RTSP real-time image transmission combined with AIDemo

Effect Demonstration

> Note: The WIFI chip on the K230 has limited performance. Please use the image transmission near a WIFI signal hotspot.

> *This example is experimental and may not run with high quality in different network environments.

When we run the modified example in this section, the console will have output information similar to the following:

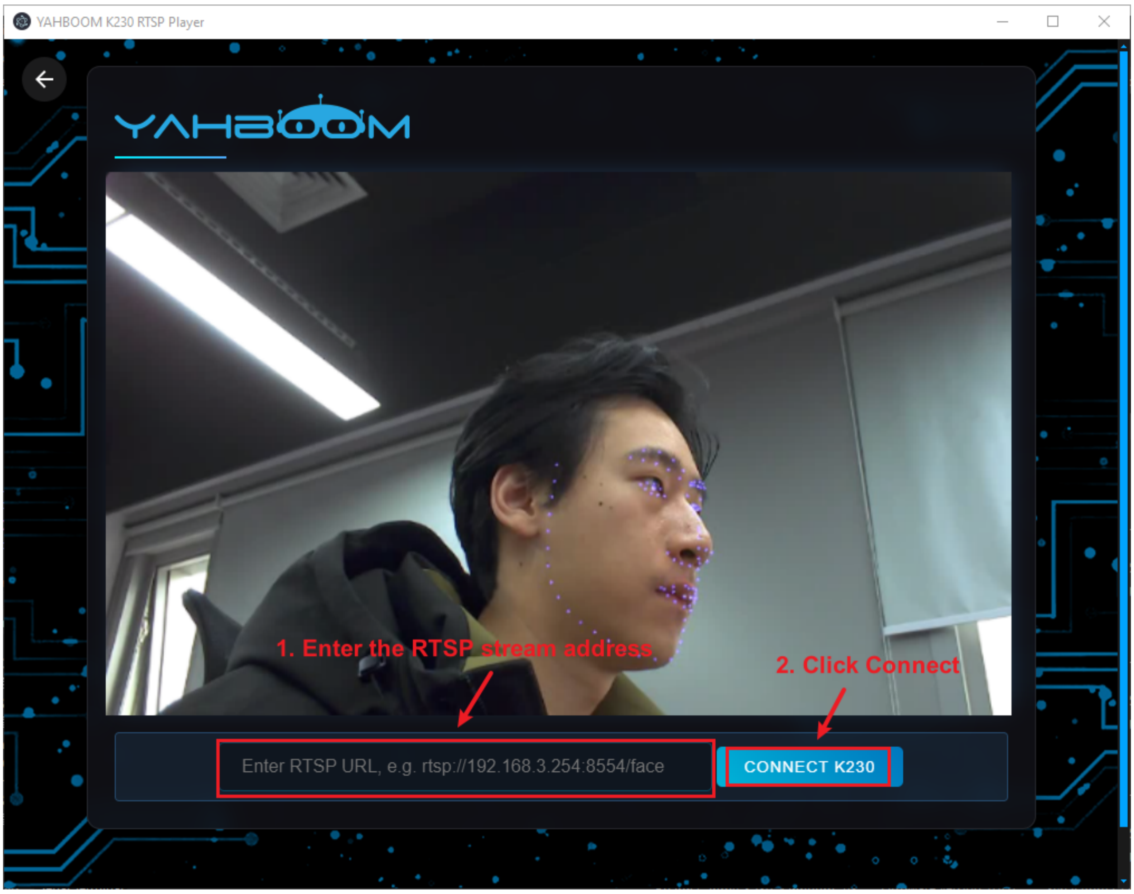
```
[WIFI] Connecting to network...
[WIFI] 连接网络中 ...
[RTSP] Starting...
[RTSP] 启动中 ...
find sensor gc2093_csi2, type 9, output 1920x1080@60
sensor(0), mode 0, buffer_num 4, buffer_size 0
rtsp server start: rtsp://192.168.2.116:8554/face
[RTSP] Started successfully!
[RTSP] 启动成功！
```

This address is the RTSP streaming address.

We can use a player that supports RTSP streaming to remotely play the K230 screen. This section takes face key point detection as an example.

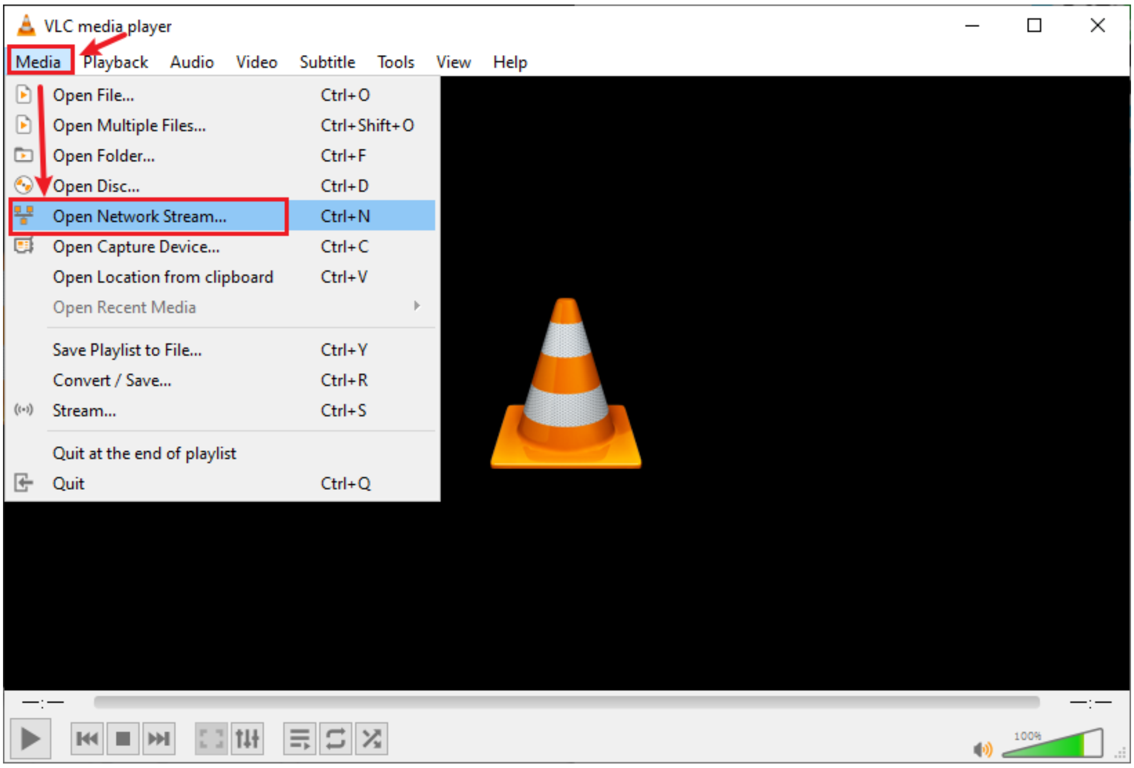
You can use VLC Media Player or the rtsp_player we provide. Both software are in our download area [Software Tools]

1. It is recommended to use rtsp_player first, which has simple functions and does not require complicated configuration. Just double-click to open it.

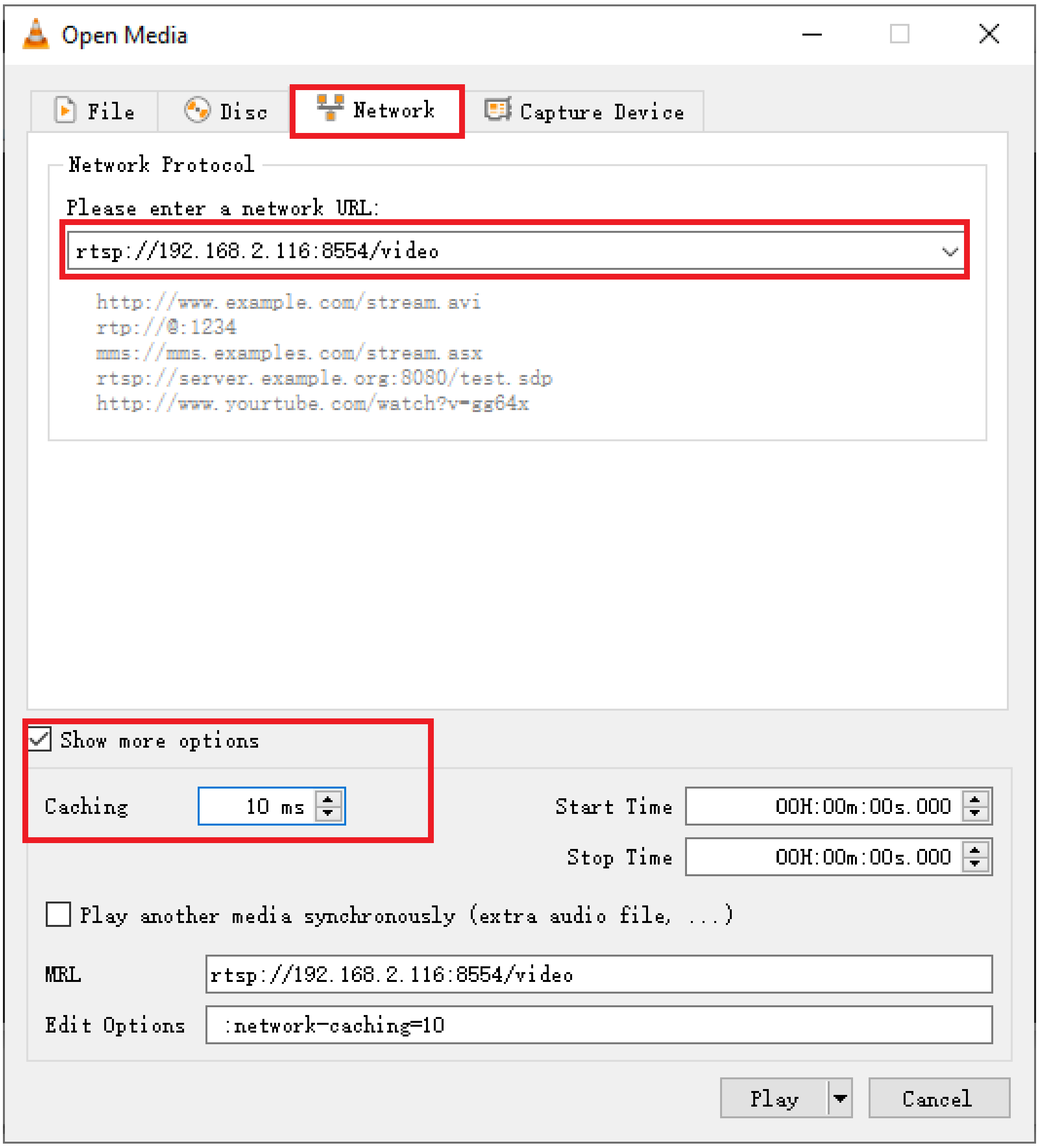

YAHBOOM K230 RTSP Player
YAHBOOM
1. Enter the RTSP stream address
2. Click Connect
Enter RTSP URL, e.g. rtsp://192.168.3.254:8554/face
CONNECT K230

2. VLC Media Player is a versatile and powerful video player, but the settings are relatively cumbersome.

   We open VLC Media Player, click "Media" -> "Open Network Stream"



Enter the URL address and change the delay to a lower value in "Show more options", about 10~100ms



Network Protocol

Please enter a network URL:

rtsp://192.168.2.116:8554/video

```
http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x
```

☑ Show more options

Caching 10 ms

Start Time 00H:00m:00s.000

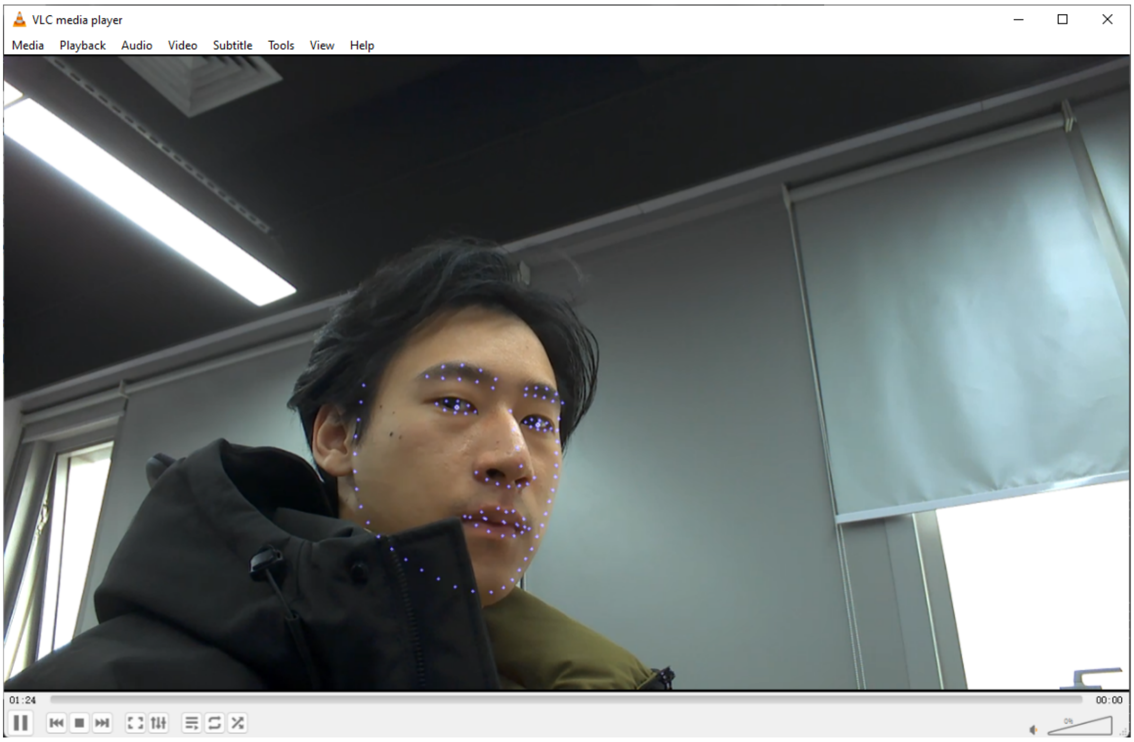
Stop Time 00H:00m:00s.000

□ Play another media synchronously (extra audio file, ...)

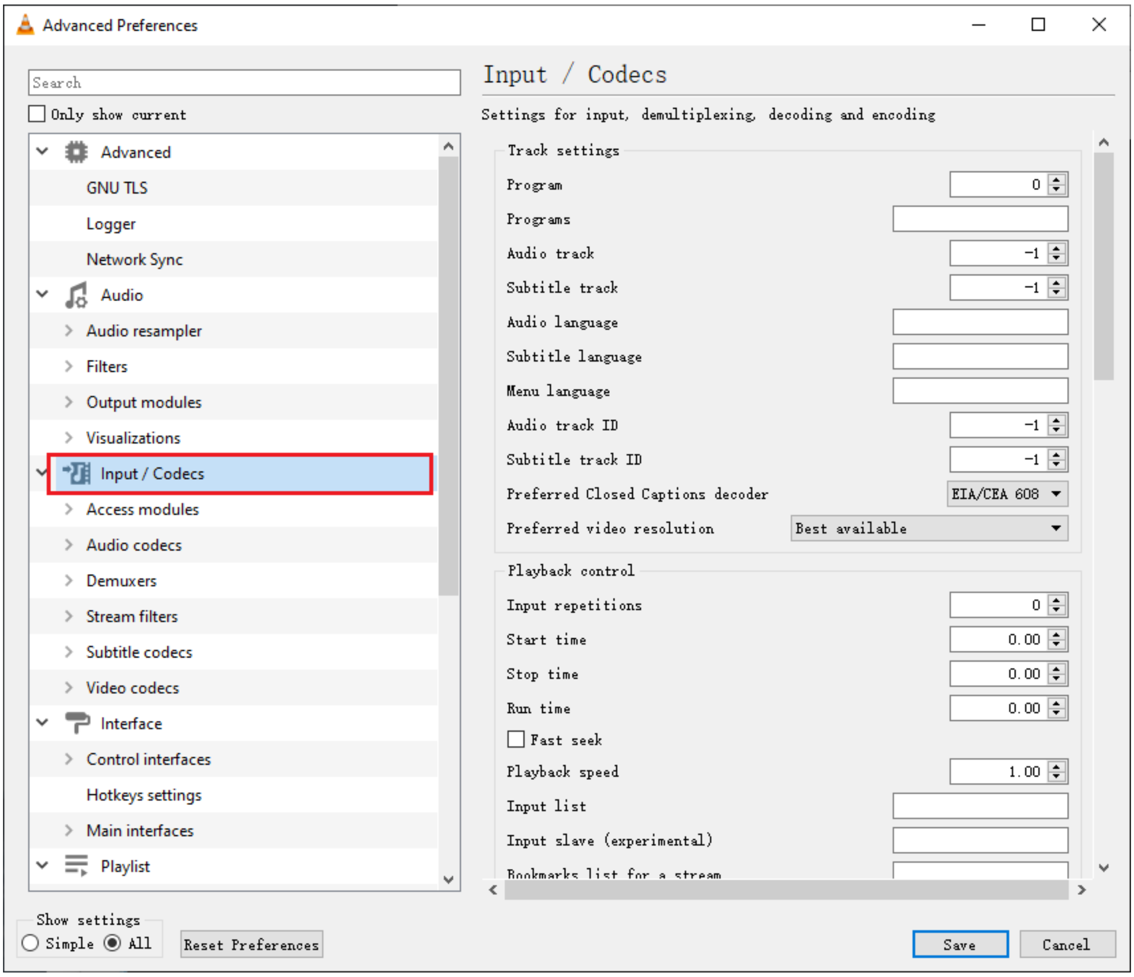
MRL rtsp://192.168.2.116:8554/video

Edit Options :network-caching=10

After the modification is completed, click the [Play] button in the lower right corner to remotely play the video transmitted by K230



If the playback performance is poor, one way is to ensure the network speed and connection stability, and the other is to try to optimize it in "Preferences".



Input / Codecs

Settings for input, demultiplexing, decoding and encoding

Track settings

Program 0

Programs

Audio track -1

Subtitle track -1

Audio language

Subtitle language

Menu language

Audio track ID -1

Subtitle track ID -1

Preferred Closed Captions decoder EIA/CEA 608

Preferred video resolution Best available

Playback control

Input repetitions 0

Start time 0.00

Stop time 0.00

Run time 0.00

□ Fast seek

Playback speed 1.00

Input list

Input slave (experimental)

Bookmarks list for a stream

Sample Code

Here is a modified image transmission code based on face detection

Put it in [Source code summary/11.Network/07.rtsp/rtsp_face_detect.py]

Modify the WIFI parameters and run to see the effect

```
if __name__ == "__main__":
    print("连接网络中 Connecting to network ...")
    Connect_WIFI("ssid", "password")
    print("启动中 Starting ...")
```

Modification steps

1. Preparation

In this tutorial, we will introduce how to remotely transmit the images recognized by AIDemo through RTSP.

> *This tutorial is somewhat challenging for beginners and requires you to have a certain code foundation and programming skills.*

Let's take face_landmark (face key point detection) as an example.

The required code is:

1. empty_rtsp_demo.py [located in source code/ 11.Network / 07.rtsp / empty_rtsp_demo.py]
2. face_landmark.py [located in source code/07.Face/02.face_landmark.py]

First, we copy a copy of the empty_rtsp_demo.py file and rename it to empty_rtsp_demo_face_landmark.py

1.1 Configure WiFi settings

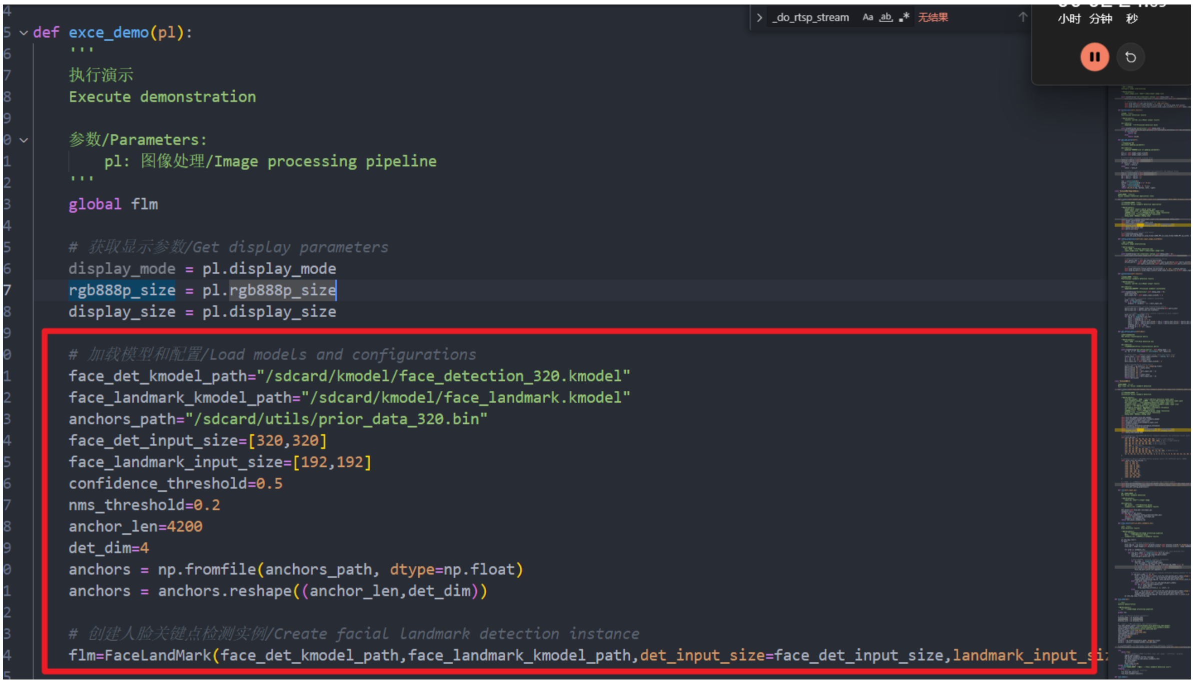
We scroll down to the bottom of the code and find the if name == "main" part

Modify the parameters of the Connect_WIFI() method to our own WIFI name and password (preferably in pure English)

```
if __name__ == "__main__":
    print("[WIFI] 连接网络中 Connecting wifi ...")
    isConnected = Connect_WIFI("Your_SSID", "Your_PASSWORD")
    if isConnected:
        print("[WIFI] 连接网络成功 Connect successfully")
    else:
        import sys
        print("[WIFI] 连接网络失败! 请检查配置 Error Please check your wifi 
information")
        time.sleep_ms(10)
        sys.exit()

    print("[RTSP] 启动中 .. | Starting ...")
    time.sleep(1)
```

2. Modify _do_rtsp_stream()

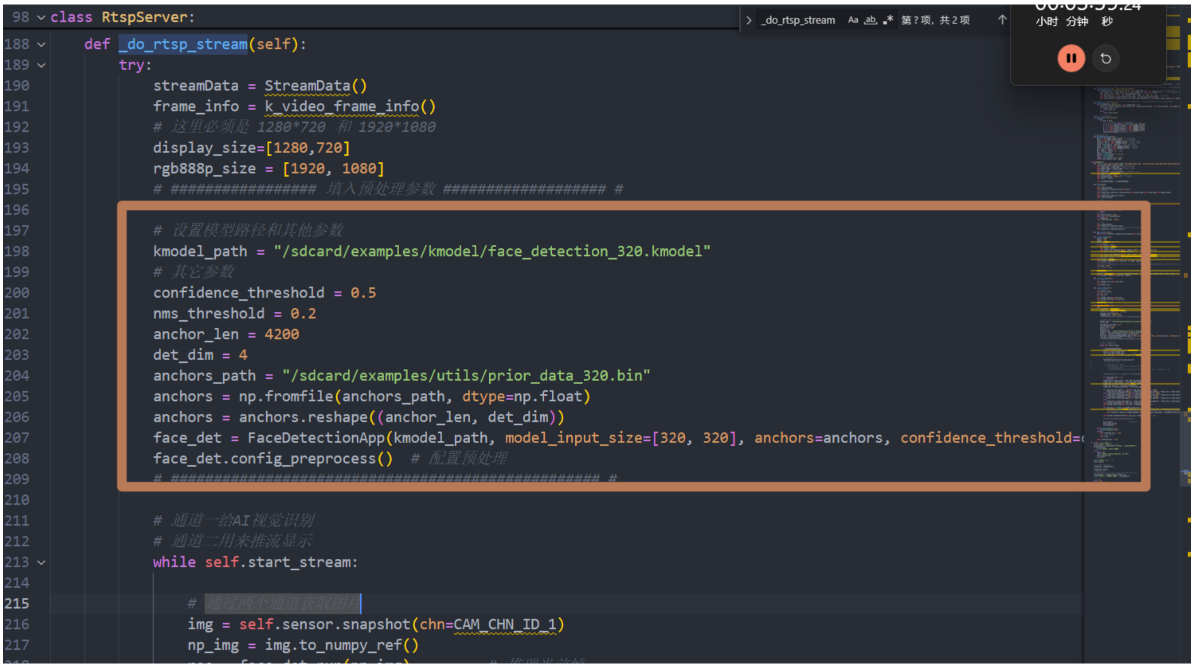
After copying, we open face_landmark.py, find the exce_demo() method, and copy the part in the red box (all the code before try-while)



```
def exce_demo(pl):
    '''
    执行演示
    Execute demonstration

    参数/Parameters:
        pl: 图像处理/Image processing pipeline
    '''
    global flm

    # 获取显示参数/Get display parameters
    display_mode = pl.display_mode
    rgb888p_size = pl.rgb888p_size
    display_size = pl.display_size

    # 加载模型和配置/Load models and configurations
    face_det_kmodel_path="/sdcard/kmodel/face_detection_320.kmodel"
    face_landmark_kmodel_path="/sdcard/kmodel/face_landmark.kmodel"
    anchors_path="/sdcard/utils/prior_data_320.bin"
    face_det_input_size=[320,320]
    face_landmark_input_size=[192,192]
    confidence_threshold=0.5
    nms_threshold=0.2
    anchor_len=4200
    det_dim=4
    anchors = np.fromfile(anchors_path, dtype=np.float)
    anchors = anchors.reshape((anchor_len,det_dim))

    # 创建人脸关键点检测实例/Create facial landmark detection instance
    flm=FaceLandMark(face_det_kmodel_path,face_landmark_kmodel_path,det_input_size=face_det_input_size,landmark_input_si
```

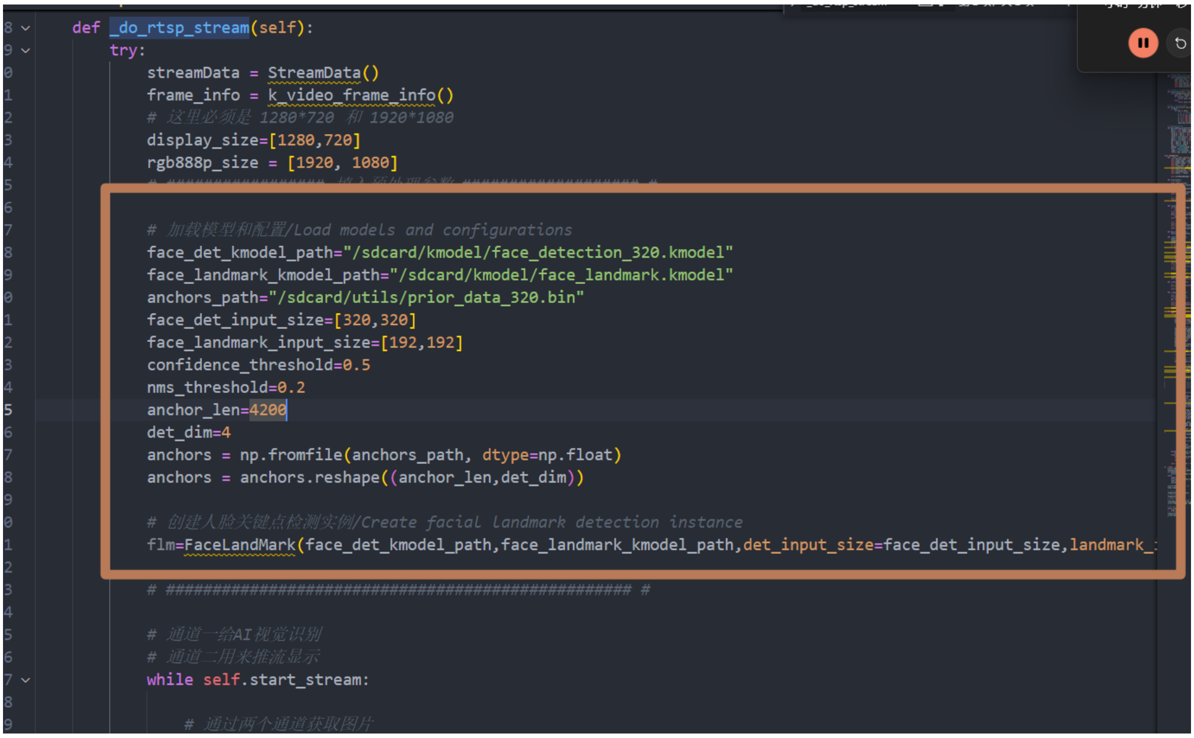
Open the newly created [empty_rtsp_demo_face_landmark.py] file and find the [_do_rtsp_stream()] method

Replace the content in the brown box below with the content in the red box we copied



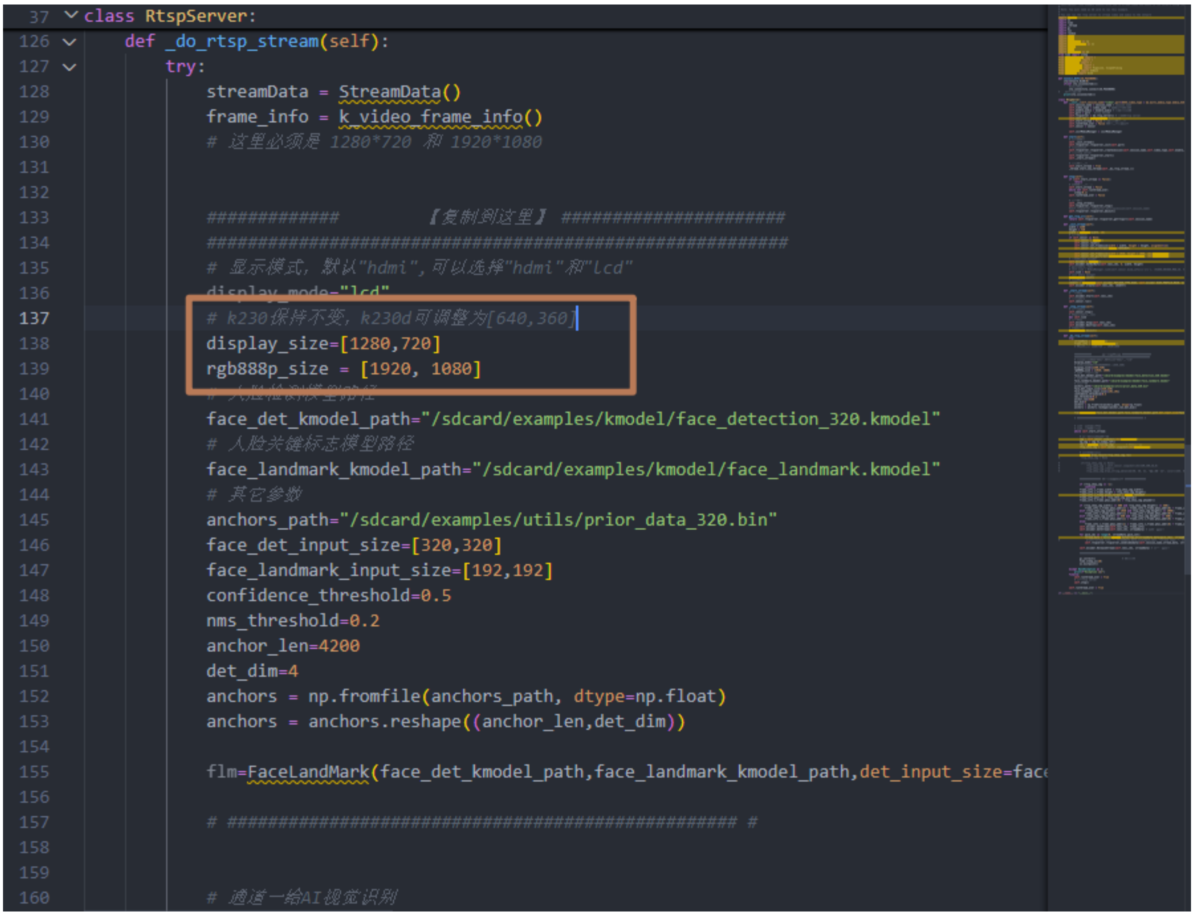
```
class RtspServer:
    def _do_rtsp_stream(self):
        try:
            streamData = StreamData()
            frame_info = k_video_frame_info()
            # 这里必须是 1280*720 和 1920*1080
            display_size=[1280,720]
            rgb888p_size = [1920, 1080]
            # ################# 填入预处理参数 ################### #

            # 设置模型路径和其他参数
            kmodel_path = "/sdcard/examples/kmodel/face_detection_320.kmodel"
            # 其它参数
            confidence_threshold = 0.5
            nms_threshold = 0.2
            anchor_len = 4200
            det_dim = 4
            anchors_path = "/sdcard/examples/utils/prior_data_320.bin"
            anchors = np.fromfile(anchors_path, dtype=np.float)
            anchors = anchors.reshape((anchor_len, det_dim))
            face_det = FaceDetectionApp(kmodel_path, model_input_size=[320, 320], anchors=anchors, confidence_threshold=
            face_det.config_preprocess()  # 配置预处理
            # ################################################## #

            # 通道一给AI视觉识别
            # 通道二用来推流显示
            while self.start_stream:

                # 通过两个通道获取图片
                img = self.sensor.snapshot(chn=CAM_CHN_ID_1)
                np_img = img.to_numpy_ref()
```

The replaced code is as follows



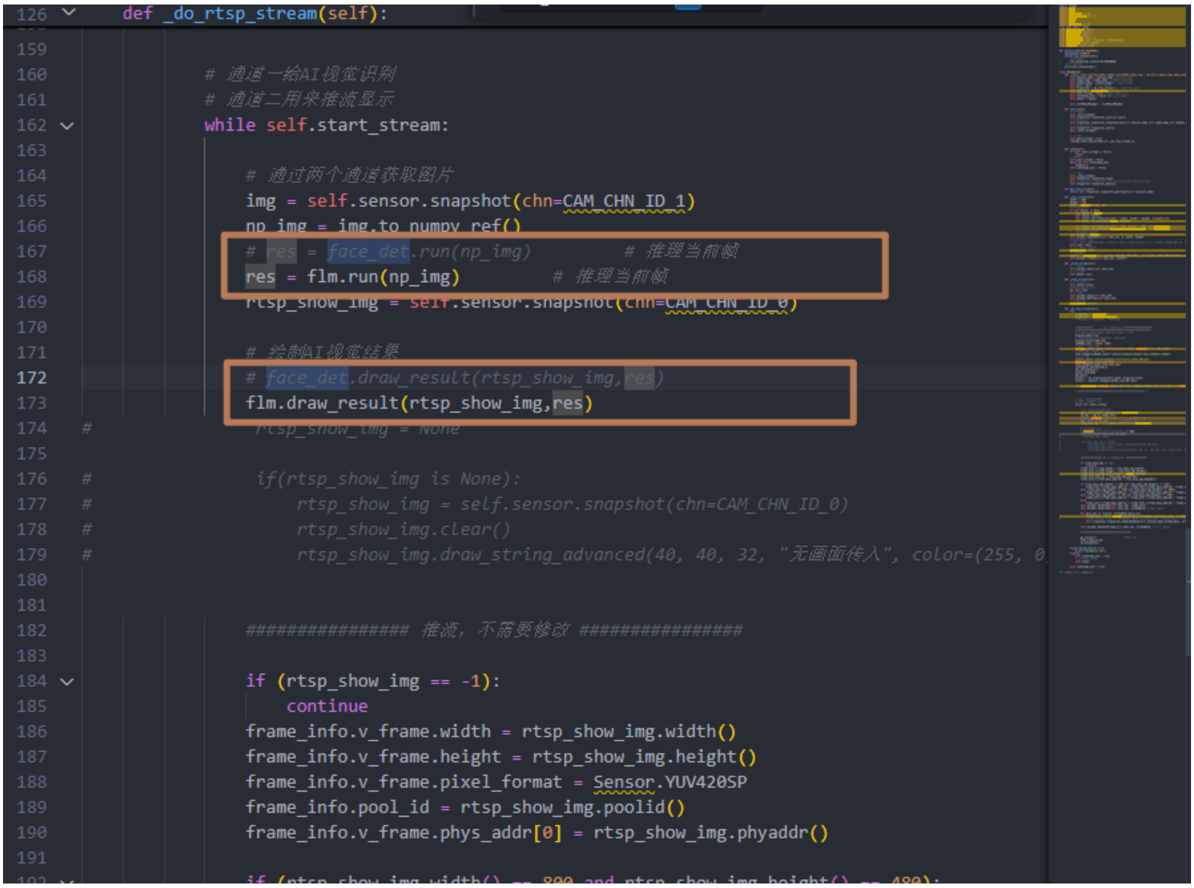
```
    def _do_rtsp_stream(self):
        try:
            streamData = StreamData()
            frame_info = k_video_frame_info()
            # 这里必须是 1280*720 和 1920*1080
            display_size=[1280,720]
            rgb888p_size = [1920, 1080]


            # 加载模型和配置/Load models and configurations
            face_det_kmodel_path="/sdcard/kmodel/face_detection_320.kmodel"
            face_landmark_kmodel_path="/sdcard/kmodel/face_landmark.kmodel"
            anchors_path="/sdcard/utils/prior_data_320.bin"
            face_det_input_size=[320,320]
            face_landmark_input_size=[192,192]
            confidence_threshold=0.5
            nms_threshold=0.2
            anchor_len=4200
            det_dim=4
            anchors = np.fromfile(anchors_path, dtype=np.float)
            anchors = anchors.reshape((anchor_len,det_dim))

            # 创建人脸关键点检测实例/Create facial landmark detection instance
            flm=FaceLandMark(face_det_kmodel_path,face_landmark_kmodel_path,det_input_size=face_det_input_size,landmark_

            # ################################################## #

            # 通道一给AI视觉识别
            # 通道二用来推流显示
            while self.start_stream:

                # 通过两个通道获取图片
```

Modify [display_size] to [1280,720], the modified result is as follows



```python
class RtspServer:
    def _do_rtsp_stream(self):
        try:
            streamData = StreamData()
            frame_info = k_video_frame_info()
            # 这里必须是 1280*720 和 1920*1080


            #############          【复制到这里】 ######################
            ##############################################################
            # 显示模式，默认"hdmi",可以选择"hdmi"和"lcd"
            display_mode="lcd"
            # k230保持不变，k230d可调整为[640,360]
            display_size=[1280,720]
            rgb888p_size = [1920, 1080]
            # 人脸检测模型路径
            face_det_kmodel_path="/sdcard/examples/kmodel/face_detection_320.kmodel"
            # 人脸关键标志模型路径
            face_landmark_kmodel_path="/sdcard/examples/kmodel/face_landmark.kmodel"
            # 其它参数
            anchors_path="/sdcard/examples/utils/prior_data_320.bin"
            face_det_input_size=[320,320]
            face_landmark_input_size=[192,192]
            confidence_threshold=0.5
            nms_threshold=0.2
            anchor_len=4200
            det_dim=4
            anchors = np.fromfile(anchors_path, dtype=np.float)
            anchors = anchors.reshape((anchor_len,det_dim))

            flm=FaceLandMark(face_det_kmodel_path,face_landmark_kmodel_path,det_input_size=face

            # ################################################## #


            # 通道一给AI视觉识别
```

The following code also needs to be modified. We change [face_det] to [flm] (the commented-out part in the figure is before the modification) (refer to exce_demo() for modification. This modification is a preliminary modification and will be further modified later)



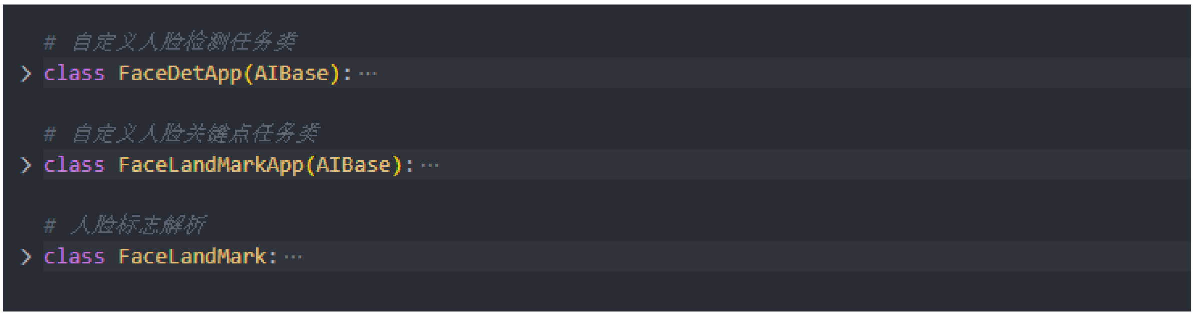
```python
    def _do_rtsp_stream(self):

            # 通道一给AI视觉识别
            # 通道二用来推流显示
            while self.start_stream:

                # 通过两个通道获取图片
                img = self.sensor.snapshot(chn=CAM_CHN_ID_1)
                np_img = img.to_numpy_ref()
                # res = face_det.run(np_img)          # 推理当前帧
                res = flm.run(np_img)          # 推理当前帧
                rtsp_show_img = self.sensor.snapshot(chn=CAM_CHN_ID_0)

                # 绘制AI视觉结果
                # face_det.draw_result(rtsp_show_img,res)
                flm.draw_result(rtsp_show_img,res)
#                rtsp_show_img = None

#                if(rtsp_show_img is None):
#                    rtsp_show_img = self.sensor.snapshot(chn=CAM_CHN_ID_0)
#                    rtsp_show_img.clear()
#                    rtsp_show_img.draw_string_advanced(40, 40, 32, "无画面传入", color=(255, 0


                ################ 推流，不需要修改 ################

                if (rtsp_show_img == -1):
                    continue
                frame_info.v_frame.width = rtsp_show_img.width()
                frame_info.v_frame.height = rtsp_show_img.height()
                frame_info.v_frame.pixel_format = Sensor.YUV420SP
                frame_info.pool_id = rtsp_show_img.poolid()
                frame_info.v_frame.phys_addr[0] = rtsp_show_img.phyaddr()

                if (rtsp_show_img.width() == 800 and rtsp_show_img.height() == 480):
```

At this point, the modification of the [_do_rtsp_stream()] part is completed

3. Copy FaceLandMark and related classes

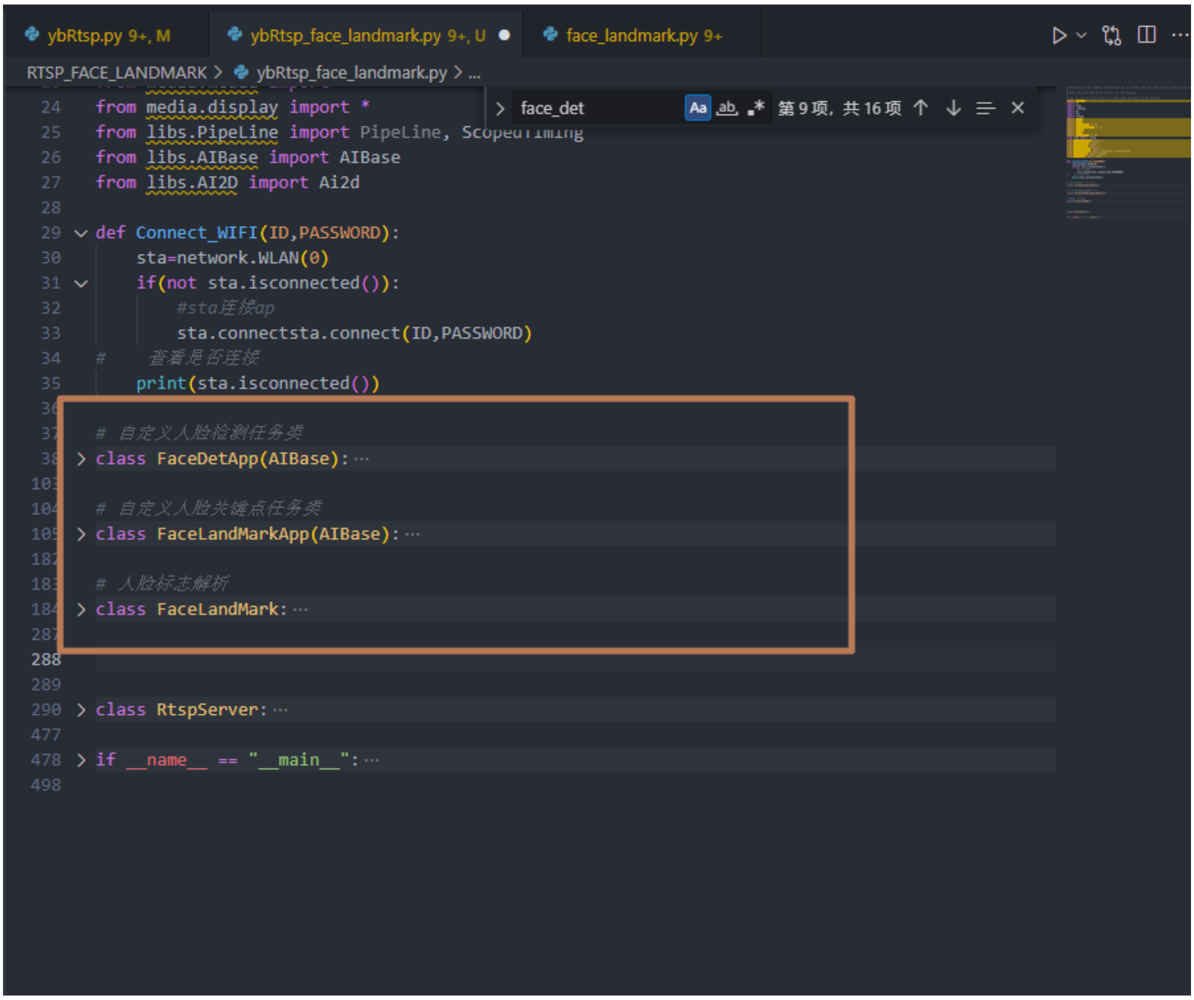
In this section, we use three classes to achieve AI visual effects, so we need to copy all three classes.

> Because the code is too long, I used folding. We need to copy all the codes of these three classes.



```
# 自定义人脸检测任务类
class FaceDetApp(AIBase):···

# 自定义人脸关键点任务类
class FaceLandMarkApp(AIBase):···

# 人脸标志解析
class FaceLandMark:···
```

After copying, we paste it into the [empty_rtsp_demo_face_landmark.py] file

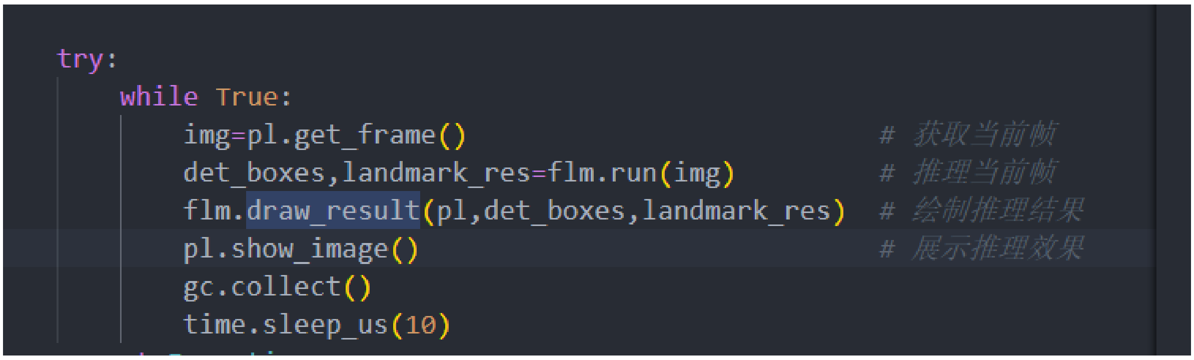


ybRtsp.py 9+, M
ybRtsp_face_landmark.py 9+, U
face_landmark.py 9+
RTSP_FACE_LANDMARK > ybRtsp_face_landmark.py > ...
face_det
Aa
ab
.*
第 9 项，共 16 项

```
from media.display import *
from libs.PipeLine import PipeLine, ScopedTiming
from libs.AIBase import AIBase
from libs.AI2D import Ai2d

def Connect_WIFI(ID,PASSWORD):
    sta=network.WLAN(0)
    if(not sta.isconnected()):
        #sta连接ap
        sta.connectsta.connect(ID,PASSWORD)
#    查看是否连接
    print(sta.isconnected())

# 自定义人脸检测任务类
class FaceDetApp(AIBase):···

# 自定义人脸关键点任务类
class FaceLandMarkApp(AIBase):···

# 人脸标志解析
class FaceLandMark:···


class RtspServer:···

if __name__ == "__main__":···
```

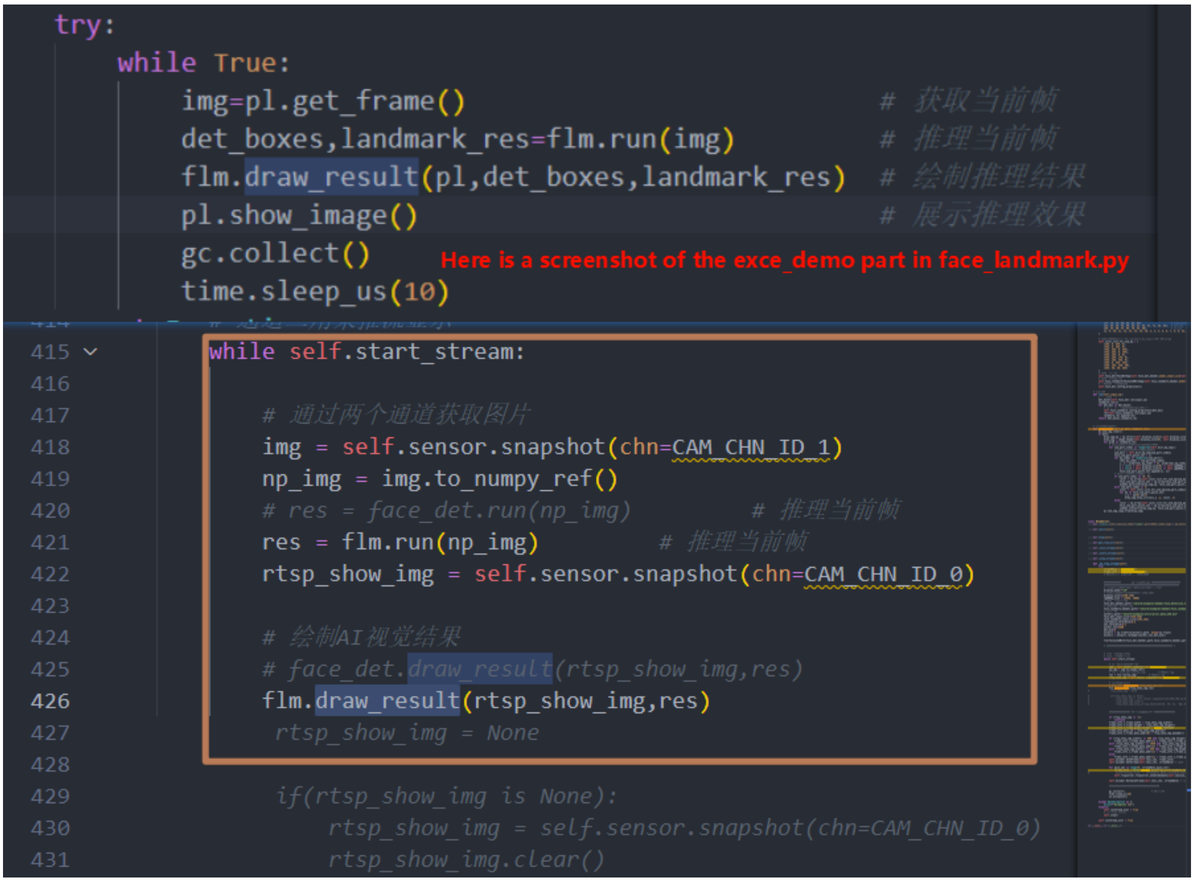
Let's go back to the [face_landmark.py] file and find the while loop in exce_demo

Now we need to transplant this part into the RTSP streaming process. For easy viewing, I put this part of the screenshot on the side.



```
try:
    while True:
        img=pl.get_frame()                              # 获取当前帧
        det_boxes,landmark_res=flm.run(img)             # 推理当前帧
        flm.draw_result(pl,det_boxes,landmark_res)      # 绘制推理结果
        pl.show_image()                                 # 展示推理效果
        gc.collect()
        time.sleep_us(10)
```

Then we go back to [empty_rtsp_demo_face_landmark.py] and find the brown box part

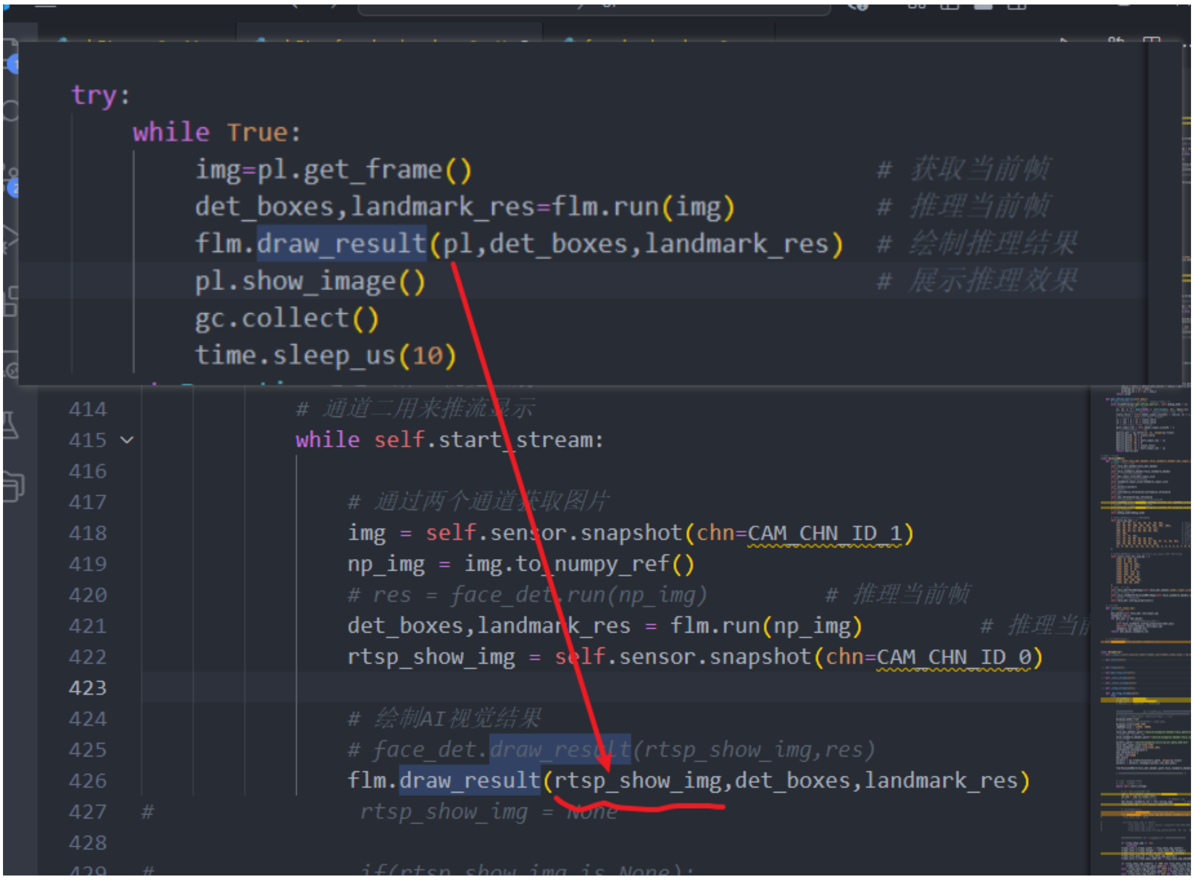


```
try:
    while True:
        img=pl.get_frame()                              # 获取当前帧
        det_boxes,landmark_res=flm.run(img)             # 推理当前帧
        flm.draw_result(pl,det_boxes,landmark_res)      # 绘制推理结果
        pl.show_image()                                 # 展示推理效果
        gc.collect()
        time.sleep_us(10)
```

Here is a screenshot of the exce_demo part in face_landmark.py

```
while self.start_stream:

    # 通过两个通道获取图片
    img = self.sensor.snapshot(chn=CAM_CHN_ID_1)
    np_img = img.to_numpy_ref()
    # res = face_det.run(np_img)          # 推理当前帧
    res = flm.run(np_img)           # 推理当前帧
    rtsp_show_img = self.sensor.snapshot(chn=CAM_CHN_ID_0)

    # 绘制AI视觉结果
    # face_det.draw_result(rtsp_show_img,res)
    flm.draw_result(rtsp_show_img,res)
     rtsp_show_img = None

     if(rtsp_show_img is None):
         rtsp_show_img = self.sensor.snapshot(chn=CAM_CHN_ID_0)
         rtsp_show_img.clear()
```

We modify the part in the brown box according to the writing method in the screenshot. The modified code is as follows:



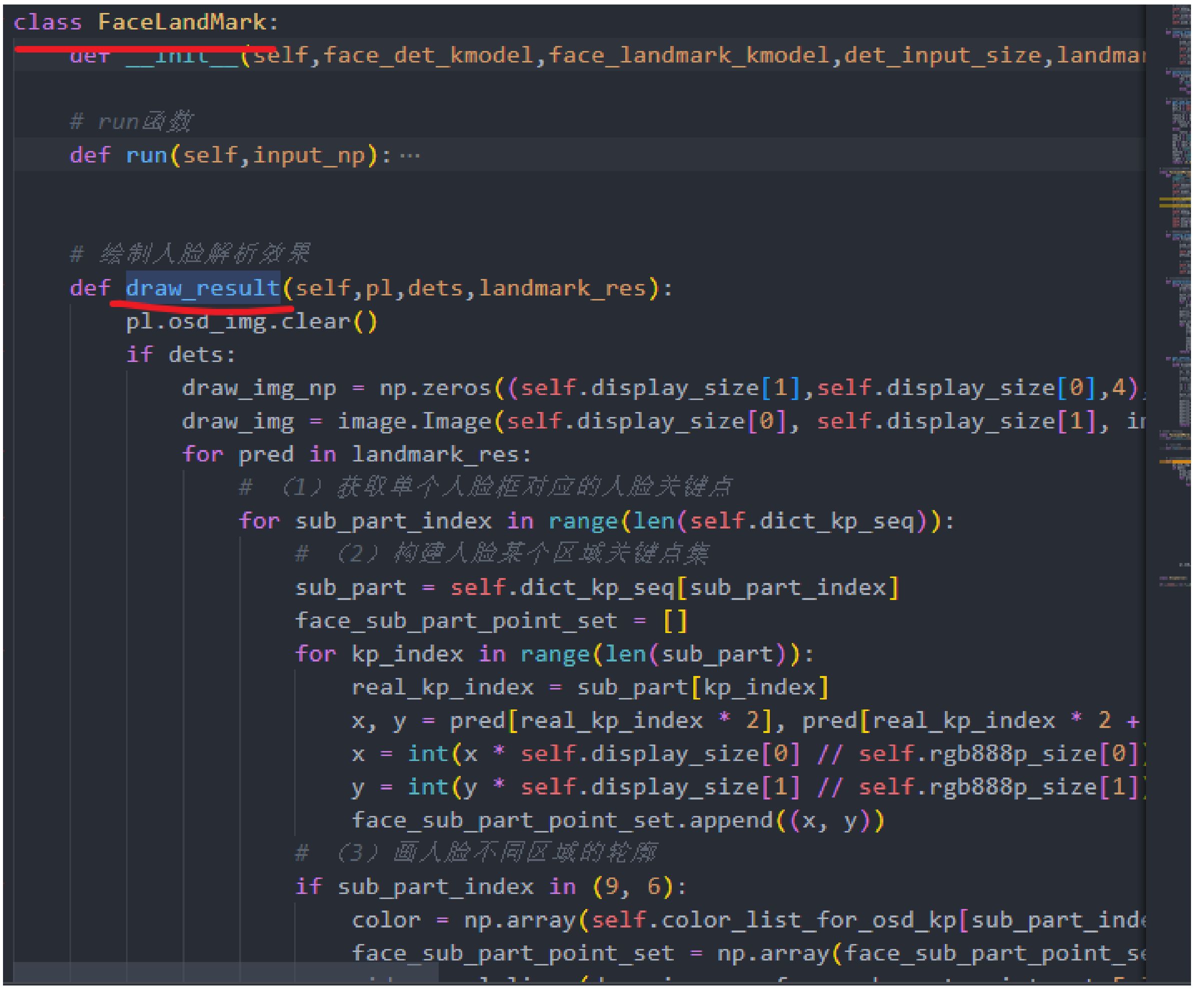
```
try:
    while True:
        img=pl.get_frame()                              # 获取当前帧
        det_boxes,landmark_res=flm.run(img)             # 推理当前帧
        flm.draw_result(pl,det_boxes,landmark_res)      # 绘制推理结果
        pl.show_image()                                 # 展示推理效果
        gc.collect()
        time.sleep_us(10)
```

```
            # 通道二用来推流显示
            while self.start_stream:

                # 通过两个通道获取图片
                img = self.sensor.snapshot(chn=CAM_CHN_ID_1)
                np_img = img.to_numpy_ref()
                # res = face_det.run(np_img)          # 推理当前帧
                det_boxes,landmark_res = flm.run(np_img)          # 推理当
                rtsp_show_img = self.sensor.snapshot(chn=CAM_CHN_ID_0)

                # 绘制AI视觉结果
                # face_det.draw_result(rtsp_show_img,res)
                flm.draw_result(rtsp_show_img,det_boxes,landmark_res)
#                rtsp_show_img = None

#                if(rtsp_show_img is None):
```

> res = flm.run(np_img) changed to det_boxes,landmark_res = flm.run(np_img)
>
> flm.draw_result(rtsp_show_img, res) changed to flm.draw_result(rtsp_show_img,det_boxes,landmark_res)
>
> **Note that pl should be changed to rtsp_show_img**

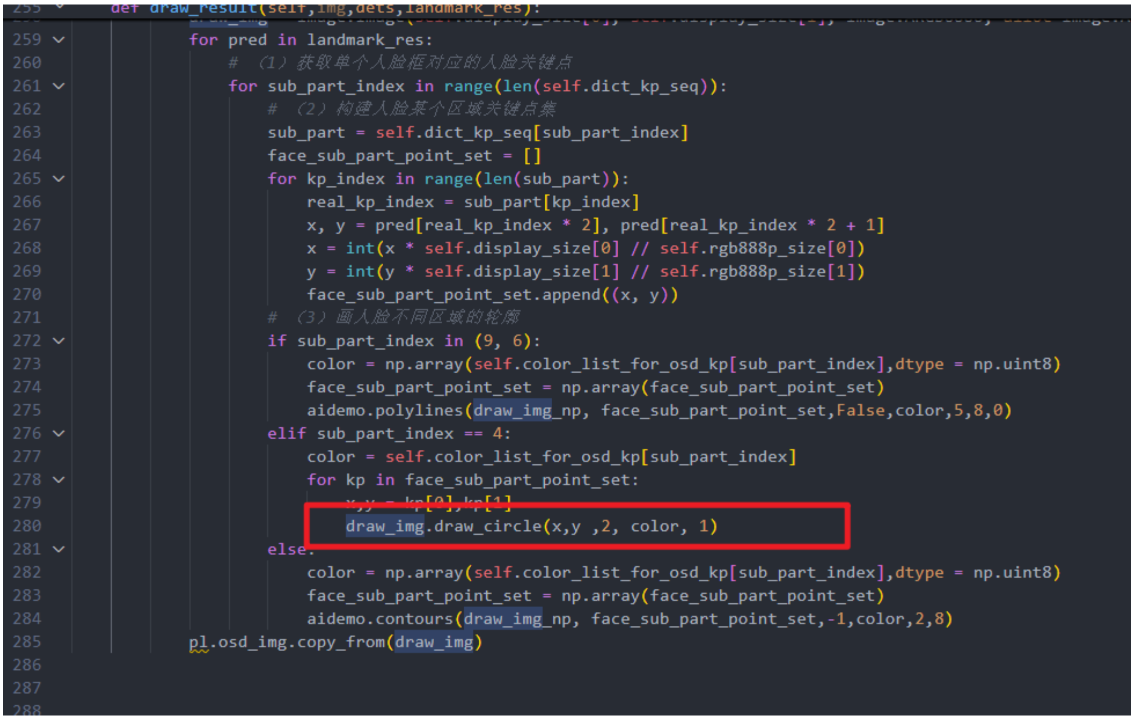
4. Modify FaceLandMark and related classes

We find the [draw_result] method in the [FaceLandMark] class

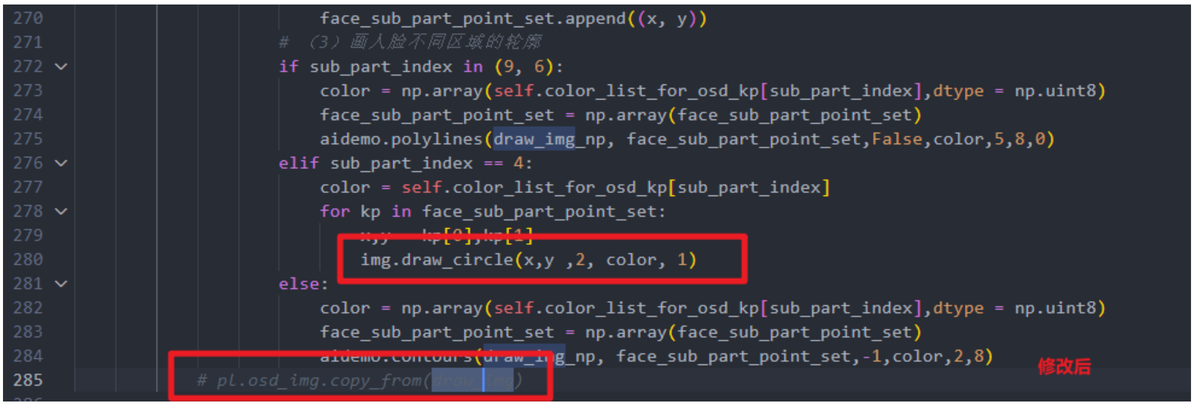


```
class FaceLandMark:
    def __init__(self,face_det_kmodel,face_landmark_kmodel,det_input_size,landmar

    # run函数
    def run(self,input_np):⋯


    # 绘制人脸解析效果
    def draw_result(self,pl,dets,landmark_res):
        pl.osd_img.clear()
        if dets:
            draw_img_np = np.zeros((self.display_size[1],self.display_size[0],4)
            draw_img = image.Image(self.display_size[0], self.display_size[1], ir
            for pred in landmark_res:
                # （1）获取单个人脸框对应的人脸关键点
                for sub_part_index in range(len(self.dict_kp_seq)):
                    # （2）构建人脸某个区域关键点集
                    sub_part = self.dict_kp_seq[sub_part_index]
                    face_sub_part_point_set = []
                    for kp_index in range(len(sub_part)):
                        real_kp_index = sub_part[kp_index]
                        x, y = pred[real_kp_index * 2], pred[real_kp_index * 2 +
                        x = int(x * self.display_size[0] // self.rgb888p_size[0]
                        y = int(y * self.display_size[1] // self.rgb888p_size[1]
                        face_sub_part_point_set.append((x, y))
                    # （3）画人脸不同区域的轮廓
                    if sub_part_index in (9, 6):
                        color = np.array(self.color_list_for_osd_kp[sub_part_inde
                        face_sub_part_point_set = np.array(face_sub_part_point_se
```

1. Modify the pl in the parameter to img
2. Delete the line pl.osd_img.clear()
3. Find the line draw_img.draw_circle(), change it to img.draw_circle(), and delete line 285



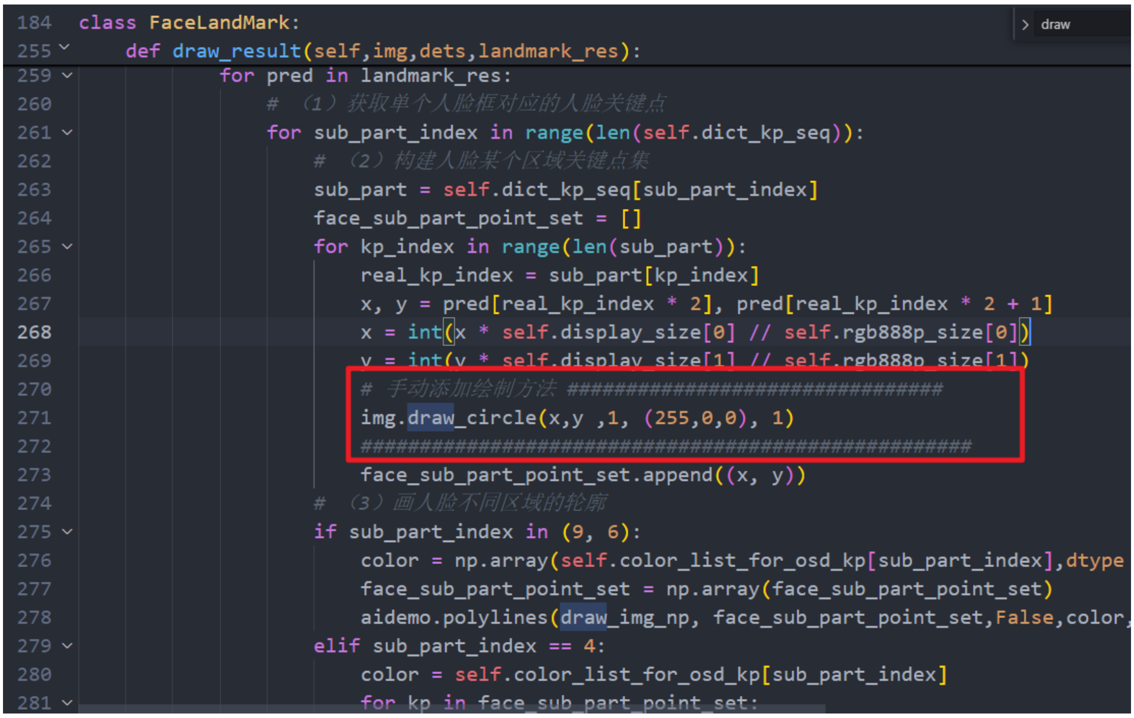
```
255     def draw_result(self,img,dets,landmark_res):
259             for pred in landmark_res:
260                 # （1）获取单个人脸框对应的人脸关键点
261                 for sub_part_index in range(len(self.dict_kp_seq)):
262                     # （2）构建人脸某个区域关键点集
263                     sub_part = self.dict_kp_seq[sub_part_index]
264                     face_sub_part_point_set = []
265                     for kp_index in range(len(sub_part)):
266                         real_kp_index = sub_part[kp_index]
267                         x, y = pred[real_kp_index * 2], pred[real_kp_index * 2 + 1]
268                         x = int(x * self.display_size[0] // self.rgb888p_size[0])
269                         y = int(y * self.display_size[1] // self.rgb888p_size[1])
270                         face_sub_part_point_set.append((x, y))
271                     # （3）画人脸不同区域的轮廓
272                     if sub_part_index in (9, 6):
273                         color = np.array(self.color_list_for_osd_kp[sub_part_index],dtype = np.uint8)
274                         face_sub_part_point_set = np.array(face_sub_part_point_set)
275                         aidemo.polylines(draw_img_np, face_sub_part_point_set,False,color,5,8,0)
276                     elif sub_part_index == 4:
277                         color = self.color_list_for_osd_kp[sub_part_index]
278                         for kp in face_sub_part_point_set:
279                             x,y = kp[0],kp[1]
280                             draw_img.draw_circle(x,y ,2, color, 1)
281                     else:
282                         color = np.array(self.color_list_for_osd_kp[sub_part_index],dtype = np.uint8)
283                         face_sub_part_point_set = np.array(face_sub_part_point_set)
284                         aidemo.contours(draw_img_np, face_sub_part_point_set,-1,color,2,8)
285             pl.osd_img.copy_from(draw_img)
286
287
288
```



```
                    face_sub_part_point_set.append((x, y))
                #  （3）画人脸不同区域的轮廓
                if sub_part_index in (9, 6):
                    color = np.array(self.color_list_for_osd_kp[sub_part_index],dtype = np.uint8)
                    face_sub_part_point_set = np.array(face_sub_part_point_set)
                    aidemo.polylines(draw_img_np, face_sub_part_point_set,False,color,5,8,0)
                elif sub_part_index == 4:
                    color = self.color_list_for_osd_kp[sub_part_index]
                    for kp in face_sub_part_point_set:
                        x,y = kp[0],kp[1]
                        img.draw_circle(x,y ,2, color, 1)
                else:
                    color = np.array(self.color_list_for_osd_kp[sub_part_index],dtype = np.uint8)
                    face_sub_part_point_set = np.array(face_sub_part_point_set)
                    aidemo.contours(draw_img_np, face_sub_part_point_set,-1,color,2,8)
        # pl.osd_img.copy_from(draw_img)
```

> This requires a case-by-case discussion
>
> 1. If the routine draws very few lines (such as face detection or face recognition), then the image is most likely drawn directly on img [the key feature is the appearance of methods such as img.draw_xxxx()]
>
>   In this case, you can change it here
>
> 2. If the routine draws a lot of lines (such as drawing key points of a face), in addition to the direct draw_Xxxx(), there may also be drawing functions such as [aidemo.xxxx()]. In this case, we need to flexibly modify the code. The idea is to directly obtain the recognized key points, and then manually add a method to draw on the image.

1. This example draws a complex image, in which the original code of the eye part uses the draw_circle method. We can directly follow step 3 [find the line draw_img.draw_circle() and modify it to img.draw_circle()]. However, the aidemo.polylines() and aidemo.contours() methods are used at other key points. The parameters of these two methods must be the nparray converted from the rgb888 type of image, which does not match the image parameter img we passed in (img is in Yuv420sp format, and RTSP real-time image transmission can only use images in this format), so we need to add a drawing function ourselves. In this example, after obtaining the key points, we use the draw_circle method to manually draw these key points



```
class FaceLandMark:
    def draw_result(self,img,dets,landmark_res):
            for pred in landmark_res:
                #  （1）获取单个人脸框对应的人脸关键点
                for sub_part_index in range(len(self.dict_kp_seq)):
                    #  （2）构建人脸某个区域关键点集
                    sub_part = self.dict_kp_seq[sub_part_index]
                    face_sub_part_point_set = []
                    for kp_index in range(len(sub_part)):
                        real_kp_index = sub_part[kp_index]
                        x, y = pred[real_kp_index * 2], pred[real_kp_index * 2 + 1]
                        x = int(x * self.display_size[0] // self.rgb888p_size[0])
                        y = int(y * self.display_size[1] // self.rgb888p_size[1])
                        # 手动添加绘制方法 ##################################
                        img.draw_circle(x,y ,1, (255,0,0), 1)
                        #########################################################
                        face_sub_part_point_set.append((x, y))
                    #  （3）画人脸不同区域的轮廓
                    if sub_part_index in (9, 6):
                        color = np.array(self.color_list_for_osd_kp[sub_part_index],dtype
                        face_sub_part_point_set = np.array(face_sub_part_point_set)
                        aidemo.polylines(draw_img_np, face_sub_part_point_set,False,color,
                    elif sub_part_index == 4:
                        color = self.color_list_for_osd_kp[sub_part_index]
                        for kp in face_sub_part_point_set:
```

Notes [Must Read]

Q: Why does my K230 CanMV IDE show that [RTSP] is started successfully and there is address information, but the video cannot be connected?

1. If the address is 0.0.0.0, there is probably a problem with the network connection. Please check whether you can connect to WIFI correctly.
2. Please make sure that the computer or mobile phone you want to watch remotely is connected to the same WIFI as K230 (or in the same LAN)
3. After testing, only two or less devices can be connected at the same time. Any more than this number cannot be connected.
4. After eliminating the above situations, if you still cannot connect to RTSP, please disconnect the K230 from the power supply and press the RST reset button. Wait for 10 seconds and then power on again to run the program.

Q: Why does the color I draw not match what I set?

A: Because the current img drawing method is not compatible with the image format used for streaming (Yuv420sp), the colors in the RGB format cannot be parsed normally. There is currently no good solution to this problem.

Q: Why do I draw multiple rectangles on the image?

A: Please make sure that display_size is set to [1280x720]

Q: I experience high latency when watching videos on the mobile version of VLC for Android?

A: This is determined by the player settings. We did not find any relevant settings in this app.