华南理工大学

South China University of Technology

《Artificial Intelligence and 3D Vision》 Report

Project Name :	3D Lipstick Special Effects
School :	School of Future Technology
Major :	Data Science and Big Data Technology
Student Name :	全秦霄、邓靖丰、刘晶、刘青淳、晏璐涛
Teacher :	张怀东
Submission Date:	2023-06-23

1. Requirement Analysis for System

1.1 The Background and Motivation of System

With the continuous development of Internet technology, more convenient and affordable e-shopping is gradually becoming the main shopping method of this generation. Although e-shopping has the advantages of convenience, affordability and variety of choices, people may end up buying products that do not meet their expectations, especially in the beauty industry, because they only rely on pictures and videos to know the products and cannot actually feel whether the products are suitable for them. Therefore, some e-shopping platforms provide a virtual makeup trial function for beauty products, allowing users to visually see the suitability of their skin tone and facial features when using different colors of makeup on their faces, helping them to make better decisions. Currently, the virtual makeup trial function on the market mainly provides people to feel how different shades of lipsticks fit their skin tone after wearing them. Therefore, our group also hopes to help users feel the actual effect of using different shades of lipsticks by adding lipstick effects through the video obtained from the user's own computer camera.

1.2 System Objectives

After the user starts the program, the detection of the face and the extraction of key points can be achieved by capturing video frames through the camera. After further determining the key points of the lip area, the lipstick special effect of the face is realized by creating a mask of the lip area, and, furthermore, the lipstick color is adjusted by depth data to make the generated result more realistic and three-dimensional.

2. Program Analysis

2.1 Key Issues for System

1. Detect the face through the camera and get the key points of the five facial features to determine the position and outline of the lips.

2. Based on the RGB value and depth data set by the user through the slider, the color is adjusted and the lip area is colored.

2.2 Duty Assignments

Yanlutao, liuqingchun, liujing to achieve detection of the face as well as extraction of key points of the five facial features; quanqingxiao, dengjingfeng to achieve coloring the lip area as well as adjusting the lipstick color according to the depth data of the beauty effects function. The report of the project was completed by five people dividing the work together.

3. Technical Routine

3.1 Runtime Environment

Windows 11, Python 3.7 (OpenCV, NumPy), Media-pipe

3.2 General Design

This project can be seen as an example of the application of artificial intelligence and 3D vision technology. Combining knowledge and technology from computer vision, machine learning, deep learning and other fields, the Face Mesh model in Mediapipe used by the program is a pre-trained deep learning model that can accurately detect the position of 468 key points on a face. This involves knowledge of deep learning and computer vision.

The program takes into account the depth information of each pixel in the image, and colors the lip area based on the depth information, so that the resulting result is more realistic and three-dimensional. In this project, OpenCV, a powerful computer vision library, was used to create masks, apply colors, blur and other steps to modify and optimize the original image.

Overall, the design and implementation of this project involved knowledge and technology of artificial intelligence and 3D vision. The artificial intelligence part is mainly reflected in the use of MediaPipe, a deep learning library to detect face key points, and the 3D vision part is mainly reflected in changing the color of the lips according to the depth information of the key points, which involves the knowledge of 3D vision.

We use this flowchart to describe the entire program running process:

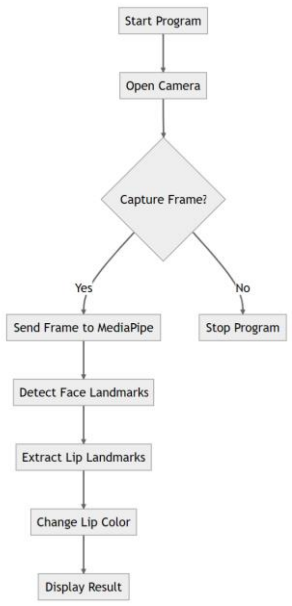


Figure 1 Program flowchart

① The user launches the program.

② The program turns on the camera to start capturing video frames.

③ In the process of capturing each frame of the image, the program will send the image frame to MediaPipe for the detection of face keys. Then, for each frame of the image, we use a pre-trained Face Mesh model for face detection and key point extraction. This step yields coordinates of 468 key points, including those in the lip area.

④ MediaPipe returns the detected face key. On the basis of key point extraction, we further determine the key points in the lip area and create a mask of the lips based on these key points.

⑤ The program extracts the key points of the lips.

⑥ The program changes the color of the lips according to the color and depth information selected by the user. Testers select their desired color via a slider in an OpenCV window. This step produces an RGB color value. Then we apply the selected color to the lip mask and adjust the color considering the depth information of each pixel to make the resulting result more realistic and three-dimensional.

⑦ The program displays the results in the window of OpenCV.

⑧ The tester can stop the program at any time.

3.3 Detailed Design

3.3.1Extract face key points

In this part, we mainly use the Face Mesh method in the mediapipe module. By importing facemesh model and configuring its parameters, we can directly invoke the process method of FaceMesh model to detect 468 key points of human faces in the image.

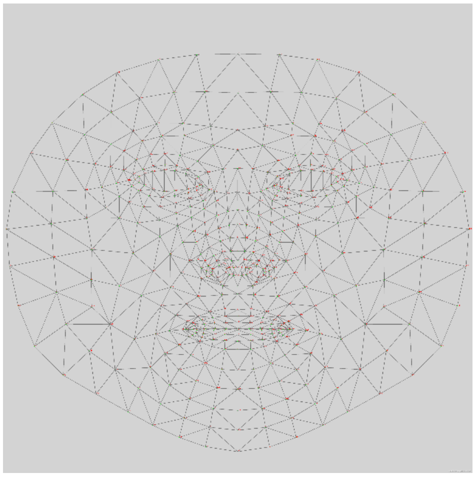
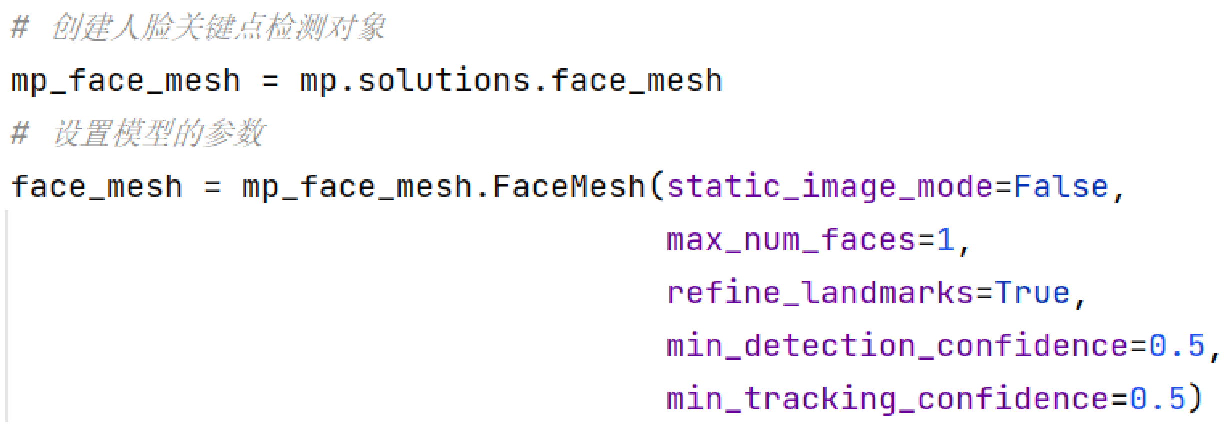


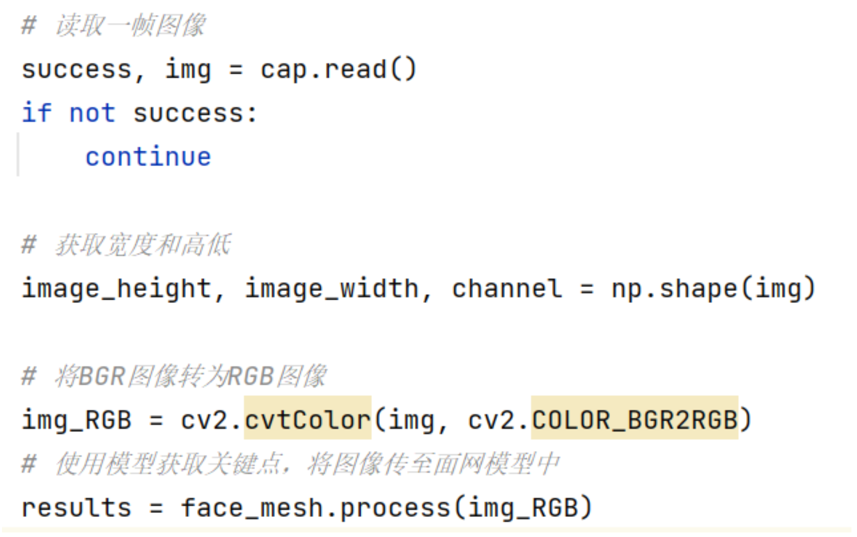
Figure 2 468 FaceMesh

Let's explain the implementation in detail. First, we import the mediapipe module, then we create a face key point detection object, and import the key point detection model FaceMesh to it, and set the parameters of the model.Different parameters have different meanings. static_image_mode is set to False to detect static images. max_num_faces is set to 1, which indicates the maximum number of faces that can be detected.refine_landmarks is set to True to indicate keypoints for locating the lips, eyes, and pupils. min_detection_confidence and min_tracking_confidence represent the confidence of face detection and face tracking, respectively.



```python
# 创建人脸关键点检测对象
mp_face_mesh = mp.solutions.face_mesh
# 设置模型的参数
face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False,
                                  max_num_faces=1,
                                  refine_landmarks=True,
                                  min_detection_confidence=0.5,
                                  min_tracking_confidence=0.5)
```

Then we read an image, use the cv2.cvtColor method to convert the color channel of the image from BGR to RGB, and then call the Process method of the FaceMesh model to obtain 468 key points of the face.

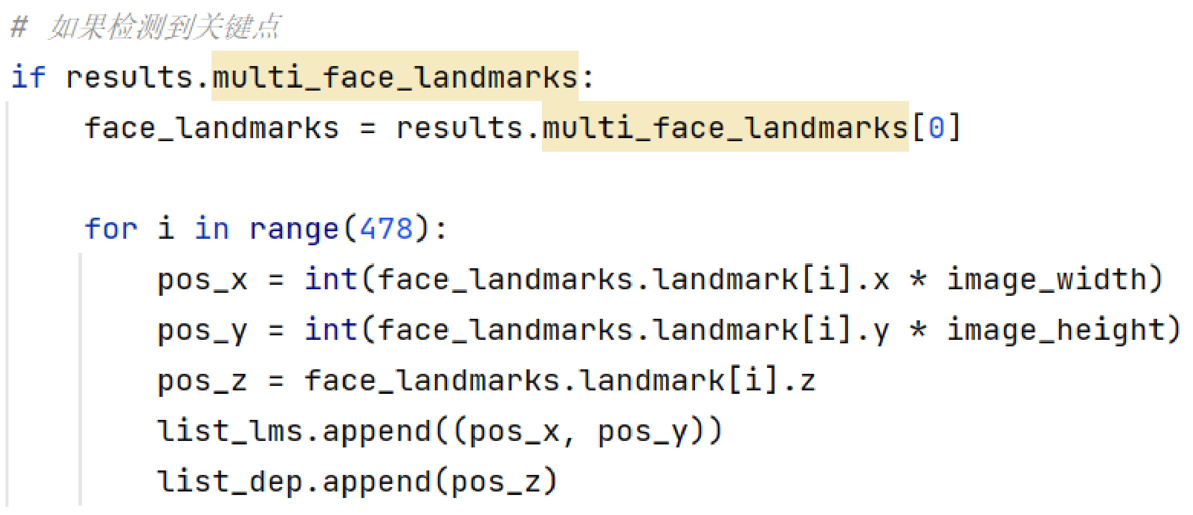


```python
# 读取一帧图像
success, img = cap.read()
if not success:
    continue

# 获取宽度和高低
image_height, image_width, channel = np.shape(img)

# 将BGR图像转为RGB图像
img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# 使用模型获取关键点，将图像传至面网模型中
results = face_mesh.process(img_RGB)
```

Finally, we save the obtained key point coordinates in list_lms and list_dep. landmarks stores each face key point in a list-like structure, and each "landmark" key field in the structure stores the coordinates of the key point, including x, y, z. x and y normalized to [0.0, 1.0] by image width and height, respectively. z represents the depth of the key point, taking the depth of the center of the head as the origin. The smaller the value, the closer the key point is to the camera.



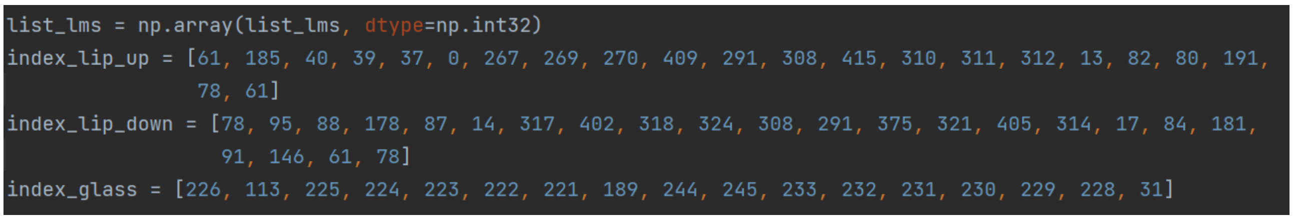
```
# 如果检测到关键点
if results.multi_face_landmarks:
    face_landmarks = results.multi_face_landmarks[0]

    for i in range(478):
        pos_x = int(face_landmarks.landmark[i].x * image_width)
        pos_y = int(face_landmarks.landmark[i].y * image_height)
        pos_z = face_landmarks.landmark[i].z
        list_lms.append((pos_x, pos_y))
        list_dep.append(pos_z)
```

3.3.2 Realize lipstick effect

For this part, the general steps can be explained as extracting the key points of the face using the face_mesh in 3.3.1, which include each area of the face (such as eyebrows, eyes, nose, lips, etc.), extract the feature points of the lip part, generate color according to the RGB value set by the user through the slider, first create a mask of the lip area, and then adjust the color of the lip area according to the depth map information, here it is also combined with the color adjustment algorithm in 3.3.3 to color, and return the image of the lip area with color changes. The implementation process is explained in detail below:

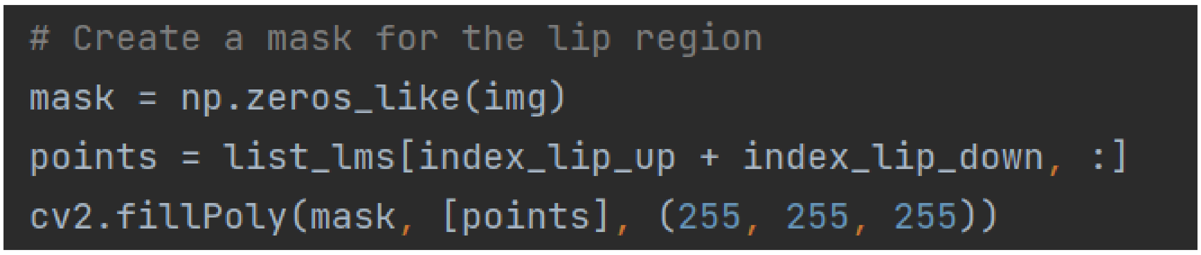
1. The position information of these key points extracted by "face_mesh" is stored in the variable list_lms, and the feature points of the lip part are extracted through specific indexes (index_lip_up and index_lip_down), and the lip area is picked.



```
list_lms = np.array(list_lms, dtype=np.int32)
index_lip_up = [61, 185, 40, 39, 37, 0, 267, 269, 270, 409, 291, 308, 415, 310, 311, 312, 13, 82, 80, 191,
                78, 61]
index_lip_down = [78, 95, 88, 178, 87, 14, 317, 402, 318, 324, 308, 291, 375, 321, 405, 314, 17, 84, 181,
                  91, 146, 61, 78]
index_glass = [226, 113, 225, 224, 223, 222, 221, 189, 244, 245, 233, 232, 231, 230, 229, 228, 31]
```

Figure 3 landmarks

2. Extract the shape of the lips from the face key and create a mask of the same size as the original image. cv2.fillPoly(mask, [points], (255, 255, 255)) This line of code draws a polygon on the mask, the vertices of the polygon are specified by points, and the fill color is white (255,255,255). Because points contain the coordinates of the key points at the edge of the lips, the polygon is the outline of the lips. In this way, the lip area in the mask is filled with white, and the rest of the area is still black. This enables the keying of the lip area for use in subsequent coloring steps.



```
# Create a mask for the lip region
mask = np.zeros_like(img)
points = list_lms[index_lip_up + index_lip_down, :]
cv2.fillPoly(mask, [points], (255, 255, 255))
```

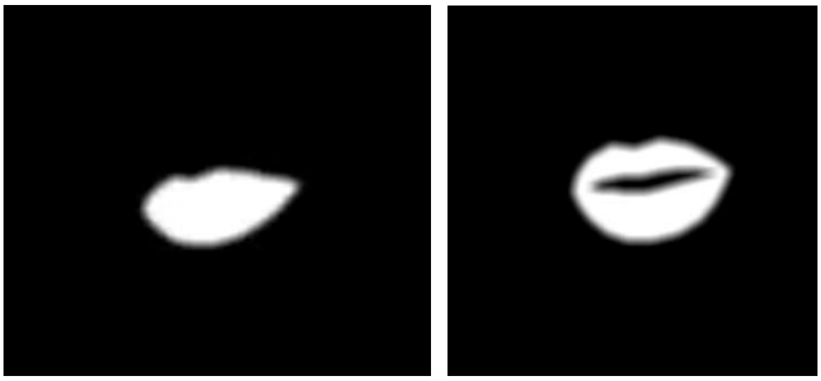
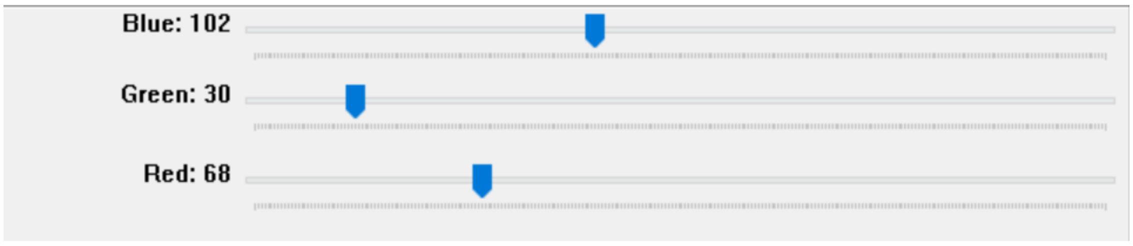


Figure 4 mask

3. Create an OpenCV window and add three color sliders (Blue, Green, Red) to adjust the color of your lips in real time. Colors are generated based on the RGB values set by the user via the slider. Then call the change_color_lip_with_depth function to color the lips.



Blue: 102

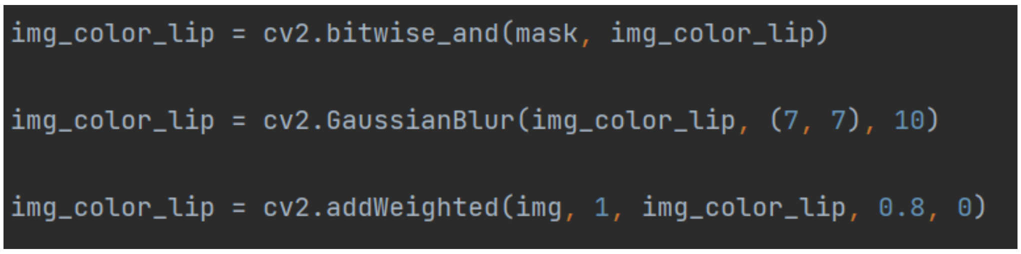
Green: 30

Red: 68

Figure 5 color slider

4. Apply masks and paint to the original image

When the mask of the lip area is obtained, it is fused with the original image to achieve the effect of placing the colored lips on the face. The following is a code example to achieve mask and face fusion:



```
img_color_lip = cv2.bitwise_and(mask, img_color_lip)

img_color_lip = cv2.GaussianBlur(img_color_lip, (7, 7), 10)

img_color_lip = cv2.addWeighted(img, 1, img_color_lip, 0.8, 0)
```

In this step, cv2.bitwise_and (mask, img_color_lip) is a bitwise AND operation, which can perform a bitwise and operation on the colored lip image (img_color_lip) and the mask (mask), which means that the pixels in the lip area maintain the color of img_color_lip, and the pixels in other areas become black.

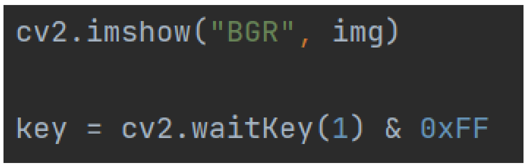
Apply the painted lip mask to the original image. Then use cv2. GaussianBlur blurs to make the color of the lips blend more naturally with the rest of the face. Use the cv2.addWeighted function to fuse the mask and the original image.

cv2.addWeighted is a function in the OpenCV library that weights two images. The basic principle is as follows:

The five parameters of the cv2.addWeighted(src1, alpha, src2, beta, gamma) function correspond one-to-one to the elements of the following formula

$$dst = src1 * alpha + src2 * beta + gamma$$

For each pixel in src1 and src2, the function multiplies their weights alpha and beta, respectively, and then adds the result, plus gamma, an optional scalar. This process processes every pixel in src1 and src2, so the resulting image is the same size as src1, src2. The cv2.addWeighted() function is used to merge the original image and the painted lip image together. The weights of the original image and the painted lip image are set to 1 and 0.8, respectively, which means that the original image contributes slightly more in the output image. The goal is to make the painted lips look more natural and not too bright.



```
cv2.imshow("BGR", img)

key = cv2.waitKey(1) & 0xFF
```

This completes the image processing for each frame, and uses cv2.imshow ()to display the processed image in the window. These steps are repeated every frame for real-time lip coloring.

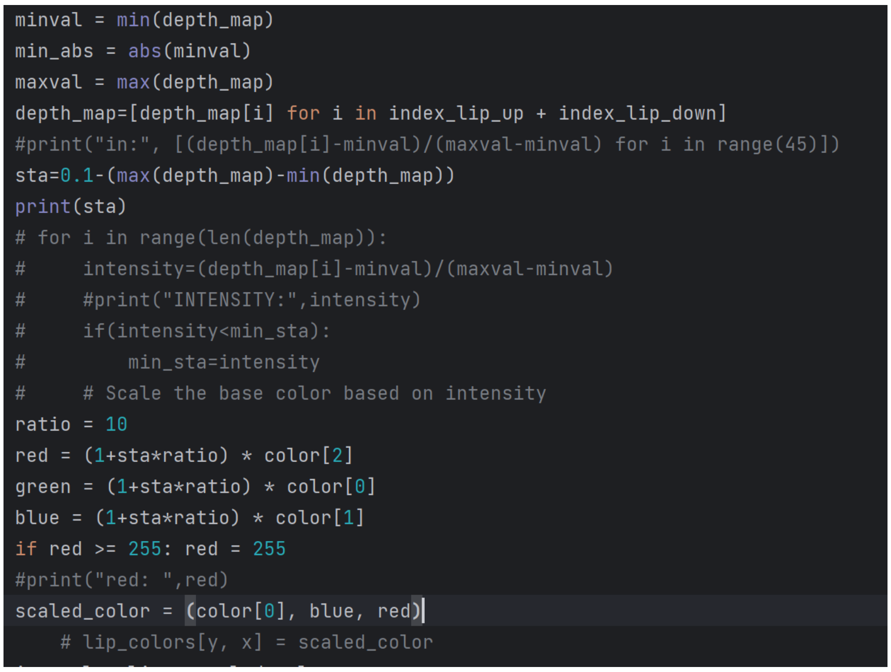
3.3.3 Lipstick Color Change and RGB Ratios

According to the depth data, we want to realize the change of lipstick color shades when the face is rotated to increase the realism of lipstick effects, for which we have two implementation methods in total.

Method 1：

When the face is facing the camera, the depth data of the 45 contour points should not differ much, but as the head turns, the larger the rotation, the larger the difference in the depth data of the contour points will be, so we can adjust the lipstick color according to the difference in the magnitude of the difference brought by the different rotation of the face.

We first obtain the maximum value maxval and the minimum value minval of the depth data depth_map of the 45 contour points of the lips, because the more vivid the lipstick color is when the face is facing the camera, the higher the RGB color is, while the more the face turns, the less vivid the lipstick color is, the lower the RGB color is, so we take the inverse of the difference value of the depth data.

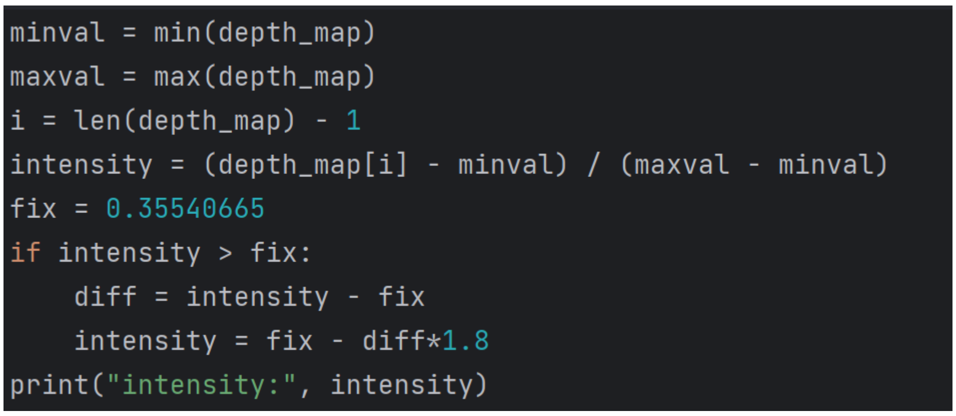


```
minval = min(depth_map)
min_abs = abs(minval)
maxval = max(depth_map)
depth_map=[depth_map[i] for i in index_lip_up + index_lip_down]
#print("in:", [(depth_map[i]-minval)/(maxval-minval) for i in range(45)])
sta=0.1-(max(depth_map)-min(depth_map))
print(sta)
# for i in range(len(depth_map)):
#     intensity=(depth_map[i]-minval)/(maxval-minval)
#     #print("INTENSITY:",intensity)
#     if(intensity<min_sta):
#         min_sta=intensity
#     # Scale the base color based on intensity
ratio = 10
red = (1+sta*ratio) * color[2]
green = (1+sta*ratio) * color[0]
blue = (1+sta*ratio) * color[1]
if red >= 255: red = 255
#print("red: ",red)
scaled_color = (color[0], blue, red)
    # lip_colors[y, x] = scaled_color
```

The constant 0.1 in the scaling factor sta is set according to our many experimental observations so that the new red, blue and green values will always be within the normal range as the difference changes. (1+sta*ratio) is used as the final adjustment parameter to change the color. Because the RGB color changes due to the different degree of superposition of the three colors, when the RGB ratio does not change, the RGB color does not change, so we only change the values of red and blue, and the size of green directly uses the value obtained from the slider.

Method 2:

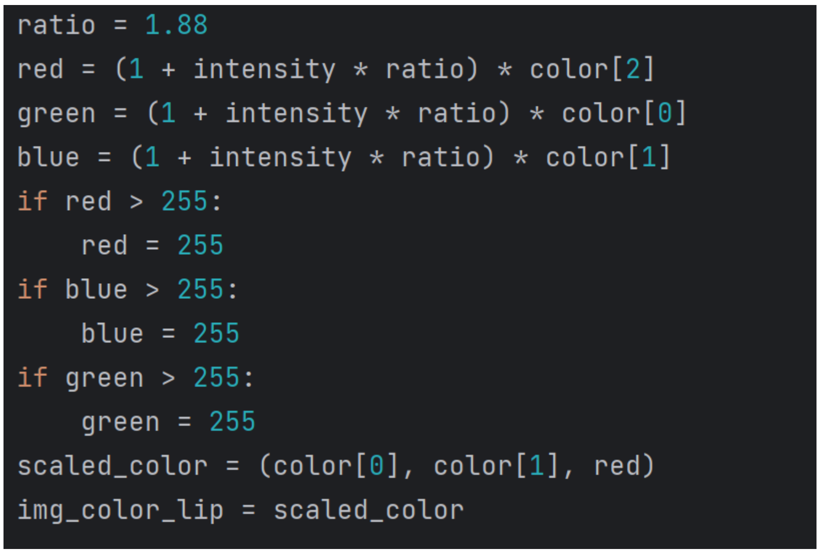
In the second method , we use intensity as a parameter to adjust the lipstick color, which is implemented as follows:



```
minval = min(depth_map)
maxval = max(depth_map)
i = len(depth_map) - 1
intensity = (depth_map[i] - minval) / (maxval - minval)
fix = 0.35540665
if intensity > fix:
    diff = intensity - fix
    intensity = fix - diff*1.8
print("intensity:", intensity)
```

Get the maximum and minimum values of the 45 contour points depth data and get the index of the last lip area depth data in the depth map. intensity = (depth_map[i] - minval) / (maxval - minval):

Calculate the intensity of the lip area depth value (between 0 and 1) based on the minimum depth value and the maximum depth value. A correction factor fix=0.35540665 is set to adjust the intensity values. If the intensity value is greater than the correction factor, the adjustment is made. Adjust the intensity value based on the difference, multiply by 1.8 to increase the degree of adjustment.



```
ratio = 1.88
red = (1 + intensity * ratio) * color[2]
green = (1 + intensity * ratio) * color[0]
blue = (1 + intensity * ratio) * color[1]
if red > 255:
    red = 255
if blue > 255:
    blue = 255
if green > 255:
    green = 255
scaled_color = (color[0], color[1], red)
img_color_lip = scaled_color
```

Lipstick color and RGB value ratios:

Through information search, we obtained the RGB ratios of popular lipstick colors in makeup on the market, as follows:

Color	Red	Green	Blue
Brick Red	178	48	41
Positive Red	194	3	13
Oxblood Red	106	5	0
Tomato Orange	160	33	18
Warm persimmon red	239	93	71
Positive Orange	191	27	28
Coral powder	242	122	122
Rose	208	10	57
Plum color	106	18	45

The color sequence obtained by running the program after setting the RGB values is as follows:

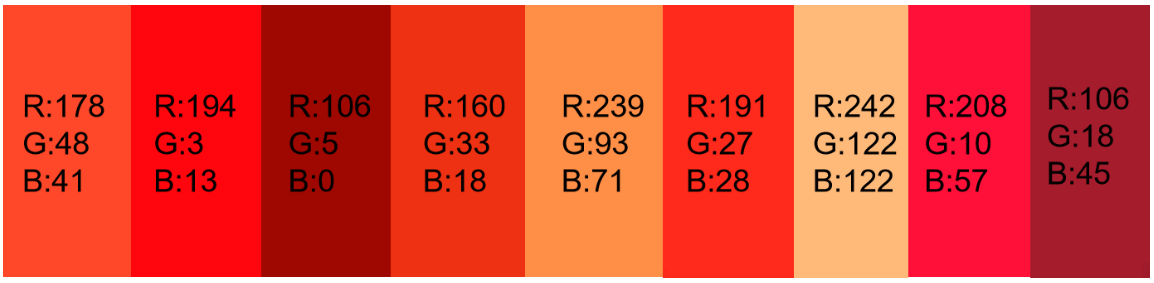


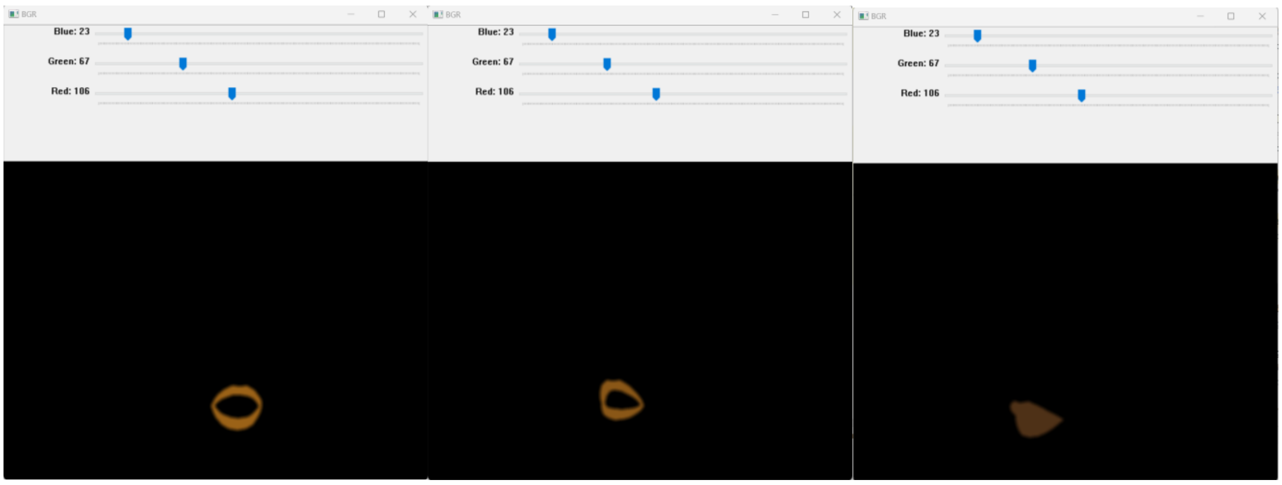
Figure 6 Classic color scheme

4. Programming Progress

Phases of Mission	**Period**	**Planned Completion**	**Actual Completion**
Learn about Github and Mediapipe	4.1-4.15	Understand the basics of Github and Mediapipe	Completed on time
Detailed project planning	4.16-4.30	Define the project direction and specific functions	Framework construction to verify project feasibility
Determine the direction of the project	5.1-5.17	Face 3D key point detection	Tutorial and build a basic framework
Refine the specific functions of the project	5.24-6.1	Realize lip discoloration function	Basically done
Control the experiment, adjust the parameters	6.1-6.15	Learn code logic, read related papers, add more functions, and optimize the display effect	Realize lip picking, depth reading, color and depth correlation algorithms
Package the project, write the report	6.15-6.20	Determine the division of labor and fill in the content	Completed on time

5. Testing report

1. Without the face layer, use the black layer


Blue: 23
Green: 67
Red: 106

Blue: 23
Green: 67
Red: 106

Blue: 23
Green: 67
Red: 106

Figure 7 rendering

2. Experiments with different RGB ratios

Here, we chose three classic color schemes to test, and adjusted the weights for the face layer and the lipstick color layer.

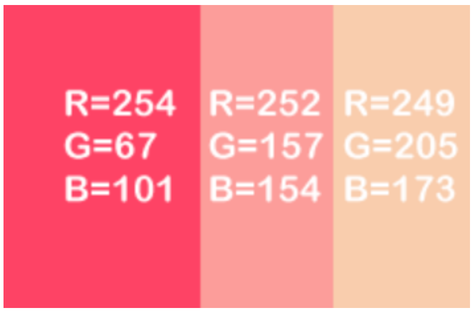
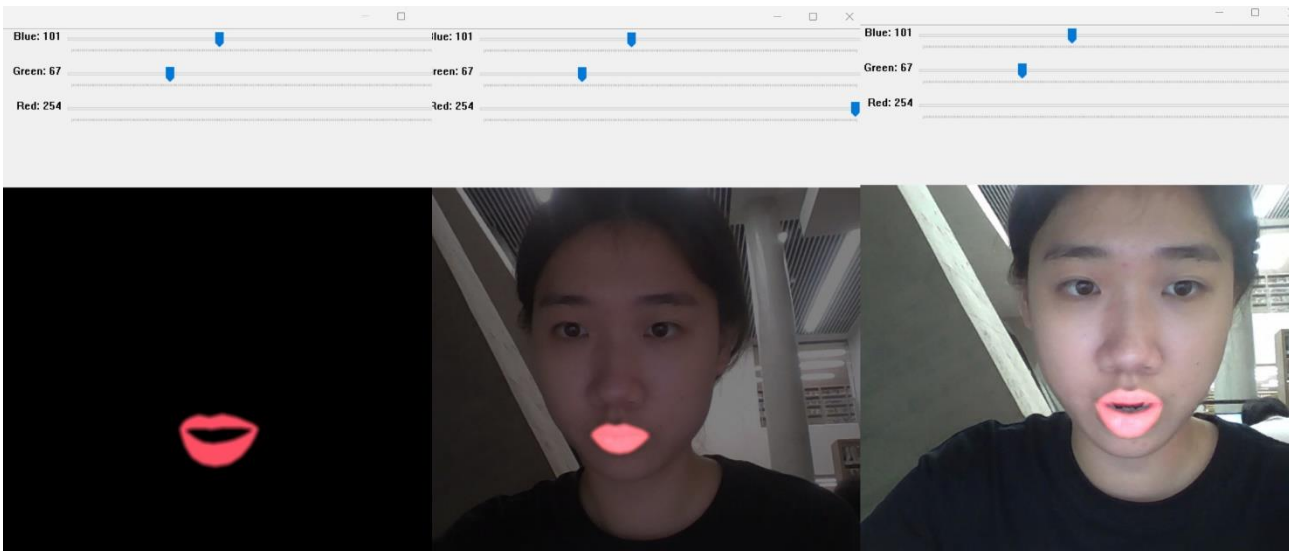

R=254
G=67
B=101
R=252
G=157
B=154
R=249
G=205
B=173

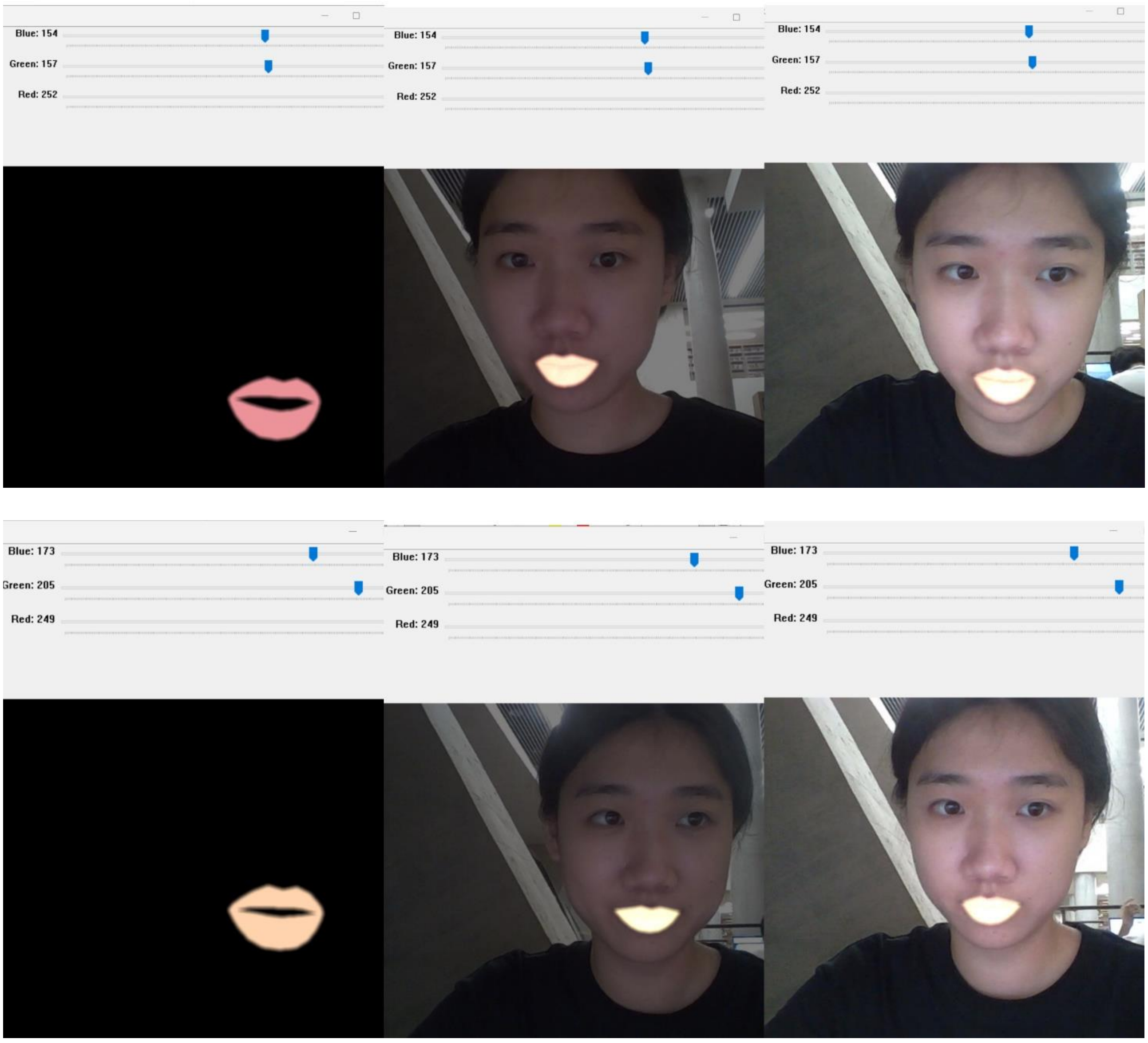
Figure 8 Color Matching

From left to right, the weights of the face layer and the lipstick color layer are 0:1, 0.5:1, and 1:0.8, respectively. You can see that the higher the weight, the more obvious the corresponding layer color will be.


Blue: 101
Green: 67
Red: 254

lue: 101
reen: 67
Red: 254

Blue: 101
Green: 67
Red: 254


Blue: 154
Green: 157
Red: 252

Blue: 154
Green: 157
Red: 252

Blue: 154
Green: 157
Red: 252

Blue: 173
Green: 205
Red: 249

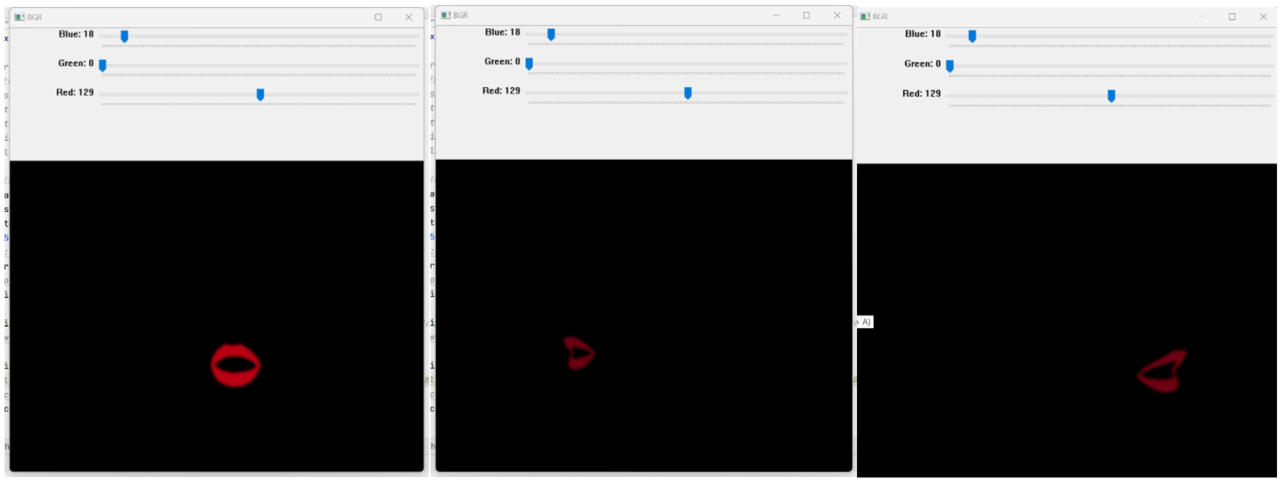
Blue: 173
Green: 205
Red: 249

Blue: 173
Green: 205
Red: 249

Figure 9 rendering

3. Comparative experiment with or without depth data adjustment

Add depth data, as the Angle changes, the color depth changes.

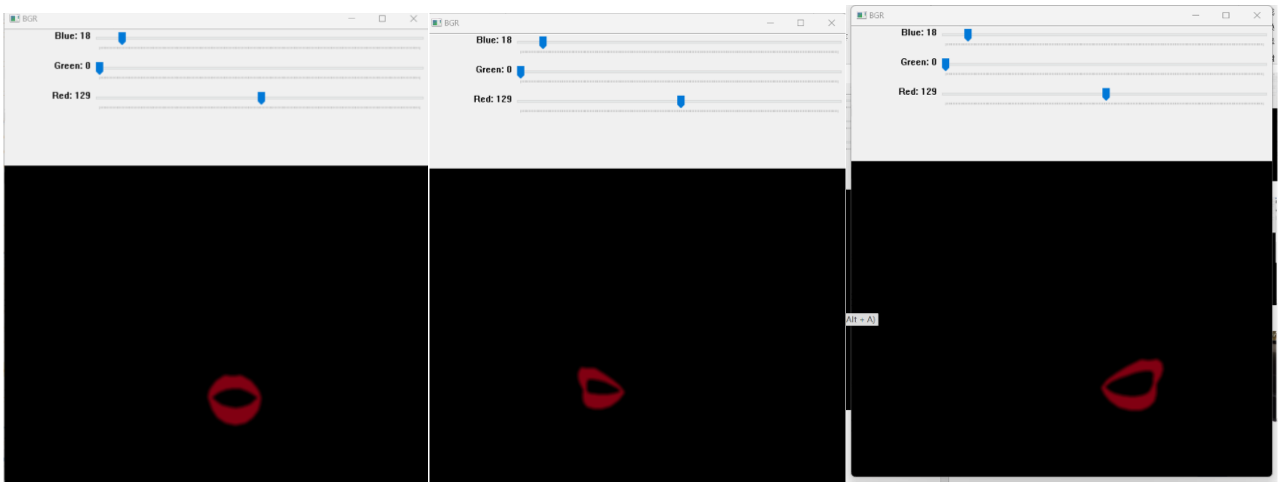

Blue: 18
Green: 0
Red: 129

Blue: 18
Green: 0
Red: 129

Blue: 18
Green: 0
Red: 129

Figure 10 rendering

Without depth data, there is no change in color depth as the Angle changes.



Blue: 18
Green: 0
Red: 129

Blue: 18
Green: 0
Red: 129

Blue: 18
Green: 0
Red: 129

Figure 11 rendering

4.Experiments comparing mobile phone pictures with real faces (based on a system adjusted with depth data)

When tested with mobile phone images, the face depth data was the same, so as the face turned, the shade of lipstick did not change.

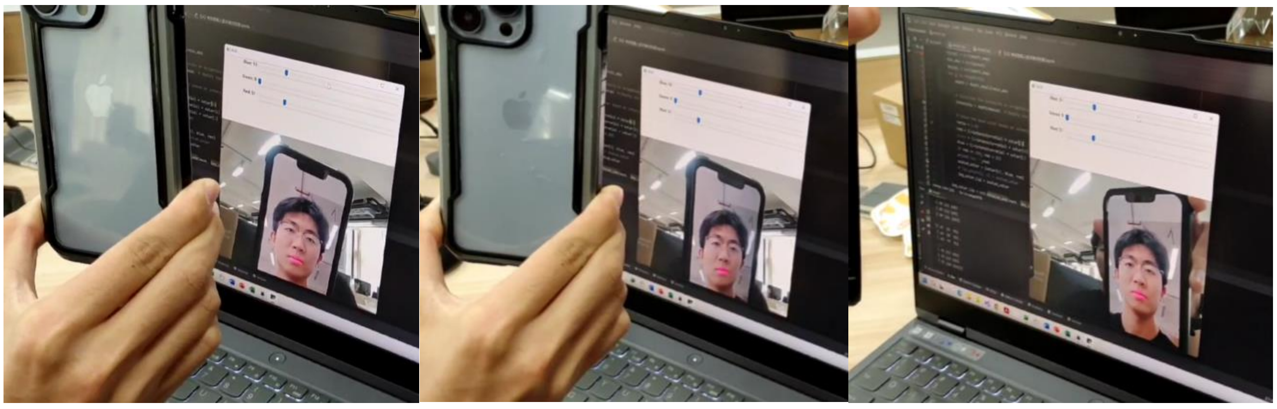


Figure 12 rendering

When tested with real faces, the face depth data is different, so as the face turns, the lipstick color will change.

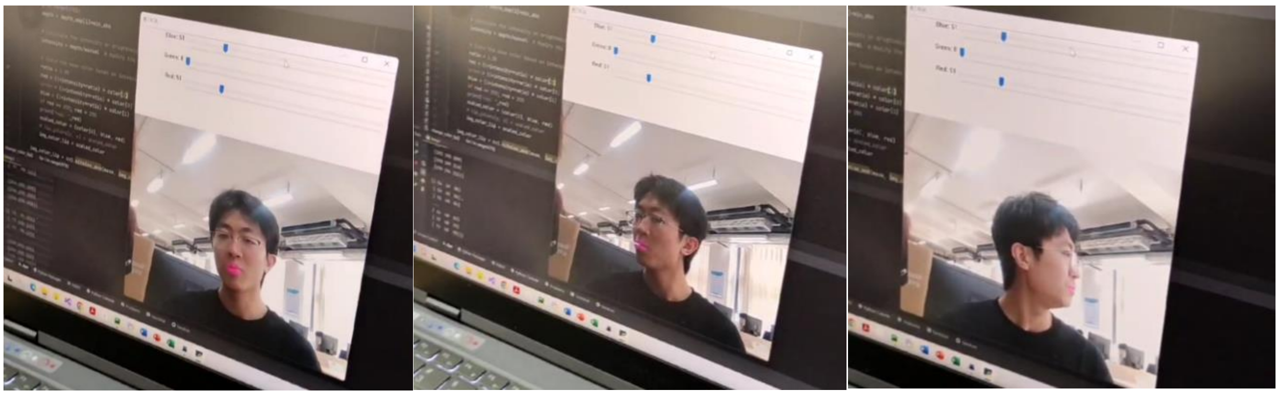
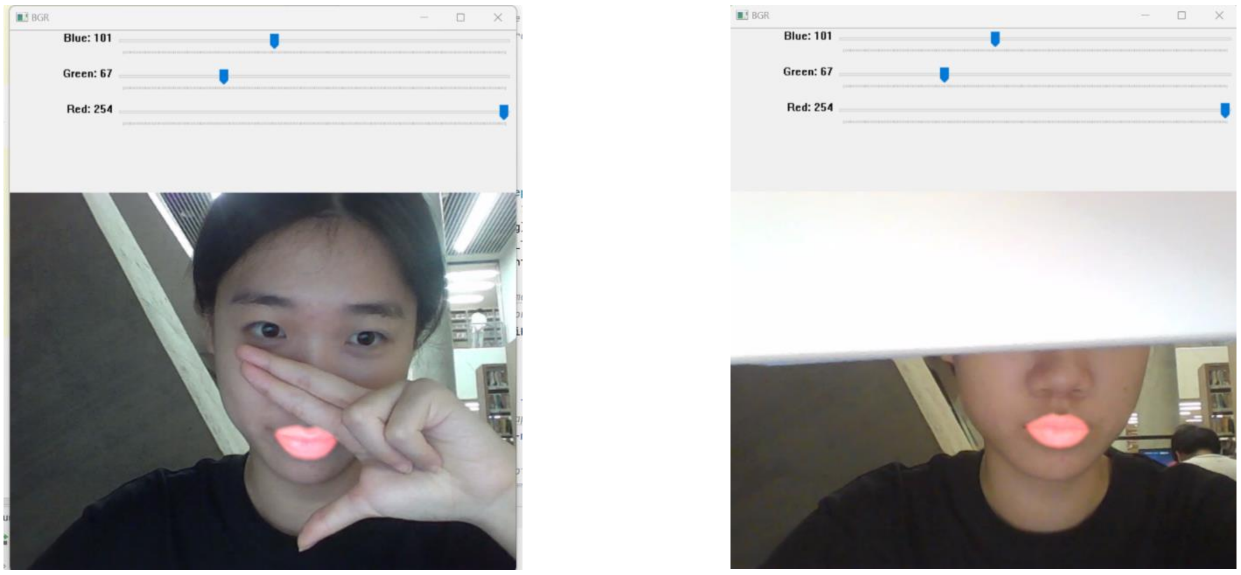


Figure 13 rendering

5.Covering part of your features

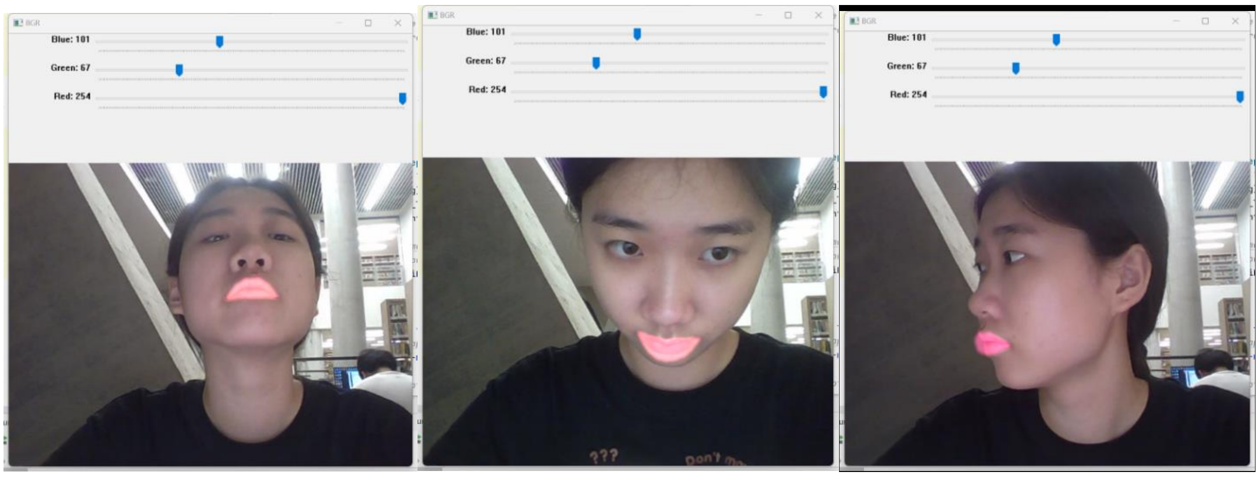


Blue: 101
Green: 67
Red: 254

Blue: 101
Green: 67
Red: 254

Figure 14 rendering

6.Changing facial orientation and expression



Blue: 101
Green: 67
Red: 254

Blue: 101
Green: 67
Red: 254

Blue: 101
Green: 67
Red: 254

Figure 15 rendering

6. Personal Summary

Liu Qingchun:

Through this course, I have learned more knowledge about artificial intelligence and 3D vision, including various interesting models and algorithms. In the final project of this course, I gained a deeper understanding of depth data through practice, and learned how to use the mediapipe module, which has many interesting applications such as pose recognition, face detection, gesture recognition, and hair segmentation. Additionally, during the testing process of the program, I also realized the impact of different factors on the lip effect of the lipstick special effect. In summary, although we only used the FaceMesh method in the module for our project, I have experienced the charm of combining machine learning and 3D vision, and gained a sense of accomplishment from working with my team to complete this lipstick special effect.

Yan lutao：

In this project, we successfully applied Mediapipe's Face Mesh model to obtain face keys in computer vision tasks and understand how to use these keys to complete the task of lip coloring. Our team sets clear goals, develops appropriate plans and timelines, and monitors progress to ensure the smooth running of the project. All in all, this big assignment provided me with a valuable opportunity to learn and apply knowledge in practice, and the team members worked hard to cooperate with each other, and my teamwork skills improved. The course of artificial intelligence and 3D vision also allowed me to learn a lot of AI knowledge in the first half, and did a lot of experiments on 3D vision in the second half, and constantly experienced the application process of knowledge in practice.

Quan Qinxiao:

First of all, I deeply studied the various solutions of mediapipe, learned to call mediapipe to solve problems, and figured out its algorithm logic. With the foundation of mediapipe, I started to innovate and research the generation of lipstick special effects. In the process, I have a better understanding of various libraries of opencv, and I feel the vividness of lipstick brought by deep data. I deeply feel To the charm of the field of 3D vision. And the overall idea of lipstick special effects was obtained by our innovation, which demonstrates our innovative ability. In short, at the technical level, I have improved my coding ability and gained a deeper understanding of the field of 3D vision. In this project, I also served as the team leader. Our group often discusses technical solutions together. During the whole organization process, I have exercised my organization and coordination skills, communication skills, and time management skills.

Deng Jingfeng:

In the first place, since I had not studied or used any Mediepipe related technology or algorithm model before I approached the project, my teammates and I searched the relevant literature, explored in depth the functions of its related models and core functions, and had a preliminary knowledge of the technical framework necessary for us to complete the project. Then, with the support of mediapipe technology, we were able to obtain depth information more quickly and easily, which led to an innovative proposal of color grading based on depth data to achieve 3D face effects. During this process, I got to know the various functions of opencv and its power - it was fun to implement interesting 3D dynamic effects with a few lines of refined code. Although the process of writing the code was not smooth - our 3D lipstick color effect did not reflect the depth change obviously or correctly, but eventually, with the help of communication with the team leader, we managed to achieve the target function, allowing the lipstick color to change more accurately with the fluctuation of depth data at key points, and I also found the 3D I found the fun of the 3D visualization hands-on project and the meaning of teamwork.

Liu jing:

Firstly, I think the biggest gain I got through this 3d vision course is that I have a general understanding of the theoretical knowledge related to 3d vision. Although many teachers have mentioned the practical applications of 3d vision, such as face recognition, body movement recognition, picture recognition, etc., I actually have no basic theoretical knowledge about these applications, so my understanding of it is only superficial. It was only through the ideas, algorithms and models introduced by the teacher in the 3d vision class that I felt I began to really understand what kind of research 3d vision actually is. The other thing I learned is that I learned to use mediapipe module and opencv module, and I learned to call the process method of facemesh model to get the 486 key points on the human face, and other related usage by completing the project by myself. After learning this course, I realized that 3d vision is really very much used in our daily life, and it is very responsive to the needs of the times.

7. Reference

[1] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." *arXiv preprint arXiv:1906.08172* (2019).

[2] Grishchenko, Ivan, et al. "Attention mesh: High-fidelity face mesh prediction in real-time." *arXiv preprint arXiv:2006.10962* (2020).

[3] MediaPipe | Google for Developers

[4] GitHub - google/mediapipe: Cross-platform, customizable ML solutions for live and streaming media.

[5] Aman, Sangal A L. Drowsy Alarm System Based on Face Landmarks Detection Using MediaPipe FaceMesh[C]//Proceedings of First International Conference on Computational Electronics for Wireless Communications: ICCWC 2021.

[6] Dantone, Matthias, et al. "Real-time facial feature detection using conditional regression forests." *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012.

[7] Bazarevsky, Valentin, et al. "Blazepose: On-device real-time body pose tracking." *arXiv preprint arXiv:2006.10204* (2020).

[8] Zhang, Fan, et al. "Mediapipe hands: On-device real-time hand tracking." *arXiv preprint arXiv:2006.10214* (2020).