻子到眼睛眉毛嘴巴的距離
將距離紀錄到對應點
計算兩組數據對應點之間的距離
平均值
高斯濾波穩定最終結果
相似度
35
輸出

Comparison steps of facial expressions

1.dlib gets 68 key points

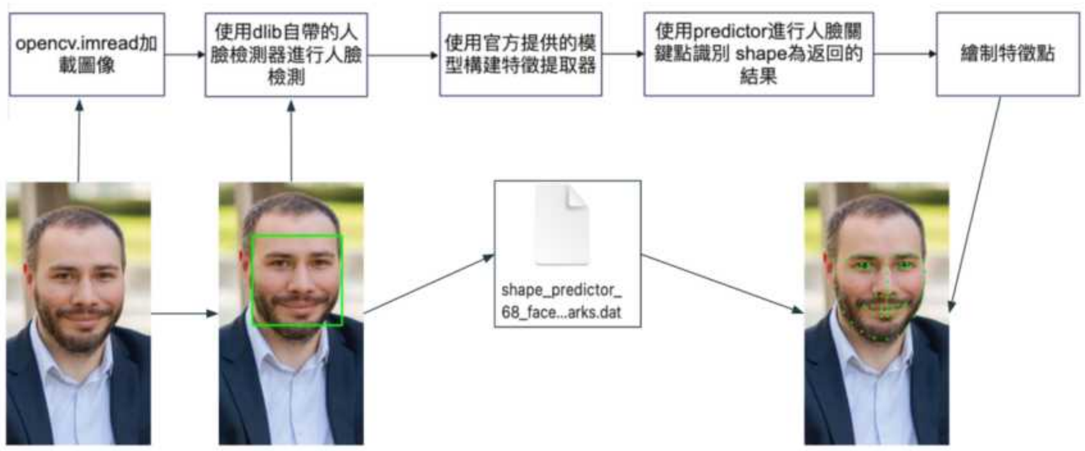
2. Align (affine transformation), so that the two faces have the same angle, the same ratio, and the same position

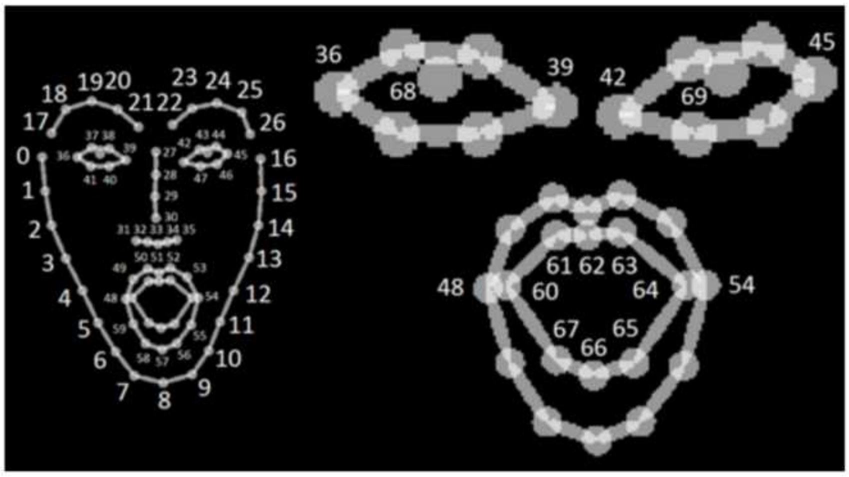
3. Construct a vector matrix, that is, construct a distance matrix showing the distance from the facial features to the tip of the nose with the tip of the nose as the center

4. Use Euclidean distance and cosine distance to get the similarity of data

5. Stabilize the result with Gaussian filtering

Step 1: extraction of facial key points: Dlib


opencv.imread加載圖像
使用dlib自帶的人臉檢測器進行人臉檢測
使用官方提供的模型構建特徵提取器
使用predictor進行人臉關鍵點識別 shape為返回的結果
繪制特徵點
shape_predictor_68_face...arks.dat



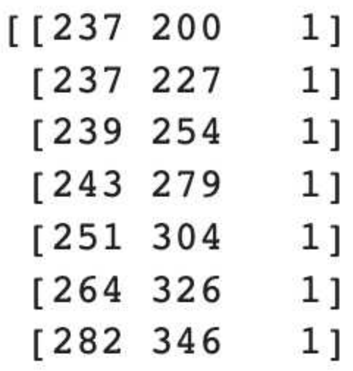
dlib68 face feature data output

The system provides us with results in a matrix. The matrix structure is as follows:

#Add a line to facilitate matrix transformation calculation

Each detected face has a matrix (68*3)

	x	y	The probability of x, y on the screen
Point 0			1
Point 1			1
Point 2			1
………			………
Point 67			1



```
[[237 200   1]
 [237 227   1]
 [239 254   1]
 [243 279   1]
 [251 304   1]
 [264 326   1]
 [282 346   1]
```

Step 2: Alignment (affine transformation)

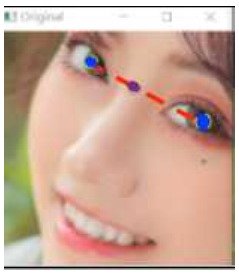
In this step, we save a matrix of faces detected in reference images and webcam frames to calculate facial expression distances.
The goals of this step are to:
1. Eliminate the influence of natural face differences (such as fat face and thin face) and the distance between the player and the camera
2. Align the two faces to the same angle, the same ratio, and the same position.

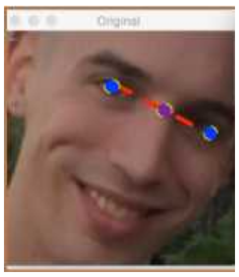
Thinking:

Method 1: opencv three-point method

Generally, face alignment is to straighten the face to the standard face, so the original opencv uses cv2.getRotationMatrix2D to perform affine transformation on the face and the reference coordinates. The built-in affine transformation of opencv only needs three points (left eye, right eye, center of left and right eye):

The distance between the left and right eyes determines the scale and angle
The left and right eye center points determine the rotation center point



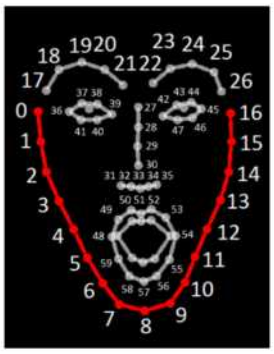


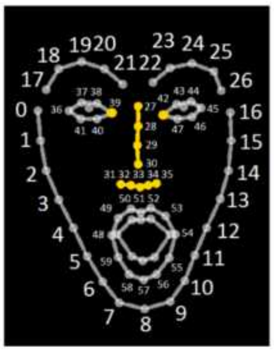
The resulting problem:
1. The distance between the eyes of each person is different, and the zoom ratio calculated based on this is not accurate enough
2. The position of the center of rotation calculated by the side face and the front face is different, which will make the alignment of the eyebrows and mouth deviate

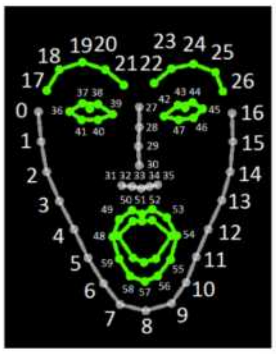
Solve the problem:

Method 2: Multi-point alignment

1. Point grouping


Groupe 1


Groupe 2


Groupe 3

The first group: irrelevant points (0~16)
These points are not directly related to facial expressions or facial positions. On the other hand, they are related to the natural shape of the face.

Group Two: Anchor Points
[27,28,29,30 (nose nose tip)
31,32,33,34,35 (the lower edge of the nose)
36,39,42, (the corners of the left and right eyes)]
These points are used to position/rotate the face. We will use them in the alignment step.

The third group: feature points (17~26, 36~59)
These points are related to facial expressions, and we can recognize easy expressions based on these points.

2. Calculate the transformation matrix:

To make our demo algorithm robust (Robustness) to scaling, transformation and rotation, we will construct an affine transformation matrix (T) to align the two faces:

$$\begin{bmatrix}\vec{y}\\ 1\end{bmatrix} = \begin{bmatrix}A & \vec{b}\\ 0,\ldots,0 & 1\end{bmatrix}\begin{bmatrix}\vec{x}\\ 1\end{bmatrix}$$

After transformation, both faces will have the same position, same scale and same rotation:

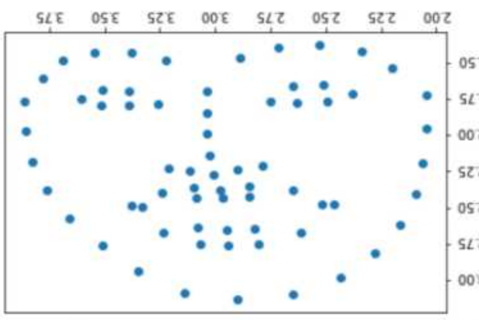
Similarity transformation is to use rotation, translation, and proportional scaling to make the second picture fit the first picture as much as possible.

A. Reduce the first image first, and the reduction ratio is 1/(Euclidean distance from nose (30) to chin (8)).
Purpose:
1) Reduce the face to a unit distance

2) Reduce the individual differences between different faces to a certain extent
3) Slow down the problem of numerical explosion in similarity calculation

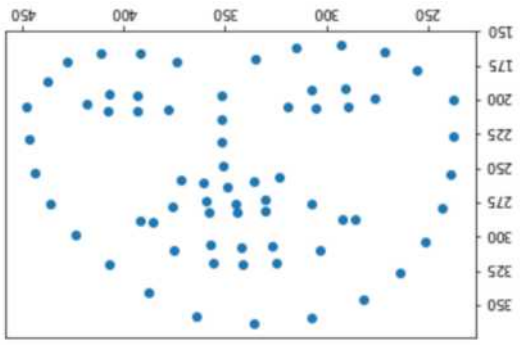
縮放矩陣

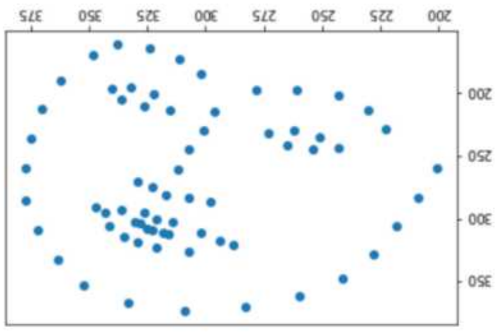
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x\, x \\ s_y\, y \\ s_z\, z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



```
# alignement_1 對齊1
#縮放
a = 1/(distance.euclidean(face_key_points_1[30, :2], face_key_points_1[8, :2]))
T_1 = np.array([[a,0,0],[0,a,0],[0,0,1]])
face_key_points_1_transformed = np.transpose(np.dot(T_1, np.transpose(face_key_points_1)))
```

B. Define the Q matrix as a matrix composed of coordinate points of the input image, S is a matrix composed of coordinate points of the target face, and M is the affine matrix that maps the input to the target face





2

S的關鍵點　　　　　　Q的關鍵點

C. Then there is $Q' = QM$,

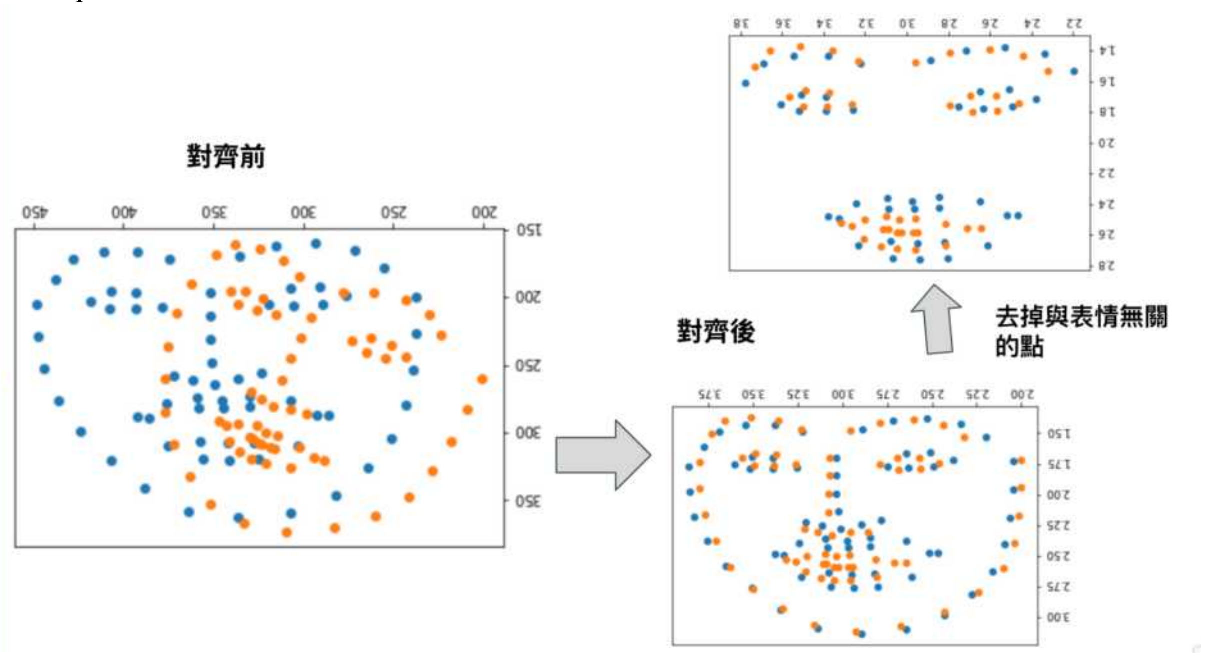
And what we need to minimize is $minimize \sum (Q' - S)^2$, that is, when M takes what value, the mapped Q' can be closest to S

- Right now $minimize \sum (QM - S)^2$

- By the method of least squares we get $M = (Q^T Q)^{-1} Q^T S$

```
M = np.linalg.inv(Q.T @ Q) @ Q.T @ S
```

D. Experimental results:

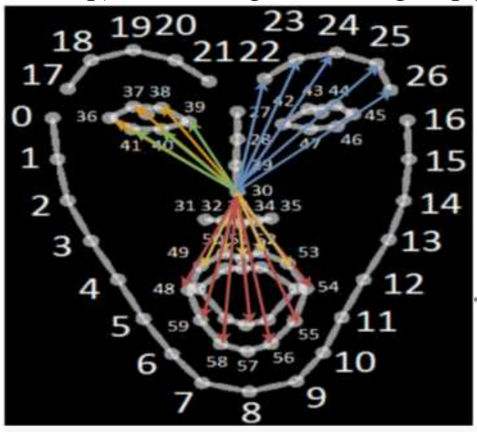


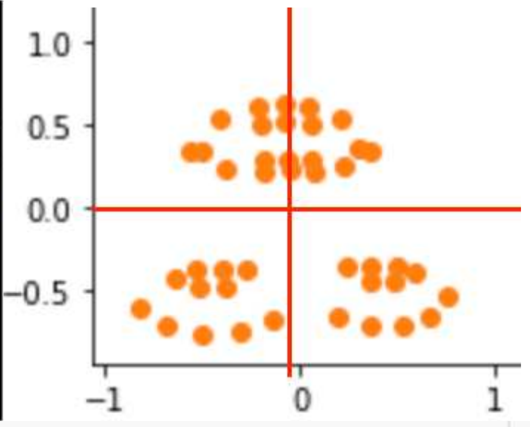
Step 3: Compare (distance calculation)

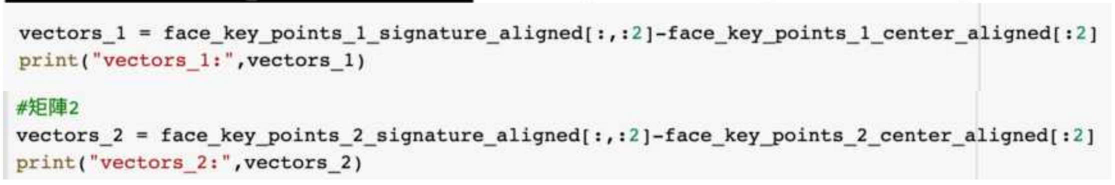
The purpose of this step: Calculate the position difference between the feature points of the two faces and convert it into similarity

1. Build a vector matrix

For each face, we'll build a signature matrix with vectors starting at the reference point (30: nose tip) and ending at all 3rd group points.



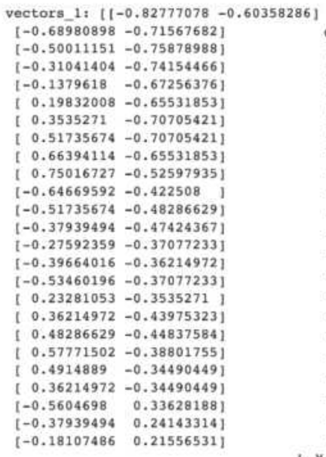




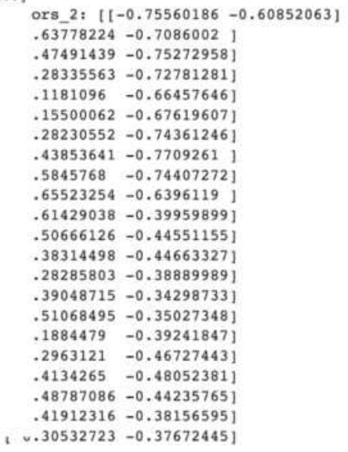
```
vectors_1 = face_key_points_1_signature_aligned[:,:2]-face_key_points_1_center_aligned[:2]
print("vectors_1:",vectors_1)

#矩陣2
vectors_2 = face_key_points_2_signature_aligned[:,:2]-face_key_points_2_center_aligned[:2]
print("vectors_2:",vectors_2)
```

Output vector matrix:

	x	y
Point 17		
Point 18		
Point 19		
………		
Point 59		



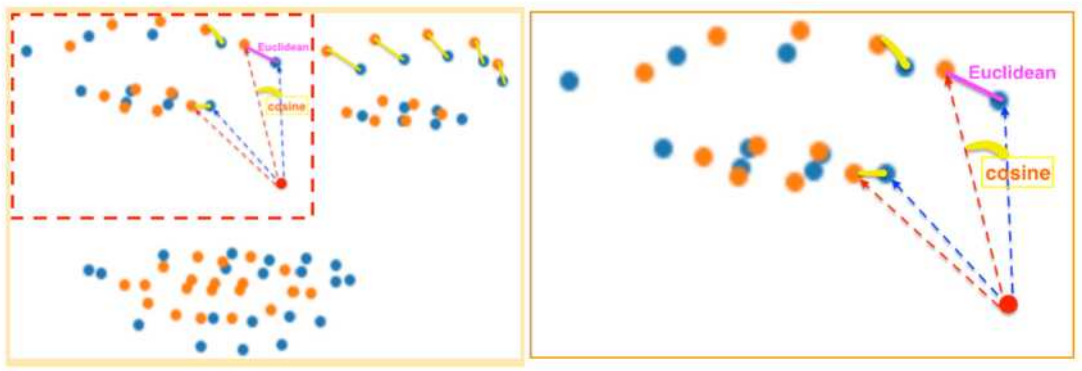
```
vectors_1: [[-0.82777078 -0.60358286]
 [-0.68980898 -0.71567682]
 [-0.50011151 -0.75878988]
 [-0.31041404 -0.74154466]
 [-0.1379618  -0.67256376]
 [ 0.19832008 -0.65531853]
 [ 0.3535271  -0.70705421]
 [ 0.51735674 -0.70705421]
 [ 0.66394114 -0.65531853]
 [ 0.75016727 -0.52597935]
 [-0.64669592 -0.422508  ]
 [-0.51735674 -0.48286629]
 [-0.37939494 -0.47424367]
 [-0.27592359 -0.37077233]
 [-0.39664016 -0.36214972]
 [-0.53460196 -0.37077233]
 [ 0.23281053 -0.3535271 ]
 [ 0.36214972 -0.43975323]
 [ 0.48286629 -0.44837584]
 [ 0.57771502 -0.38801755]
 [ 0.4914889  -0.34490449]
 [ 0.36214972 -0.34490449]
 [-0.5604698   0.33628188]
 [-0.37939494  0.24143314]
 [-0.18107486  0.21556531]
```



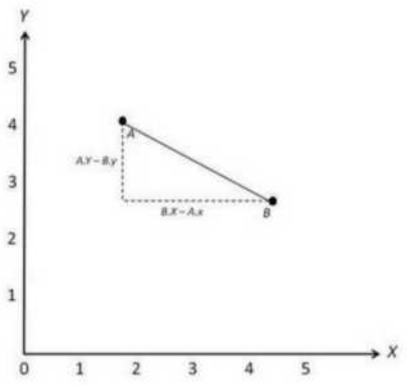
```
ors_2: [[-0.75560186 -0.60852063]
.63778224 -0.7086002 ]
.47491439 -0.75272958]
.28335563 -0.72781281]
.1181096  -0.66457646]
.15500062 -0.67619607]
.28230552 -0.74361246]
.43853641 -0.7709261 ]
.5845768  -0.74407272]
.65523254 -0.6396119 ]
.61429038 -0.39959899]
.50666126 -0.44551155]
.38314498 -0.44663327]
.28285803 -0.38889989]
.39048715 -0.34298733]
.51068495 -0.35027348]
.1884479  -0.39241847]
.2963121  -0.46727443]
.4134265  -0.48052381]
.48787086 -0.44235765]
.41912316 -0.38156595]
.30532723 -0.37672445]
```

2. Calculate the Euclidean and cosine distances between these 2 vector matrices

The expression difference map of the two faces:



A. Euclidean distance:



$$d(x,y)=\sqrt{\left(\sum (x_i-y_i)^2\right)}$$

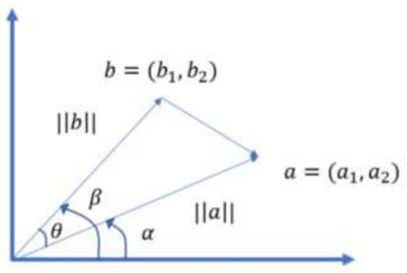
$$sim(x,y)=\frac{1}{1+d(x,y)}$$

Taking the distance from the tip of the nose as the coordinate axis, draw the two vector matrix data on the coordinate system, and calculate the straight-line distance between them.

```
# squared Euclidean distance 平方欧几里得距离
squared_euclidean_distance = 0
for i in range(len(signature_indexes)):
  squared_euclidean_distance += (distance.euclidean(vectors_1[i, :], vectors_2[i, :])
```

B. Cosine distance calculation: distance = 1- cosθ

Cosine similarity, as the name implies, measures the similarity between two vectors by measuring the cosine of the angle between them.
The smaller the angle between the vectors, the closer the directions of the two vectors are



$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}.$$

```
cos_distance = 0
for i in range(len(signature_indexes)):
    cos_distance += distance.cosine(vectors_1[i, :], vectors_2[i, :])
```

The biggest difference between cosine similarity and Euclidean distance is that the measurement of cosine similarity will not be affected by the size of the vector, because the calculation will be divided by the size of the vector itself, similar to a standardized action. Since our calculation must consider both the actual distance and the similarity of the angle, we finally use the result of adding and averaging the two distances

3. Convert distance to similarity

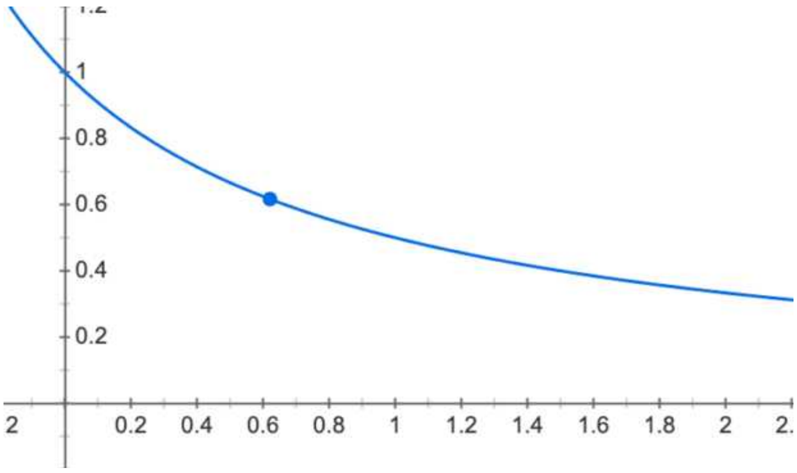
1) Euclidean distance:

To lock the similarity between 0-1:

Initial setting method: similarity = 1/1+d

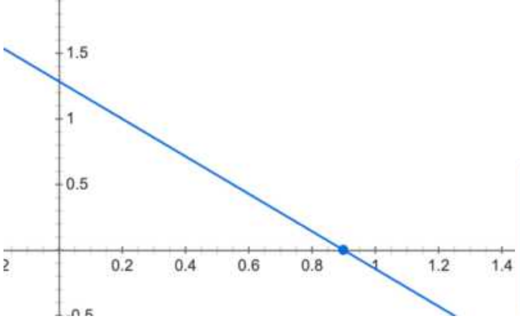
Problem: This method is very simple and crude, but we found the following problems:

1. After the distance increases, the change of the similarity is too small, and the result is not intuitive enough for the player, so it is more intuitive to use the linear function
2. After the distance is very small, x can only have a high similarity when the distance is close to 0. Considering the difference between r faces, we need to set a tolerance value
3. The characteristics of this function make our final result easy to hover in the middle value (that is, between 60% and 40%)



Solve the problem:

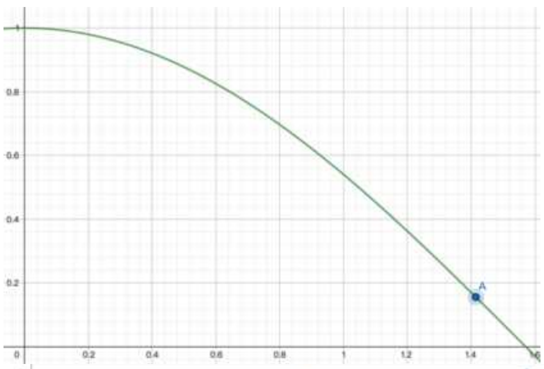
We customize two distances:
When d<0.2, the similarity is set to 1
When d>0.9, the similarity is 0

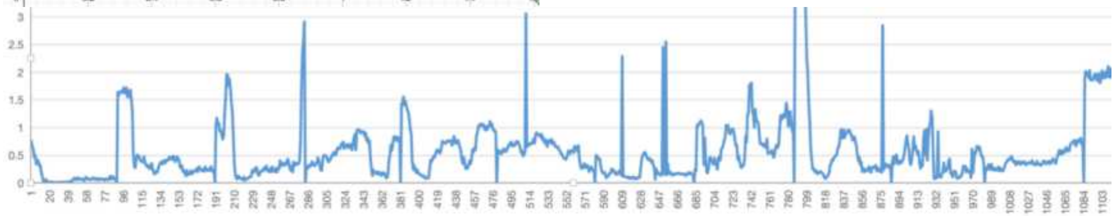


if d<0.2 similraty = 1
if d>0.9 similraty = 0
if 0.2<= d <=0.9 (0.9– d) / (0.9–0.2)

2) Cosine distance:

It has a similar problem with the Euclidean distance. It is that x can only have a high similarity when it is close to 0 distance, and it is equal to 0 when the distance is large.





Through the experimental data, it is found that there is a gap of about 0.2-0.3 between different faces, so d<0.2, the similarity is set to 1

The distance of the cosine distance approaching zero is relatively reasonable. The experimental data shows that there is a big gap if it exceeds 1.5, so when d>1.5, the similarity is equal to 0

4. Smoothing of results

problem found:
Although the result of the similarity is very accurate, the transformation is too fast. Each calculation of the similarity is displayed according to the number of video frames. In addition, the performance of facial feature points and the high-speed image of the camera make the result mixed with noise. resulting in very unstable results.

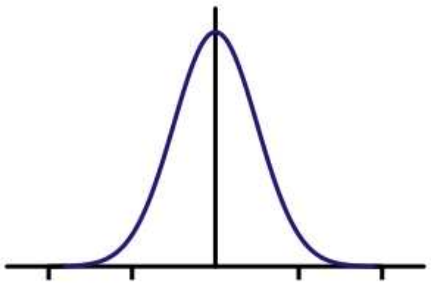
Solve the problem:
In the field of image processing, there are many different types of filters. We need to smooth the results and filter out noise, so there are
A. Mean filter
B. Median filter
C. Gaussian Filter
Finally, the result of using Gaussian filtering is the most stable:

1) Sampling
For sampling we use the nearest three similarities:

```
self_last_similarity_0 = 0 // this time similarity
self_last_similarity_1 = 0 // last similarity
self_last_similarity_2 = 0 // last similarity
```

2) Calculate the weight
Then, the calculation method of the Gaussian equation in Gaussian filtering is used to calculate the proportion that each sampling should occupy in the Gaussian distribution (normal distribution) curve.

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/(2\sigma^2)}$$



In this basic Gaussian equation in the one-dimensional case, we set the deviation to 1 to facilitate calculations. And the time when the similarity is generated is used as the basis for its weight distribution, so the meaning of the x-axis is the time difference between each similarity and the similarity calculated this time.

Specifically:

X0 = 0 (this time, in the center of Gaussian distribution)

X1 = -1 (last time)

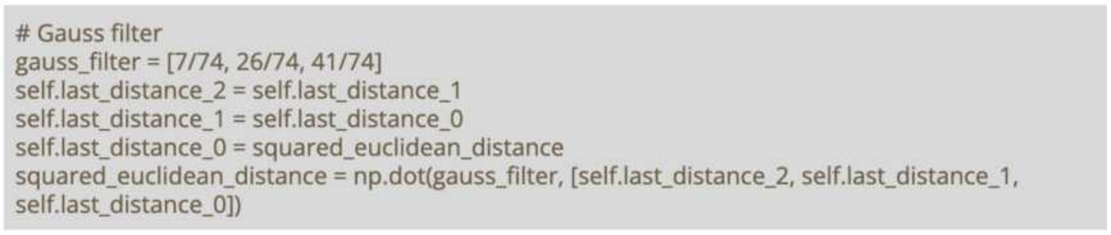
X2 = -2 (last time)

In this way, the corresponding values are summed up after being brought into the formula and calculated, and used as the denominator to calculate the weight of each corresponding similarity , so as to achieve the effect that the sum of the weights is 1.

After the calculation result is converted into a matrix, it is as follows:

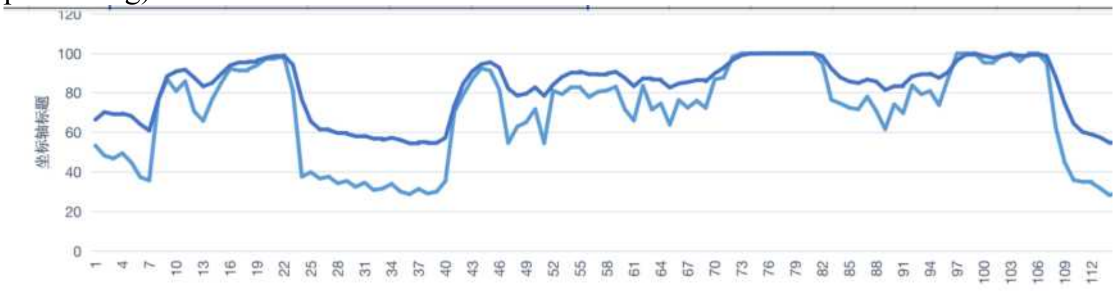
gauss_filter = [7/74, 26/74, 41/74]

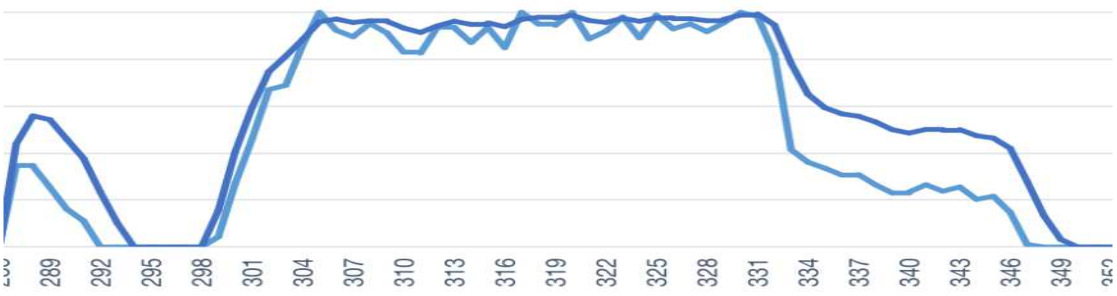
to do point multiplication with the newly formed similarity matrix every time a new similarity is calculated, so as to reduce the noise interference caused by extreme situations.



```
# Gauss filter
gauss_filter = [7/74, 26/74, 41/74]
self.last_distance_2 = self.last_distance_1
self.last_distance_1 = self.last_distance_0
self.last_distance_0 = squared_euclidean_distance
squared_euclidean_distance = np.dot(gauss_filter, [self.last_distance_2, self.last_distance_1,
self.last_distance_0])
```

The following is a comparison chart before and after Gaussian filtering (light blue is before processing)



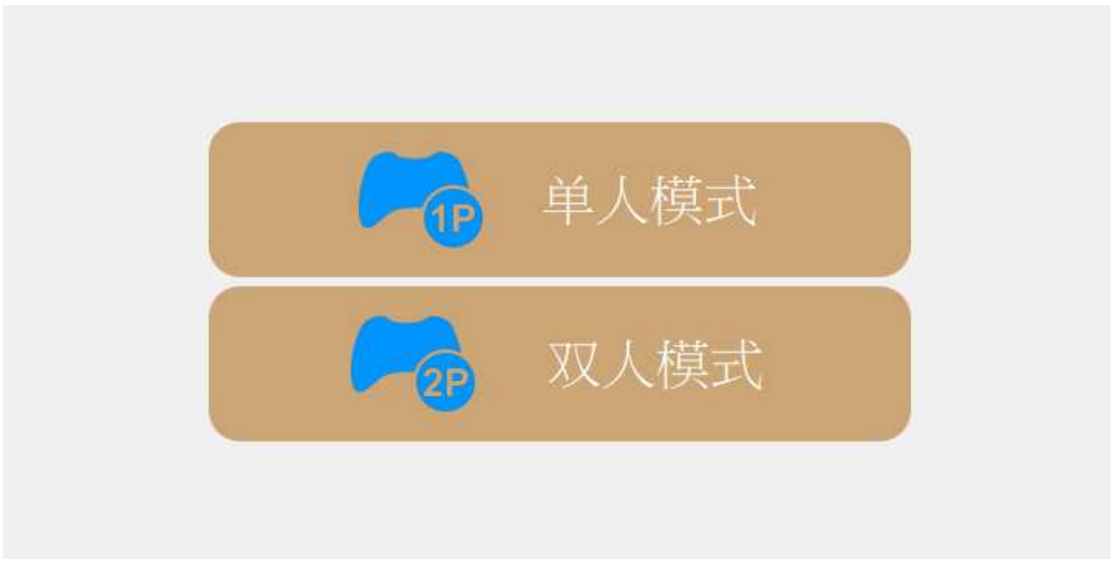


The third stage: presentation of game content

Isnterface display

1. Start interface:

2. Select mode: You can choose single mode or double mode



3. Difficulty of level selection in single player mode:

Game level selection interface:

Level difficulty can be selected here

level 1: 10 seconds per picture, 10 pictures in total.

level 2: 7 seconds per picture, 14 pictures in total.

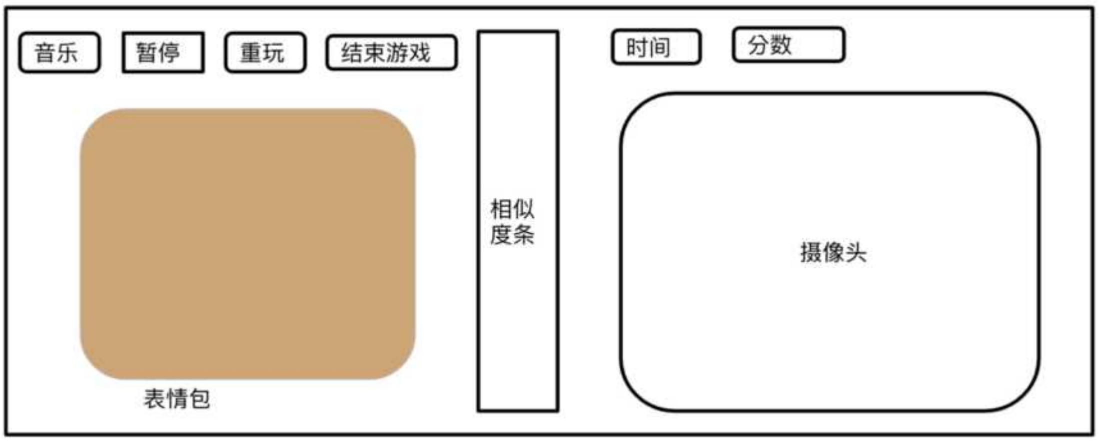
level 3: 4 seconds per picture, 25 pictures in total.

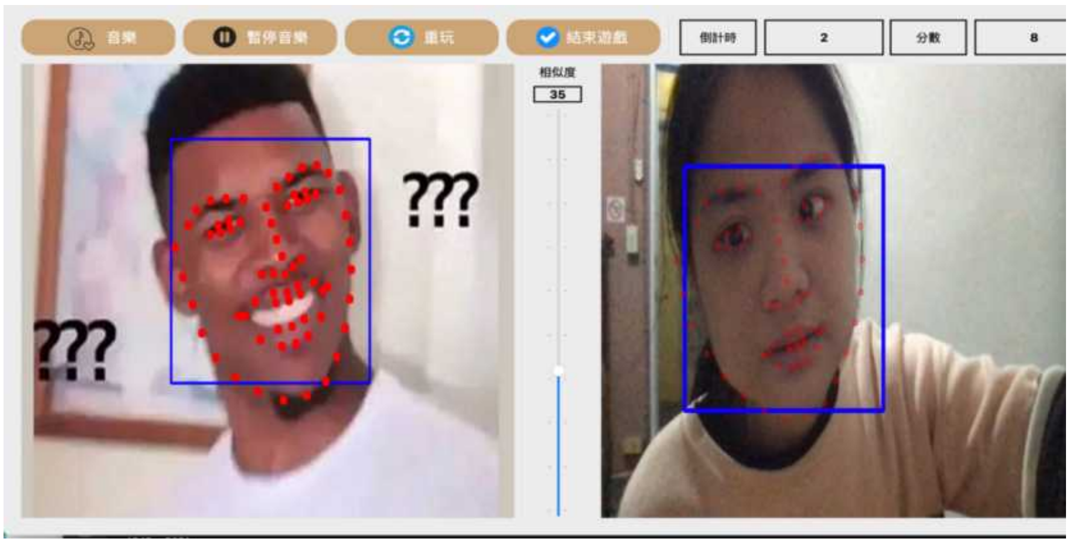
4. Game interface:

Emoticons or other players randomly appear on the left side, and are scored according to the expressions displayed by the camera on the right side. The closer to the left side, the higher the emotional score.

- Emoticons appear randomly
- music click to continue music
- Replay back to start page
- End the game to go to the end page
- time display countdown
- Pause button to pause music

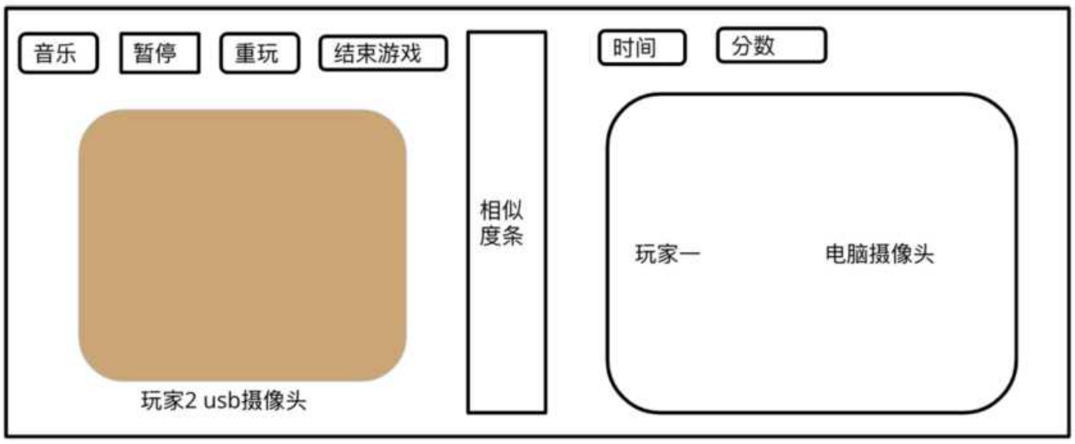
Single player mode:

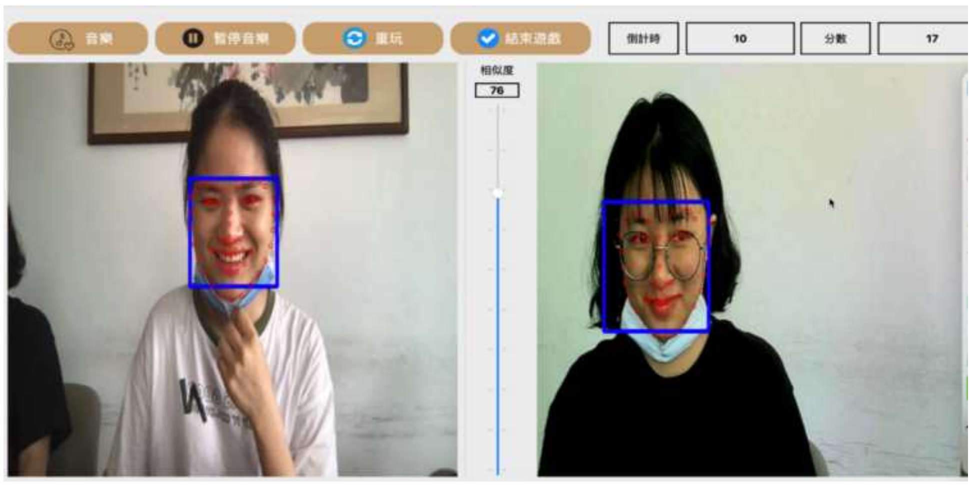




Double mode:

Players on both sides have 15s to make expressions, and after 15s, compare the similarity and give scores

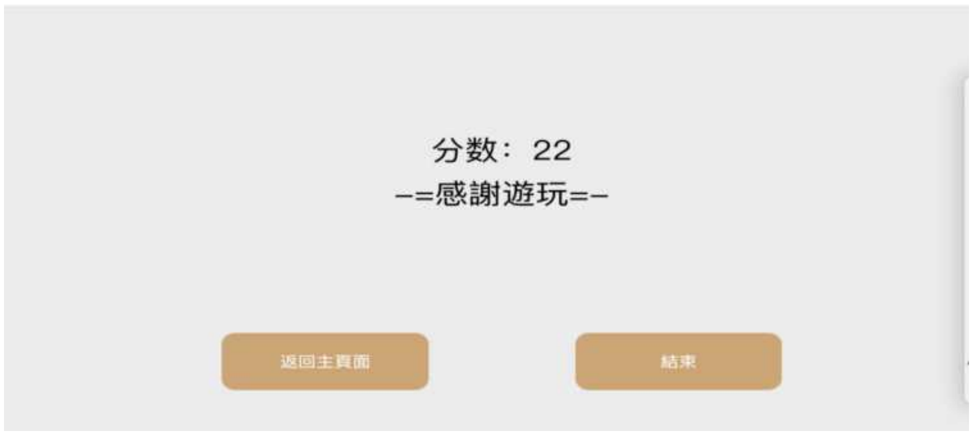




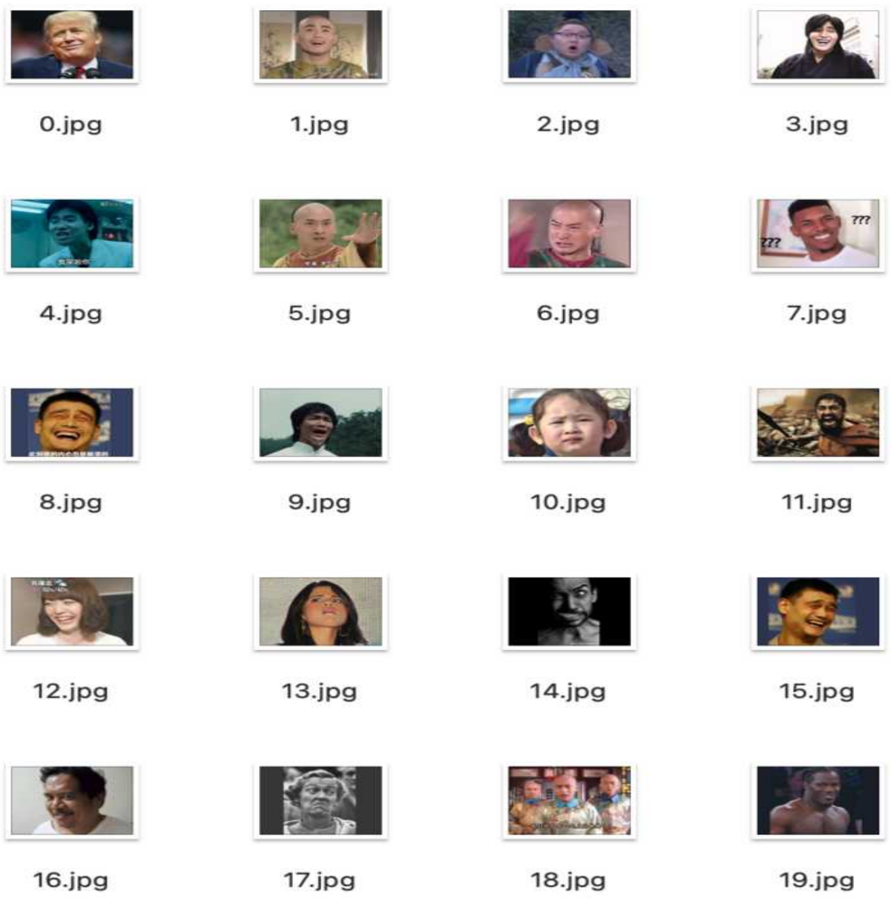
If no camera is detected on the left side of the two-player mode, the image will be replaced by: no_signal.jpg

Scoring rules:
Points are given according to the matching degree of the expression:
A matching degree of 80% or above is considered perfect , and the perfect score is 10 points;
A matching degree of 60-80% is excellent , and the excellent score is 7 points;
If the matching degree reaches 30-60%, it is good , and the good score is 4 points; if the matching degree is less than 30%, it is loss , and there is no score for loss .

5. Game end interface: The final score is displayed here.



6.Emoticons:



Conclusion

Development difficulties

Difficulty:

1. The problem of drawing key points in real time
2. How to eliminate the face difference problem
3. The Algorithm of the Alignment Step: How to Unify the Scaling Matrix and Angle Alignment
4. Calculation formula of similarity: how Euclidean and cosine functions are converted into similarity
5. How to smooth the result: Gaussian filter values

Future Outlook

In the production and study of the topic, we learned the method of facial expression recognition. We use dlib to capture face feature points, and operate and calculate these feature points. Although at this stage we can get results for expressions with larger movement ranges, but for some micro-expressions and expressions with small movement ranges, the test results are still not so accurate.

After consulting papers and searching for relevant research, we learned that the existing facial expression recognition technology has a relatively limited scope of application. Basically, it can only recognize exaggerated facial expressions, and the recognition range for micro-expressions is extremely limited.

I hope that in the future, face recognition technology will have new breakthroughs in micro-expressions. In this way, facial expression recognition can be applied to a wider range of fields in the future, and it can also bring greater convenience to people's work and life.

Experience

This special production let us understand a series of rich achievements such as face recognition and the technologies applied to these achievements. At the beginning, these technologies made us feel very fresh who have not yet had a deep understanding of face recognition. It also inspires us to understand how these interesting results are made.

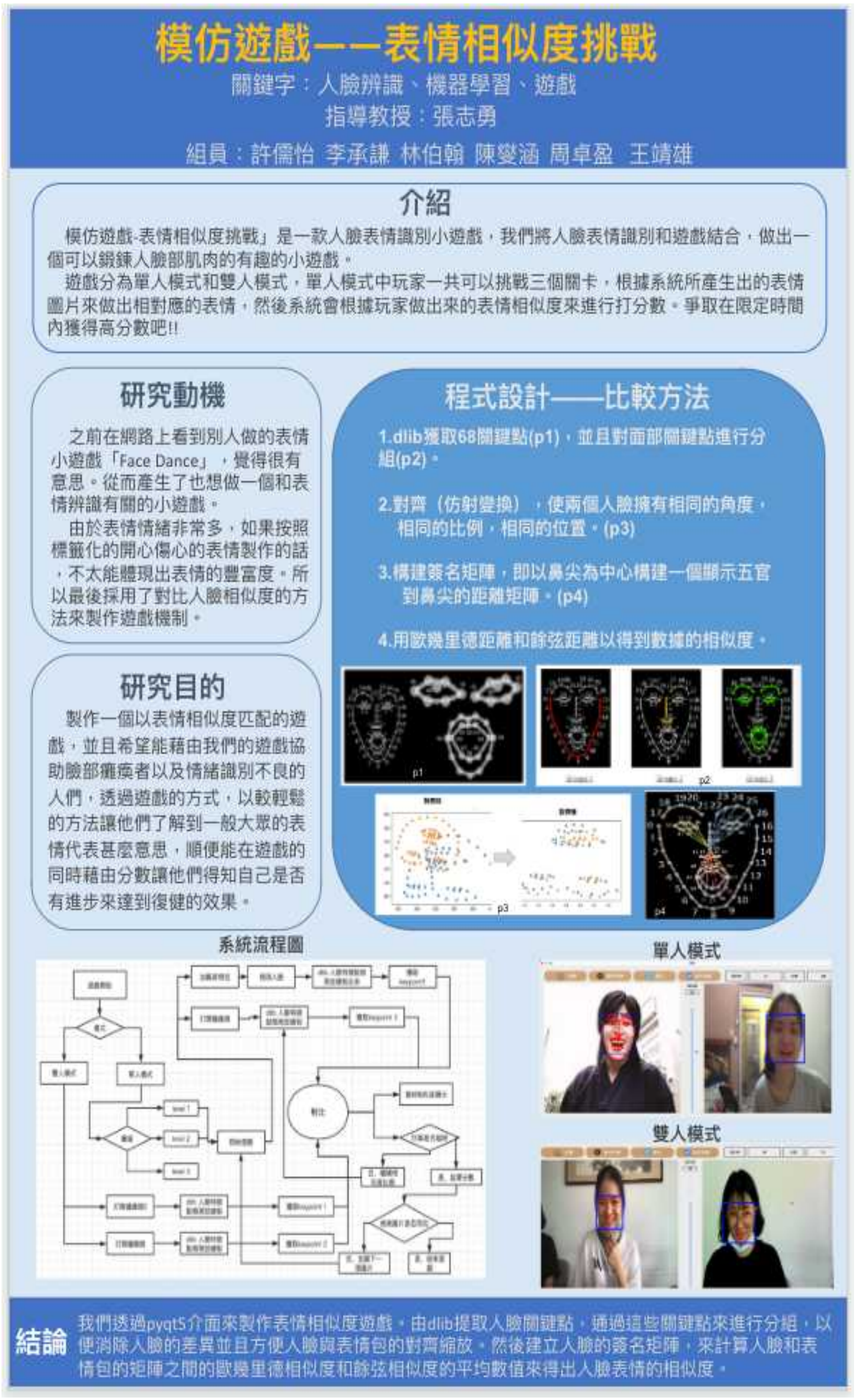
We first came into contact with face recognition through the technology of OpenCV. And in the step-by-step exploration and learning, we learned that most of the facial expression recognition now uses tags to define whether the expression is happy or sad; or directly artificially stipulates the value of the expression. In some cases, it seems a bit rigid, and we feel that this is less able to show the richness of expressions.

Through discussion within the group, we decided to use mathematical methods to determine the similarity between two expressions. Canceled the emoji expression in the original expression library, and changed it to an emoji package that is not defined by tags and is common on the Internet, which increases the fun of the game. A certain score is given by calculating the similarity between the expression pack and the player's expression. However, this directional change also correspondingly increases the difficulty of algorithm design.

In terms of solving the similarity between two expressions, we first obtained the key points of the face through dlib, then performed an affine transformation to align the expression package and the player's expression, and then calculated the similarity through Euclidean and cosine distance. In this process, we also learned a lot of mathematics that will be used in image processing, and have a deeper understanding of the importance of mathematical thinking in affecting processing.

Although some problems frequently appeared in the middle of the project, the problems were solved one by one through the mutual help of the team members. In this process, everyone also has their own experience of the topic they have done, and a deeper understanding of the applied technology.

Poster



模仿遊戲——表情相似度挑戰

關鍵字：人臉辨識、機器學習、遊戲

指導教授：張志勇

組員：許儒怡 李承謙 林伯翰 陳變涵 周卓盈 王靖雄

介紹

模仿遊戲-表情相似度挑戰」是一款人臉表情識別小遊戲，我們將人臉表情識別和遊戲結合，做出一個可以鍛鍊人臉部肌肉的有趣的小遊戲。

遊戲分為單人模式和雙人模式，單人模式中玩家一共可以挑戰三個關卡，根據系統所產生出的表情圖片來做出相對應的表情，然後系統會根據玩家做出來的表情相似度來進行打分數。爭取在限定時間內獲得高分數吧!!

研究動機

之前在網路上看到別人做的表情小遊戲「Face Dance」，覺得很有意思。從而產生了也想做一個和表情辨識有關的小遊戲。

由於表情情緒非常多，如果按照標籤化的開心傷心的表情製作的話，不太能體現出表情的豐富度。所以最後採用了對比人臉相似度的方法來製作遊戲機制。

研究目的

製作一個以表情相似度匹配的遊戲，並且希望能藉由我們的遊戲協助臉部癱瘓者以及情緒識別不良的人們，透過遊戲的方式，以較輕鬆的方法讓他們了解到一般大眾的表情代表甚麼意思，順便能在遊戲的同時藉由分數讓他們得知自己是否有進步來達到復健的效果。

程式設計——比較方法

1.dlib獲取68關鍵點(p1)，並且對面部關鍵點進行分組(p2)。

2.對齊（仿射變換），使兩個人臉擁有相同的角度，相同的比例，相同的位置。(p3)

3.構建簽名矩陣，即以鼻尖為中心構建一個顯示五官到鼻尖的距離矩陣。(p4)

4.用歐幾里德距離和餘弦距離以得到數據的相似度。

系統流程圖

單人模式

雙人模式

結論 我們透過pyqt5介面來製作表情相似度遊戲。由dlib提取人臉關鍵點，通過這些關鍵點來進行分組，以便消除人臉的差異並且方便人臉與表情包的對齊縮放。然後建立人臉的簽名矩陣，來計算人臉和表情包的矩陣之間的歐幾里德相似度和餘弦相似度的平均數值來得出人臉表情的相似度。

Division of labor

Ruyi Xu: scheme , UI , report writing, algorithm research

Zhuoying Zhou: game design, poster production

Chengqian Li: front-end testing, suite testing

Xiehan Chen: algorithm research, kit learning, front-end testing

Bohan Lin: Find information, kit learning

Jingxiong Wang: searching for information, making slideshows

References

1. Anis Kacem, Mohamed Daoud, Boulbaba Ben Amor, Juan Carlos Alvarez Paiva A Novel Space-Time Representation on the Positive Semidefinite Cone for Facial Expression Recognition, DOI: 10.1109/ICCV.2017.345
2. Mo, S., & Huang, Y. (2019, January 23). 2018Fall-FaceDanceMachine . GitHub. https://github.com/NTUEE-ESLab/2018Fall-FaceDanceMachine
3. Cheng, C. (2020, February 9). Comparison of Euclidean distance and cosine similarity . Medium. https://medium.com/qiubingcheng/Comparison of Euclidean distance and cosine similarity-c78163ad51b
4. The teacher can obviously rely on appearance . (2021, September 1). The basic calculation process of face alignment . Bilibili. https://www.bilibili.com/read/cv12974179
5. Handwritten ai . (2021, August 9). Alignment technology in face recognition, Similarity Transformation . Zhihu. https://zhuanlan.zhihu.com/p/393037076
6. 3blue1brown. (2016, August 8). Linear Transformations and Matrices | Chapter 3, Essence of Linear Algebra . Youtube. https://www.youtube.com/watch?v=kYB8IZa5AuE
7. Opencv affine transformation: https://docs.opencv.org/4.x/da/d54/

Code Link ：

https://github.com/rubymiaomiao/Facial-expression-similarity